Machine learning approaches for drug virtual screening

Benoit Playe

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à Mines ParisTech

Méthodes d'apprentissage statistique pour le criblage virtuel de médicament

Machine learning approaches for drug virtual screening

Soutenue par

Benoit Playe

Le 02 Juillet 2019

École doctorale n°621

Ingénierie des Systèmes, Matériaux, Mécanique, Energétique

Spécialité

bio-informatiques

Composition du jury :

Yoshihiro Yamanishi Professor, Kyushu Institute of Technology	*Président du jury*
Christophe Ambroise Professeur des universités, Université d'Évry Val d'Essonne	*Rapporteur*
Dragos Horvath Directeur de recherche, CNRS Université de Strasbourg	*Rapporteur*
Laurent Jacob Chargé de recherche, CNRS Université Lyon 1	*Examinateur*
Véronique Stoven Professeure, Mines ParisTech	*Directrice de thèse*

MINES ParisTech | PSL

Contents

Part I

Context

Chapter 1

Drug-target interaction prediction as a guide for therapeutic research

Abstract: *The overall process of drug development is currently estimated to require on average about 1.8 billion US dollars over nearly 13 years. With the technological breakthrough of high-throughput screening technologies in past decades, increased availability of substantial amounts of bio-activity data enable the use of statistical approaches, especially machine learning. This chapter provides an overview of the rational drug discovery process, put in its historical context. Moreover, this chapter motivates the development of bio-activity prediction approaches, and reviews the prospects and challenges in bio-activity data analysis. Finally, it overviews the contributions of the thesis work in this research area.*

Résumé: *On estime actuellement que le processus de développement de médicaments nécessite environ 1,8 milliard de dollars américains sur environ 13 ans. Avec la percée des technologies de criblage à haut débit au cours des dernières décennies, la disponibilité d'une quantité substantielle de données de bioactivité pour la recherche thérapeutique a motivé la tendance actuelle à utiliser des outils informatiques, en particulier l'apprentissage statistique à l'ère de la science des données. Ce chapitre fournit une vue d'ensemble du processus de découverte de médicaments replacé dans son contexte historique, passe en revue les perspectives et les défis de l'analyse des données de bioactivité chimique et résume les contributions du travail de cette thèse dans ce domaine de recherche.*

1.1 Introduction to therapeutic research

1.1.1 Historical perspectives on drug discovery

A drug is, in essence, any substance with curative or preventive properties with regards to human and animal diseases.

For centuries, drugs were natural products, discovered by chance with no known mechanism of action, and not always industrially synthesizable. Drug discovery was essentially driven by serendipity. In other words, the discovery of drugs was a "happy accident".

Historically, the observation of micro-organisms by van Leeuwenhoek with the first microscope in 1670 and their connection with diseases allowed Pasteur to raise the idea of a drug as a molecule with a biological mechanism of action. Indeed, at the beginning of the twentieth century, Ehrlich stated in "Principle of Chemotherapy" that a convenient dose of a chemical can interfere with the growth of the microorganisms causing a disease.

This "principle" became a reality in the 1930s when Fleming discovered penicillin. Following this discovery, significant efforts in synthesising or isolating compounds similar to penicillin lead to the development of antibiotic families. Starting from the 1950s, the understanding of the expressed human genome, aided with seminal progress in cell and molecular biology, enabled rational drug design. In this new paradigm, biologists formulate rational assumptions about an appropriate mechanism explaining diseases. Thus, drug discovery became less serendipitous and more systematic since we now focus on associating diseases with specific proteins of biological pathways in order to target them with organic compounds. Finding diseases-associated targets and their ligands is the goal of rational drug discovery.

Finally, following the AIDS epidemic in the 1980s, a considerable research effort resulted in the development of combinatorial chemistry, enabling the synthesis of large and relatively diverse sets of organic compounds to be assayed for activity against specific diseases.

Nowadays, even if chance still plays a significant role in drug development, recent technologies and discovery breakthroughs enable to rationalise the drug discovery process. The work presented in this thesis is essentially intended to improve the rational drug discovery process.

We introduce rationalised drug design in the next section.

1.1.2 The modern drug discovery process

The current paradigm in rationalised drug design is to associate a disease to one or several molecular targets, usually proteins called target proteins, or therapeutic targets, involved in biological pathways necessary to the disease development. Therefore, targets are identified by a fundamental understanding of the disease after long-term study and are usually validated by in-vitro/in-vivo experiments such as gene-knocking.

The goal of the drug discovery process is then to identify a small molecular compound that binds the target protein in such a way that it alters the disease development. This small molecule must also be able

to reach the protein target through the organism without triggering undesired deleterious side-effects.

A sketch of the drug discovery process is displayed in Fig. 1.1. Once a protein responsible for the disease and a ligand molecule have been identified (the hit), usually thanks to in vitro assays, the process is still complex. Indeed, long and costly pre-clinical and clinical assays are performed at the final stage of the development, in order to prove the effect of the drug and satisfy the regulatory agencies requirements. The overall process of drug development was estimated in 2003 to require between 4 to 7 years and a billion US dollars on average [1]. Single-drug development is now estimated to require about 1.8 billion US dollars over about 13 years on average [2].

Figure 1.1. Scheme of the rational drug discovery and development process (picture from [3])

The five main steps of the drug discovery process are listed below.

- **Identification of the protein target**: In the cell, a large number of biological pathways made of cascades of reactions involving proteins such as transcription factors, receptors, or enzymes lead to the observed phenotype.

  Once a protein target has been identified, its inhibition or activation by interaction with a small organic compound is expected to disturb one of several pathways and reverse the disease phenotype. For instance, aspirin is known to inhibit the activity of the cyclo-oxygenases COX-1 and COX-2 involved in the inflammatory response pathway.

  According to the above definition, any protein could in principle be targeted. Based on drugs currently available, the druggable genome is restricted mainly to a few families of proteins. Indeed, half of the known human' targets are G protein-coupled receptor (GPCRs), a family of cell membrane proteins involved in signal transduction. Ion channels, i.e. membrane proteins that regulate the flow of ions across membranes, constitute the second largest group of known drug targets.

- **Identification of "hits"**: A second step is to identify a small molecular compound that inhibits (or activates) the protein target, and alters its function and that of the downstream pathway. In general, a molecule that binds to a protein and modulates its function is called a *ligand*. In the context of drug identification, the compounds initially identified as active against the target are referred to as *hits*. Note that, in this manuscript, we will use the terms "molecule" and "compound" to refer to chemical substances.

  Furthermore, we often mention the "interaction" between a ligand and its target. In terms of "activity", it is a continuous value quantifying the target activity alteration triggered by the ligand, as evaluated

by a biological assay. For instance, the activity of a ligand against an enzymatic target is measured as the concentration at which it decreases the catalytic action of the target by 50%, which is also called IC50. This aspect is further discussed in Section 1.3.

In practice, hits are often identified via high-throughput screening (HTS) experiments. These are large assays which are conceived with a simple readout, to allow massive parallel screening of hundreds of thousands of drugs in a day. HTS experiments were enabled by technological breakthroughs in miniaturisation and robotics that perform the bio-assays in parallel, as well as combinatorial chemistry for the generation of large molecular libraries built from elementary blocks.

In the pharmaceutical industry, a "classical" HTS campaign usually consists in the screening of 1-2 million compounds. It is usually followed by a dose-response assay to assess the level of bio-activity of drug candidates by evaluating their modulation of the target activity. Identified "hits" are also screened in a so-called orthogonal assay, where another mechanistically different screening technology is used to weed out compounds that happen to interfere with the screening technology.

- **Characterisation of the hits**: A good level of bio-activity is not enough, and clinical candidate molecules must meet a set of different criteria. Any potential drug must have an excellent biological profile (in particular low toxicity) and excellent pharmacokinetic properties. Instead of pharmacokinetic, experts mainly use the terms Absorption, Distribution, Metabolism, Elimination (ADME) which characterise the ability of a hit to reach its target protein and then be safely eliminated by the body. Indeed, the accumulation of foreign substances is likely to be toxic and lead to deleterious side effects.

  The structures of hits are then optimised in order to fulfil the required toxicity properties without losing the activity. Overall, converting a hit molecule into a drug candidate is a complex multidimensional optimisation process. At the end of this optimisation, the candidate molecule is called a lead.

- **Synthesis**: Once a lead is identified, finding a way to synthesise it at the industrial scale is a challenging problem. When the lead is a natural product, extraction from its natural source might limit the quantities available. Thus, synthesising the molecule from commercially available molecule building blocks is often preferable, although many molecules are also synthesised by biotechnologies.

- **Clinical validation**: The next step consists in validation of the drug in three stages of clinical trials, whose results are judged by the legal agencies (the Food and Drug Administration -FDA- in the USA and European Agency for the Evaluation of Medical Products -EMEA- in Europe).

  In phase I, the drug is tested on healthy volunteers (or sometimes in patients who have no therapeutic alternative), to assess its toxicity in humans. Phase II aims at testing various doses of the drug to estimate the therapeutic dose. Phase III is a double-blind assay, which aims at comparing the efficiency of the new drug to that of a placebo, or of a reference treatment. Each of these three phases usually requires several years.

Despite all the advances in understanding diseases, and technological breakthroughs, this rational drug discovery process has limited success, and only a few tens of new drugs reach the market every year.

The main bottleneck of the drug design process is the toxicity issue. Most of the hits identified by HTS fail to become approved drugs because of unexpected side effects and toxicity, which are in part due to unexpected interactions of the drug with off-target proteins. While HTS helps to increase the number of tested molecules to identify hits, it is not possible to conduct bio-assays at the human proteome scale [4, 5]. In addition, more than 6 million drug-like compounds are commercially available, but the drug-like molecule space is estimated to be around 10^{60} [6]. Randomly exploring this space is out of reach even by HTS, and identifying a molecule in such a vast space relies mainly on human expertise. Overall, HTS experiments are expensive, time-consuming, and require advanced laboratories with access to chemical and biological libraries. This aspect of HTS allows large pharmaceutical companies to take a monopolistic position.

In that context, computational approaches appear as promising ways to facilitate the tedious task of drug discovery. In particular, the word "chemoinformatics" appears in 1998 [7] as "the application of computational techniques to the field of chemistry". In other words, chemoinformatics can be defined as the application of informatics to chemical problems [8]. It entails two main issues: the numerical description of molecules, prior knowledge about the problem at hand, and algorithms to analyse databases and search for compounds of interest. The thesis presented in this manuscript mainly tackles these two issues.

Large databases of molecules, proteins and recorded bio-activities are freely available and maintained. Indeed, tens of millions of compounds are recorded in databases together with their known bio-activities [9, 10], in which about 9,000 FDA-approved small drug molecules are present [11]. In addition, about 550,000 reviewed protein records are available, including 20,244 human proteins [12] with nearly 2,700 known to be targeted by either approved or experimental drugs [9, 12].

Chemoinformatics algorithms can explore this knowledge and provide in silico screening methods that help and guide the drug discovery process. The topic of this thesis is devoted to in silico screening methods.

We introduce basic concepts in computational approaches for molecular bioactivity prediction in the next section.

1.2 Molecular bioactivity prediction to assist the modern drug discovery process

The identification of therapeutic targets relies on the overall knowledge about the considered disease. Several approaches have been developed to assist target identification. Differential gene expression analysis (which can also be transcribed into differential biological pathway expression analysis) aims at detecting genes whose levels of expression are the most perturbed in the patients compared to healthy controls. Such differential analysis can also be performed on other types of data qualifying gene "activities" such as copy number variation, mutation status, or epigenetic factors like methylation. In chapter 5, we will provide and example of how such data can be fruitfully combined in the context of breast cancer.

Once the target is identified, a panoply of computational methods have been proposed to help identify ligands.

1.2.1 Computational approaches for molecular bioactivity prediction

One of the main methods is "docking" [13, 14, 15]. This technique is a 3D-structure based approach in which the binding energy between a small molecule and the target protein is estimated according to a set of molecular mechanics equations that model the physical interactions between the protein binding pocket and the ligand. Docking methods are usually composed of two algorithms, a search algorithm positioning the ligand within the pocket, and a scoring function that evaluates the strength of the interaction. This score is then used to rank the molecules.

The application of docking is limited by the fact that the 3D-structure of the target is required, which is usually not the case for many important protein targets such as membrane receptors. These 3D-structures are obtained experimentally by either X-ray crystallography which requires a crystallised form of the protein, or Nuclear magnetic resonance which is essentially limited to proteins of small sizes. Therefore, in practice, docking cannot be applied at a large scale in the protein space, although it can be used at large scale in the chemical space.

More generally, in order to be applicable at the proteome scale, prediction methods need to be independent from 3D-structural data. Another approach is to develop more integrative approaches, taking into account widely available data such as target protein sequences, drug chemical structures, and known drug-target network information.

Such a methodology is primarily inspired by the field of machine learning (ML). Given a learning data set, machine learning methods discover some underlying rule about the data and provide a "model" (or a "predictor") that can be subsequently used. More precisely, if n *examples* (or equivalently *data points*, or *observations*, or *samples*) are given, these n points $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}$ form the *data set*. All data points are described by p-dimensional vectors $\boldsymbol{x}^i \in \mathbb{R}^p, i \in \{1, \ldots, n\}$. The p attributes describing the data samples are called *features*, or equivalently *descriptors*, *variables* or *attributes*.

In the case where the predicted output is a real value (for instance, the affinity of an interaction), the ML predictor is called a regression model. For a categorical predicted output (for example, a binary statement: interaction or not-interaction), the ML predictor is called a classification model, or simply a *classifier*.

Fortunately, the amount of data and information produced by chemical research has grown large enough so that applying ML methods to predict interactions between chemicals and proteins at a genome scale becomes possible. "Drug-target interaction" (DTI) prediction, or equivalently drugs "virtual screening" (VS), can be viewed as the classification of protein/molecule couples as interacting or non-interacting pairs.

For the purposes of this thesis, I extensively trained in machine learning and felt the need to synthesise this knowledge in order to mobilise it quickly and easily. The machine learning concepts required throughout this thesis are introduced in appendix A.

1.2.2 Virtual screening assists the drug discovery process at various stages

Both docking and statistical approaches can directly help the drug discovery process at each of its stages. However, in this thesis, we did not perform docking studies, because we only considered methods that did not required the 3D structure of the protein. This choice is justified in our case, since the goal is to make large scale predictions in the chemical and biological spaces, to address the questions of drug specificity, drug repositioning, or target identification.

We introduce in the following how searching the chemical space using automation and algorithms helps to discover novel molecules, reactions and mechanisms by guiding the intuition of the human expert [16]. Examples of such applications are given in chapter 5.

First, they can help to identify compounds that would bind to the considered therapeutic target. Such methods are called ligand-based drug virtual screening approaches, or Quantitative Structure-Activity Relationship (QSAR) methods [17]. These approaches use a set of compounds for which the activity has been experimentally measured, and they relate the structure of molecules to their biological activities.

The current trend is to predict ligand-protein interactions in the spirit of chemogenomics, which can be thought of as a generalisation of the ligand-based approach where the prediction models for several targets are learned simultaneously [18]. In chemogenomics, prediction of the drug biological activity for one target is expected to benefit from the activities also known for other molecules and other targets.

Taking place before the experimental high-throughput screening, drug virtual screening can help to eliminate the unlikely drug-target pairs and to test only potentially active combinations, thus reducing the cost and time required for HTS. Virtual screening can be seen as a virtual wet-lab experiment that can test at low cost a colossal number of arbitrarily dissimilar compounds, including molecules which have not been synthesised yet.

Some contributions [19] focused on sampling the chemical space as efficiently as possible, in order to reduce both cost and time of the hit generation process itself. To this end, they quantitatively identified parts of the chemical space in which the biological property of interest is especially unknown, based on the disagreement between an ensemble of predictors. From a complete other perspective, Swamidass et al. [20] even proposed an economic modelling to improve the efficiency of expensive and time-consuming HTS experiments, by explicitly encoding costs and utility in an economic framework.

Overall, virtual screening has been reported to be useful in practice. For instance, it helped to discover of a novel inhibitor of AmpC β-lactamase [21]. Others have shown that a VS approach improved the hit rate of a high-throughput screening experiment from 0.021% to 34.8% [22].

Second, computational approaches can also help in the optimisation of a hit molecule to increase its activity. Quantitative Structure-Property Relationship (QSPR) tackles the problem by predicting biological activities of compounds, i.e. their affinity value to a given protein via docking analysis. The predicted activities are used to rank compounds. Indeed, if the 3D-structure of the target is known, docking experiments may be a more efficient way to score the interaction between molecules against the considered protein binding pocket. Analysis of the docking poses can also help optimisation of the hits structures in order to favour

stabilising interactions.

Third, drug virtual screening can be used to predict drug likeliness (identified as the main bottleneck in the drug discovery process), which refers to the pharmacokinetics, toxicity properties, or the ADME criteria.

The work by Lipinski et al. [23] proposed the so-called 'rule of five', which links physical properties of molecules to their ability to become a drug [23]. These rules are commonly used as a filter to remove compounds from the pipeline. Similar rules have been derived in the context of toxicity relying on the detection of chemical groups known to cause toxicity.

However, undesired side effects of drugs are more generally due to binding with proteins other than the intended target, called "off-targets" proteins. Identification of these off-targets is thus crucial to prevent the potential side-effects, since compounds interacting with only one target protein are exceptional [24]. Indeed, a ligand for a protein is expected to have high probability to bind to proteins of the same family, but it may also bind to other evolutionary distant proteins if convergent evolution occurred at the binding sites. Experimental approaches devoted to identification of all potential off-target is clearly out of reach, because this would require to design experimental assays for all possible proteins.

Thus, genome-wide virtual screening, which consists of predicting the interaction of a set of molecules against the entire proteome, tackles the problem of drug specificity, which is related to side-effect prediction, through the prediction of "off-targets". Indeed, many studies, like the work of Zhou et al.[25], showed strong correlation between drug side-effects and predicted targets. This is critical since side-effects are the main cause of failure in the drug development process, and of market withdrawal.

Fourth, virtual screening can also be used for drug repurposing (or repositioning) and off-target effect identification.

Indeed, although unexpected effects caused by off-target interactions are often unwanted and harmful, they can sometimes be beneficial and have led to new therapeutic indications for drugs. A well known example is Sildenafil (known as Viagra), initially developed to treat angina, but which was found to extend penile erection in human volunteers, leading to a change in the therapeutic indication of the drug [26].

Many drug candidates fail in latest phases of clinical trials. In the same time, there are already thousands of FDA-approved drugs on the market. Therefore, the prediction of their overall protein target profile to propose new indications could result in saving lot of time [27, 28]. In particular, Zhou et al. [25] present some successful examples of drugs repurposing via the prediction of off-targets, both in common and rare diseases. Some studies have also focused on prioritising small molecule as candidates for drug repositioning using machine learning [29]. Up to now, we described how virtual screening, as computational algorithms that identify the properties of molecules, can help to perform more cost-effective research in combination with experimental work.

Fifth, from the opposite point of view, molecular design approaches perform a reverse search of chemical space. They start from a desired property and search for a molecular structure expected to fulfil this requirement. In the last two years, a substantial research effort has been devoted to develop generative

models for "in silico" building of drug-like compounds with a desired property [30]. We introduce and discuss these models in more details in appendix E.

In this section, we presented the main situations in which computational approaches may assist pharmaceutical research. In the next section, we present the main bottlenecks encountered in this context.

1.3 Challenges in molecular bioactivity prediction

In this section we discuss important issues in chemogenomics that we later tackle in this manuscript.

Data availability

Even if more than 100 million small molecules are recorded in molecular databases [9] (discovered in living systems or synthesised by chemists), the number of small molecules that could be synthesised is much greater (recall that the chemical space is infinite, and that the size of the pharmacologically relevant space is about 10^{60} [6]). However, only a relatively small portion of the chemical space of molecules has been explored and assayed, so that available biological properties are limited.

Machine learning, and in particular deep learning (see appendix A.8), tend to benefit from the increasing amount of available data in public databases such as the ChEMBL [10] and PubChem [9] databases.

One of the main issues in predicting biological activity of a given molecule is its similarity to molecules for which the desired property is known. This aspect is considered in detail in chapter 4.

Furthermore, the ability to perform prediction over the pharmacologically relevant space is related to the generalisation properties to unseen data of the algorithm that is used. In the context of statistical approaches, regularisation methods are usually considered in order to improve the generalisation of machine learning models. We define these methods in more detail in appendix A.1, and we often refer to them throughout this manuscript. In addition, other approaches can be considered to help the prediction on unexplored subspaces, such as "semi-supervised" approaches that try to leverage both labelled and unlabelled data. It is also possible to consider data augmentation methods that expand available data using natural or artificially generated data, via an appropriate noise or other manipulations of the original data. In this thesis work, we are to investigate these approaches too.

Data diversity

The diversity of the synthesised chemical subspace is restrained, but also biased because pharmaceutical studies are conducted with different intentions than sampling the chemical space uniformly. Indeed, drug discovery projects usually focus on molecules close to the known active molecules, for legitimate reasons such as costs, synthetic feasibility and availability in a chemical library [31].

In particular, Cleves et al. [32] showed that active ligands of a specific target that are highly similar in terms of structure are patented in a narrow time span, whereas more structurally dissimilar ligands tend to be discovered years later.

Consequently, this sampling bias in the chemical space is present in the databases and may lead to overestimate the performance of methods which use them for training and testing, a problem usually mentioned as the inductive bias.

However, even if the predictive power is limited by the data statistics, it does not mean that methods based on the available chemical data cannot be used in practice. However, we must care about the validation procedures and benchmark data sets to evaluate and compare properly different methods. In particular, it is likely that many reference datasets used in the chemoinformatics literature, distort predictions evaluation because the train and test samples are too similar (for example, when the same scaffolds are shared by the active compounds in the training and test set). In this case, the model performance is biased to specific regions of the molecular space, leading to an insufficient evaluation of the generalisation ability.

This bias is considered with special care in chapter 4 of this manuscript.

Qualitative interactions and quantified interactions

As already mentioned in Section 1.1.2, drug-target interactions are actually continuous experimental values, but there is no rational and no consensus on a threshold activity value to assume an interaction for "real".

In most cases, the "IC50" molecular value is considered. In most studies about drug identification, molecules with IC50 values of $1\mu M$ or lower are accepted as active. However, recorded activity values tend to be rather above micromolar concentrations, and the threshold is usually relaxed to $10\mu M$ to increase the number of interactions available to train models, which comes with the cost of considering noisy data.

Since "negative" results are unappealing and high IC50 values are not desirable, activities over $100\mu M$ are usually not reported in databases. As a consequence, there are very few instances of "true negatives" that can be used as negative training instances. This lead us to discuss the next issue of chemogenomics.

"Negative" interactions are actually unknown interactions

One problem in DTI prediction lies in the scarcity of known drug-protein interactions compared to the myriad of unknown interactions.

If the unknown interactions are let "unknown", the problem of DTI prediction falls in the framework of Positive-unlabelled learning (PU learning). PU learning consists in finding new positive examples from a collection of positive and unlabelled examples, from which a prediction model is built.

This field has been mainly explored for text classification problems, in which positively annotated documents are mixed with a few negatively annotated documents, and many unlabelled other texts.

Three types of approaches can be employed in such situations. First, one-class classifier can be used [33]. However, the one-class classifier was found not to perform as well as binary classification-based methods for text classification document [34, 35]. Second, some approaches focus on identifying, with near certainty, a few negative examples from the unlabelled data, so that standard machine learning classification methods can be applied [36]. Third, other approaches directly apply slightly modified standard methods while considering the unlabelled data points as negative, in order to transform a PU-learning problem into a classification problem.

Because molecules are expected to interact with a restricted number of proteins compared to the overall protein diversity, most studies consider that protein/molecule couples for which no interaction is recorded do not interact (hereafter called "negative interactions"). In this case, the number of negative interactions is dramatically larger than that of positive interactions, because the number of known positive pairs is more limited.

This issue is generally mentioned as the class imbalance problem, which means that there are many more negative samples (negative DTI) than positive samples (known positive DTI). Class imbalance has a significant impact on the choice of the prediction method, but also on the metric and evaluation procedure used (as presented in appendix A.2).

Then, a widely accepted solution is to randomly select a subset of unknown interactions as "negative" interactions (i.e. "non-interactions") to recover a balanced problem. This strategy assumes that the ratio of interacting to non-interacting pairs is so low that random selection would yield a high quality set of negatives, i.e. where very few negatives would be in fact unknown positives.

However, with this solution, the information contained in the unlabelled data points that were not used is lost. Some algorithms, like the bias-SVM algorithm, can to a certain extent deal with such imbalanced data [37, 38], and use larger training sets. We tackle the issue of negative interactions in chapter 4 and chapter 8.

Most classifiers predictions are not chemically or biologically interpretable

Most chemogenomic methods are predictive but do not provide any clue about the mechanisms of the interactions, or about which feature mainly guided the prediction, which does not favour easy interpretation of the results. A challenging issue in recent chemogenomics research is to identify the underlying associations between drug chemical substructures and protein functional involved in ligand-protein interactions.

In particular, ML methods typically suffer from the confirmation bias [39], which refers to tendency of humans to confirm a hypothesis by searching for a correlation of the hypothesis with the outcome. The confirmation bias is related to the non-causal bias, which refers to the case in which there is correlation but no causal relation. Even if a model achieve good predictions, detected correlations may not be physically founded which may lead to false conclusions.

In the case of virtual screening, the hypothesis assuming that molecules similar to known active molecules are also active can be confirmed if one only look at the high performance of the models built on this hypothesis. But this hypothesis is, in general, wrong due to the well-known "activity cliffs", which refers to the fact that a small modification in the molecular structure or the protein binding pocket can strongly change the nature of the interaction. In addition, highly different molecules may bind to the same protein, either in the same or in different binding pockets [40].

Among others, Yamanishi et al. [41] investigated this problem by applying the Sparse Canonical Correspondence Analysis (SCCA) to DTI prediction, as presented in more details in the dedicated Section 3. Random Forests also allow, to a certain extent, interpretation of the predictions, based on the most often selected features. finally, in the context of deep learning-based models, attention mechanisms (cf. appendix A.8.2) also enable the interpretation of individual predictions, as presented in more details in section 7.2.

Most methods do not allow integration of heterogeneous data types

Different data sources are likely to contain heterogeneous and thus partly independent information about the prediction task. Thus, integrating these complementary data sources is expected to increase both the quality and the coverage of DTI predictions. However, most VS methods do not allow to integrate heterogeneous data in a direct manner.

These aspects are discussed both in the context of kernel-based methods in section 2.2 and appendix B, and in the context of deep learning in chapter 8.

Databases do no distinguish direct and indirect interactions

Different relationships can describe drug-target pairs. For instance, their interaction may be direct or indirect. In addition, one could also consider the following three types of drugs modes of action: activation, inhibition or (just) binding. Therefore, *types* of DTIs can represent various drug-target relationships and drug modes of action. Besides, interactions recorded for example in the DrugBank database sometimes refer to indirect interactions via intermediate partners, although this is not specified (we found such examples in the cystic fibrosis project 5.1, and ignored these indirect interaction for training the prediction models). Therefore, knowing the type of interaction that has been recorded could be useful when they are used to build a prediction model, but also to understand the underlying drug's mechanism of action. Indeed, depending on the effect that is searched, one might want to activate or inhibit a protein, and therefore, known DTIs for this target might be used as positives or negatives, which obviously has a considerable impact on the model that is trained [42].

Predicting the type of DTIs corresponds to solve a multi-class classification problem, instead of a two classes (active/non active) problem. For such questions, the MATADOR database [43] is particularly suited as it specifies the type of each recorded interaction. To our knowledge, only the work of Wang and Zeng [42],

who used Boltzmann machine as predictors, addressed this type of concerns (see chapter 2).

DTI prediction is not directly transposable in terms of biological assays

As already mentioned in Section 1.1.2, HTS projects are typically performed in multiple stages. First, a single dose screen is executed to identify "hits", which are active compounds at a high dose. The second stage evaluates the potency of a small subset of hits by a more accurate "dose-response" experiment to identify the "true" hits. Those are finally sent for further analysis. This will be illustrated in chapter 5, in the triple negative breast cancer project.

The majority of the molecule library, as much as 99% [44], is dropped after the first stage and then is not further characterised. Most of it is expected to be inactive. However, some of these compounds may be potent in the drug-response stage.

Even if ranking DTI according to prediction scores helps identifying the most probable negatives, developing tools that are tuned to send only accurate hits for the "dose-response" experiments is of great interest.

"Active learning" deals with those situations where unlabelled data is abundant and labelling is expensive. Given a set of labelled points and a set of unlabelled points, active learning seeks to find the samples to be labelled in order to build, at a lower cost, the most efficient classifier to further assign labels to all points with high confidence. Active learning is therefore particularly suited for addressing the HTS hits selection issue, as already investigated in multiple studies [45, 46, 47, 48].

Another original approach has been reported by Joshua Swamidass' team. They argue that the selection of hits which should be sent to the dose-response stage is essentially economic. Indeed the experimental design itself implies an economic constraint: resource limitation is the only reason justifying that all compounds are not tested in the second stage. Neglecting the local economic context may, therefore, lead to irrational experimental strategies. By closely working with the Broad Institute's HTS team, they developed an economic model to improve the efficiency of expensive and time-consuming HTS experiments [20].

To conclude, collaboration with experimental scientists is essential to produce useful DTI predictions tools that enable more efficient discovery.

1.4 Contributions of the thesis

The thesis is conceived following two lines of approaches intended to address proteome-wide drug virtual screening from the methodological standpoint of machine learning.

In this domain, global performances of baseline methods on available drug-target interaction prediction datasets are high. However, they are averaged over all tasks, masking poor performance on the most challenging, although critical tasks.

A long-term objective of this study is to thereafter propose an ML approach for drug-target interaction (DTI) prediction, with a particular focus on hard drug-target interaction prediction settings. Overall, this manuscript is divided in three parts and appendices.

In part I (which contains the present Introduction chapter), chapter 2 gives some insight about how compounds and proteins are represented to be handled by ML algorithms, and about publicly available databases and toolkits used for DTI prediction. The machine learning basics required in this thesis can be found in appendix A. These are imperative prerequisites before starting investigating drug virtual screening with this type of approaches.

Then, the organisation of the thesis follows the two methodological lines of approaches, respectively presented in part II and part III.

Part II (chapters 3, 4, 5): The first line of ideas to perform DTI prediction at the proteome level is to build ML approaches that can leverage the molecule and protein descriptors and similarity measures that are defined based on expert knowledge. We refer to such expert-based attribute vectors as "data-blinded features", because they are defined a priori, and independently from the ML algorithm. Note that the choice of data-blinded features can be task-specific, since toxicity prediction and DTI prediction may benefit from different expert-based descriptors.

Chapters 3 and 4, focus on proposing computationally attractive data-blinded approaches for drug-target interaction prediction at the proteome level. In particular, it is well-known that kernel methods have found many successful applications where the input data are discrete and structured, including strings (e.g. protein sequences) and graphs (e.g. molecular graphs) [49]. Based on kernels, many off-the-shelf kernel machines are available to solve proteome-wide DTI prediction. Our observations lead us to propose the *NN-MT* algorithm, a multitask SVM for chemogenomics, usually limited in practice by computational times, that is trained on a dataset of a size similar to those used by single-task methods. Chapter 3 presents the main ML algorithms used in chemogenomics, and chapter 4 is devoted to the *NN-MT* algorithm which mainly consists in our publication on this topic [50] in the PlosOne Computational Biology journal.

Chapter 5 illustrates how drug virtual screening assists real-life applications to guide therapeutic research. Contributions in this field can be numerous: they can identify drug candidates for known therapeutic targets, identify the target and the mechanism of action of drugs discovered by phenotypic tests, anticipate potential side effects by identifying secondary protein targets or suggest repositioning of known drugs in new therapeutic indications. More precisely, we discuss our contributions in a couple of drug repositioning studies for triple negative breast cancer and for cystic fibrosis, two life threatening diseases. In particular, we perform a systematic analysis to characterise some identified bio-active compounds either by their molecular structures or mechanism of actions (i.e. based on their protein target interaction profile, either known or predicted). We managed to point out, with strong evidences for both projects, drug repositioning paths that are unrelated to those currently used to treat the patients. These projects were developed in collaboration with clinicians and biologists, and they may open new therapeutic strategies for these severe diseases.

Part III (chapters 6, 7, 8): The second line of ideas to perform proteome-wide virtual screening is to build ML approaches that extract relevant data-specific numerical descriptors directly on the molecular graph and the protein sequence, while fitting the model weights to predict interactions. We call "data-driven features" such numerical descriptors extracted from the raw data in an end-to-end fashion. In the literature, we find the nomenclature "representation learning"[51] which refers, to our knowledge, to the approaches that learn representations, either on data-blinded or data-driven features, while fitting the prediction model. In particular, models that consist of stacked fully connected neuron layers (see appendix A.8) perform representation learning on data-blinded features.

In chapter 6, we first discuss the neuron network modules that encode protein sequences and molecular graphs directly from the supervised prediction task. Besides, we review the dynamic field of graph convolutional neuron network which turns to be a promising field of research to reinforce molecular graph representation learning. In parallel, in appendix C, we provide a historical perspective of graph representation learning, in particular, for chemoinformatics purposes.

In chapters 7 and 8, we start by a critical evaluation of the most relevant studies that investigated representation learning and data-driven feature encoders for drug virtual screening. This critical overview lead us to investigate two main directions to boost data-driven DTI prediction approaches. First, we explored modifications of the neuron network architecture to extract pairwise data-driven features. Second, we examined several ways to combine heterogeneous data sources. In other words, we intensively discuss data-driven feature extraction approaches for DTI prediction by a range of experiments on chemoinformatics, proteomics and chemogenomics datasets. From our observations, we propose a data-driven approach to use for proteome-wide drug virtual screening and guide promising directions to boost this field of research.

The results of chapters 7 and 8 contain the results of a manuscript currently in preparation.

Finally, in **IV**, we discuss the most promising direction, from our point of view, to enhance proteome-wide DTI prediction. In addition, we also discuss, as perspectives and in appendix E, the popular and promising field of inverse drug design. Although we did not perform experiments in this domain, we thought that it was important to expose this recent and active field of research which finds applications in DTI, and more generally in the prediction of molecule properties. These approaches start from the desired property and search for a molecular structure fulfilling this requirement. After introducing the main multiple ways to automatically generate organic compounds with the desired biological properties, we conclude by a brief comparison of these approaches.

Chapter 2

Computational representation and storage of molecules and proteins

Abstract: *In order to guide the drug discovery process with statistical approaches, we need to define three critical aspects: (i) the mathematical representation of molecules (and proteins), or equivalently, the measure of similarity between them; (ii) the model linking molecular and protein features to the predicted properties (for example the ability to bind to a protein target, the toxicity and the pharmacokinetics); (iii) the access to large learning data sets, which is a prerequisite for machine learning methods. In this chapter, we examine and discuss the mathematical description of proteins and molecules as well as the freely available databases and gold-standard datasets for a variety of chemoinformatics tasks. We review machine learning models separately in appendix A.*

Résumé: *Afin de guider le processus de découverte de médicaments avec des approches statistiques, nous devons définir au préalable trois aspects essentiels: (i) la représentation mathématique des molécules (et des protéines), ou de manière équivalente, la mesure de la similarité entre les molécules (et des protéines); (ii) le modèle statistique permettant de prédire des propriétés (comme par exemple, la capacité de liaison à une protéine cible, la toxicité et la pharmacocinétique de molécules); (iii) l'accès à de grands ensembles de base de données d'apprentissage, ce qui est un préalable indispensable aux méthodes d'apprentissage statistiques. Dans ce chapitre, nous passons en revue les descriptions numériques de protéines et de molécules ainsi que les bases de données disponibles librement et les jeux de données de référence pour une variété de tâches liées à la chimio-informatique. Nous introduisons séparément les modéles statistiques dans l'appendice A.*

The definition of protein and molecule descriptors or similarity measures, and the access to large learning datasets are necessary prerequisites before exploring drug virtual screening. The knowledgeable reader can skip this chapter. Indeed, for each of the studies that we report in this thesis, we start by briefly introducing all the required material.

Over the past decades, both computational chemistry, interested in predicting various properties or activities of molecules [52, 53], and computational biology, interested in the prediction of various structural or functional properties of proteins [54, 55] have designed descriptors and similarity measures to build various representations of the genomic and chemical spaces.

2.1 Numerical encoding for molecules and proteins

Computationally manipulating and analysing proteins or small molecules poses the problem of representing these data as vectors or, in other words, defining a set of binary or real-valued descriptors for these data and stack them to form a vector. This process of converting raw data into something more suitable for an algorithm is called *featurisation*.

The representation of the data must be suitable in a sense that it should contain information that is relevant to the considered prediction problem, in order to help the algorithm mapping the input representation to its output. The choice of representation for a specific problem is not trivial and is, in practice, the main control stick for an increase in performance.

2.1.1 Molecule numerical descriptors

The problem of explicitly representing molecules as finite-dimensional vectors have a long history in chemoinformatics and a multitude of molecular descriptors have been proposed [56]. These descriptors include physicochemical properties of the molecules, such as its solubility, descriptors derived from the 2D structure of the molecule, such as fragment counts or structural fingerprints, or descriptors extracted from the 3D structure.

In particular, the molecular featurisation should reflect the intrinsic physical and chemical properties so that a machine learning model can learn a relationship between these properties and a prediction task (in our case, interaction with a target in particular).

To be accurate, a molecular system should be modelled by solving the Schrödinger equation for the molecular electronic Hamiltonian, representing the molecule by a set of nuclear charges and the corresponding coordinates of the atomic positions in the 3D space. However, we expect a statistical model to benefit from representations that expose more easily and specifically the physical and chemical properties of interest. Note that it might also require less computational power.

In practice, molecular descriptors are derived from three types of information: the molecular structure (for instance, ECFP descriptors [57]), the interaction with the environment (the log solubility, for instance) and experimental data (drug-induced phenotypes, for instance; but that requires the molecules to be available or synthesised and tested before any predictions can be made).

Although more than a thousand different types of molecular descriptors can be found in the literature [58, 59], the most popular molecular descriptors are fingerprints. They are binary vectors of a given size, typically taken in the range of 100–1000, in which each dimension represents presence (1) or absence (0) of a particular property. Such fingerprints are usually structural fingerprints, and represent compounds by their structural fragments and functional groups.

Historically, the first substructures to be used were listed by hand. Nowadays, substructures are labelled paths built directly from the graph by a depth-first search. More precisely, the depth-first search yields a list of all the paths of a certain length, usually lower than 10, emanating from one starting node. In particular, there are the so-called keys-based fingerprints which report the presence or absence of predefined substructures in compounds (MACCs fingerprints [60], pharmacophore fingerprints [61] and others), and circular fingerprints which represent structural properties of compounds which are not pre-defined (ECFPs [57]).

As mentioned in Section 1.2, molecular compounds are mainly represented by a string notation (mainly SMILES [62] or InChI [63]) or by a graphical representation of the Lewis formula. We present in Section 2.3.3 several available tools to generate molecular feature vectors from such standard representation.

Several studies focused on comparing molecular descriptions. Duan et al. [64] and Bender et al. [65] reported that the best fingerprints to retrieve a protein-ligand interaction depends on the protein target. They also reported that different fingerprints retrieve different active compounds for a given target. All together, these observations show that molecular descriptors best represent different properties of compounds.

In particular, Alberga et al. [66] emphasised the importance of multi-fingerprint consensus strategies to increase the confidence in predictions. In general, combining multiple types of fingerprint descriptors provided the best performance, and feature selection approaches were reported to be efficient [64, 65, 67, 66].

Indeed, it is not obvious that different fingerprints capture independent space dimensions [67]. To unveil the intrinsic correlation existing among fingerprints, one can keep only one fingerprint type of a set of correlated fingerprints [66], by comparing the Tanimoto similarity profile of a million of compounds described with these fingerprints, and setting a bottom threshold of correlation.

When used individually, ECFP4 fingerprints have been reported to perform the best [65] and are widely used for virtual screening.

2.1.2 Protein numerical descriptors

Protein descriptors are usually based on different types of information.

First, we can describe a protein sequence based on its composition reflecting the occurrence frequencies of different amino acid combinations. Second, descriptors can be derived from the distribution of physicochemical properties along the sequence (for instance, hydrophobicity, van der Waals volume, polarity, polarizability, charge, secondary structure and solvent accessibility). Third, topological properties are also often considered. They characterise amino acids according to their connectivity, geometrical characteristics (shape, size, atomic positions in space, secondary structures, flexibility and solvent accessibility of proteins for instance). Fourth, functional characteristics can also be taken into account (presence/absence of functional sites for

instance).

We present in section 2.3.3 several available tools to derive descriptors from protein sequences. In particular, a web server [68] is freely available for deriving structural and physicochemical features of proteins and peptides from amino acid sequences.

Ong et al. [69] compared the efficiency of 10 commonly used sets of descriptors (AAC, DC, autocorrelation, CTD, QSO, Pse-AAC and others -presented in Section 2.3.3-) for the prediction of functional families using support vector machines. They reported that combining the descriptors provide the best results, although none of them performed significantly better than the others individually.

Up to now, we introduced numerical representations of molecules and proteins in the form of feature vectors.

However, in the case of drug virtual screening, molecules and proteins are often handled based on similarity measures following the idea that similar targets may interact with similar compounds. We discuss molecule and protein similarity measures in the next section.

2.2 Similarity measures for molecules and proteins

Kernel methods (see appendix A.5), or more generally "similarity based" methods, is a class of algorithms that offer an efficient strategy to solve the problem of data representation.

In these methods, vector representation of data is not required, as long as a similarity measure between them is defined. More formally, instead of converting each protein (resp. molecule) into a p-dimensional vector, with p potentially large, kernel methods can manipulate data only through a function K, called a kernel, that compares any proteins pair (resp. molecules pair) and returns a real number. The use of valid kernels, which must be a symmetric and positive definite matrix, ensures the equivalence between the use of kernels, on the one hand, and the use of explicit vector representations in possibly infinite dimensional spaces, on the other hand (see appendix A.5).

In the next three subsections, we review relevant work focusing on the definition of valid kernels for proteins and small molecules, to illustrate the possibilities offered by kernels to implicitly describe the "protein" and "chemical" spaces. More than an illustration, we needed to explore the potentialities of molecule and protein similarity measures when using kernel methods for chemogenomics in chapter 4.

2.2.1 Protein similarity measures

String-based similarity measures

The most straightforward representation of a protein is its sequence of amino acids, which takes the form of a string over an alphabet of 20 letters. Based on this description, one can attempt to define a list of

descriptors to characterise various features of protein sequences, and compute the inner products between the resulting vectors to obtain a kernel.

The *string kernel*, developed for text data, was used to define a protein sequence similarity measure. The string kernel, developed by Lodhi et al. [70], considers a document as a long sequence. The feature space is generated by the set of all (non-contiguous) substrings of k symbols. The more substrings two instances have in common, the more similar they are considered to be (the higher their inner product).

Following this idea, Leslie et al. [71] built the *spectrum kernel* as a similarity measure for protein sequences. It uses as descriptors the set of all possible subsequences of amino acids of a fixed length k. If two protein sequences contain many of the same k-length subsequences (k-mers), their "inner product" under the k-spectrum kernel is to be large. Several variants of this kernel have been proposed.

The *mismatch kernel* [72] first added the biologically important idea of mutagenicity by allowing some degree of mismatching between k-mers.

A later study [73] extended this idea via the notion of probabilistic profiles, which define position-dependent mutation neighbourhoods along protein sequences for inexact matching of k-mers in the data. This kernel was called the *profile kernel*. From a machine learning point of view, the use of probabilistic profiles can be viewed as a semi-supervised approach, because it computes a position-dependent profile representation of each sequence through iterative heuristic alignment with all recorded proteins.

These descriptors could be relevant to detect homologous proteins, which are likely to contain similar proportions of the various k-mers or to predict biological properties that depend on short motifs of amino acids, such as structural or binding motifs. However, proteins are synthesised linearly by the ribosome, an organelle responsible for translation of the genetic information into proteins in the cytoplasm. Therefore, protein amino acids sequences are expected to be related to their 3D folding only up to a certain extent. For instance, it is known that some proteins, called "chaperone", assist the covalent folding of other proteins. The 3D conformation of proteins assembled by chaperones proteins may not be related to the “natural” free-energy minimisation of the linear amino acids sequence occurring in the cytoplasm. Hence these descriptors may become limited when it comes to predict properties linked to the 3D structure of the protein.

Physico-chemical descriptors-based similarity measures

Alternatively, researchers have built kernel based on a set of physicochemical properties descriptors [74, 75]. As mentioned in Section 2.1, these descriptors are interesting to encode information about the variations of physicochemical properties along the sequence, e.g., to detect elements of local structures, called the *secondary structure of the protein*, such as helices or extended sheets.

Evolutionary-based similarity measures

In order to define more biologically relevant kernels, others used the notion of similarity for protein sequences developed by the computational biology community, namely the similarity based on local alignment score. Indeed, the notion of local alignments has emerged as a powerful approach to detect evolutionary relationships

between sequences. Besides, the measure of the best local alignment with the Smith-Waterman (SW) algorithm [76] or its fast heuristics BLAST, PSI-BLAST [77] and FASTA [78] are still the methods of choice to compare the sequence of two proteins.

Liao and Noble [79] first used the SW score to define a kernel, called the pairwise kernel. It is, however, not a valid kernel because it is not positive definite.

Saigo et al. [80] introduced a kernel called the Local Alignment kernel (LAkernel) that mimics the behaviour of the SW score. While the SW score only keeps the contribution of the best local alignment between two sequences to quantify their similarity, the LAkernel sums up the contributions of all possible local alignments.

It is important to note that these kernels, being based on sequence similarity, compare proteins from the evolutionary point of view. They seem relevant for drug virtual screening because a local similarity of sequence may correspond to a local similarity of binding pockets.

3D conformations-based similarity measures

Finally, when available or predicted, one can represent a protein by its 3D structure, which is likely to contain more information related to physical interactions with other proteins or ligands. While kernels for sequences have attracted more considerable attention so far, several groups have attempted to map protein 3D structures by building kernels between structures.

First, Dobson et al. [55] explicitly represented each structure by a vector made of carefully chosen features (such as secondary structure content, amino acid propensities or surface properties).

Alternatively, Keiser et al. [81] provided a statistical method to compare proteins based on the known structures of their set of ligands. This approach can be seen as a way to describe proteins by 3D-binding pockets through the negative image of their ligands 3D configurations.

Finally, others [82] have shown that a kernel derived from a measure of structure superpositions is an efficient way to relate the structure of a protein to its function. Hoffman et al. [83] then defined a kernel between two binding pockets by aligning their atoms in the 3D space and comparing the resulting configurations with a convolution kernel. They observed pockets alignments even when the corresponding proteins shared no sequence similarities, which highlights the relevance of 3D description for DTI prediction when available.

2.2.2 Graph-based chemical similarity measures

Since the 2D chemical graph is available for all molecules, it is practical to use this representation to describe molecules. A chemical graph can be interpreted as a labelled undirected graph where the atom types are the vertex labels and the bond types (single, double, triple) are the edge labels. Additional information can be easily added to this graph description. For instance, labels to model electronic properties such as electrostaticity, polarity and molecular orbitals can be assigned to atoms.

Sub-graph decomposition-based similarity measures

Comparing graphs through their decomposition into a set of subgraphs is an idea successfully developed in the graph kernel literature [84].

In the chemoinformatics literature, the most used representation of molecules is the fingerprint feature vector representation [85, 86] as mentioned in section 2.1.

Several similarity measures between fingerprints can be built, including the straightforward inner product. However, the Tanimoto similarity measure, defined as the ratio between the number of substructures shared by the two graphs over the number of all substructures considered, is widely used in chemoinformatics. This similarity measure is also a valid kernel [87].

Other kernels related to the Tanimoto kernel have been proposed. The *MinMax* kernel [87] extended the Tanimoto kernel by considering the frequencies of the substructure whereas the Tanimoto kernel is only based on their presence.

Graph walks-based similarity measures

As an alternative, several groups have investigated strategies to define kernels between molecules that do not require the explicit computation of vector representations, by directly working on the chemical graph.

In particular, Kashima et al. [88] and Gärtner et al. [89, 90] proposed simultaneously to represent the 2D structure of a molecule by an approximately infinite-dimensional vector of linear fragment counts. These kernels handle huge but still finite dimensional feature spaces and have parameters to constrain this dimensionality burden.

Kashima's *marginalised kernel* is the inner product of the count vectors averaged over all possible labelled paths, defined as random walks on graphs. The exponential and geometric kernels between graphs proposed by Gärtner et al. [49] only take into account the vertex labels at both ends of linear fragments, and use depth-first search to build the labelled paths. Compared to the marginalised kernel, it might not be optimal when contiguous labels are essential, as in chemical compounds.

Mahé et al. [91] extended the marginalised kernel by preventing random paths from going backwards. Additionally, they proposed a trick to increase the dimension of the feature space, to make the fragments more specific, while increasing the computational speed of the kernel. The trick is based on the Morgan index, which is well known in the chemoinformatics literature. It corresponds to a procedure which seeks to find a canonical 2D representation of a molecule, by successively adding information of adjacent nodes to each node. By increasing the label specificity, this procedure shortens the computation time, as the number of common label paths between graphs decreases while the relevance of the labels increase. These modifications lead to better prediction of various physicochemical properties prediction tasks.

Several groups also extended the substructures extracted from the molecular graphs beyond linear fragments. For instance, Horvath et al. [92] considered kernels based on cyclic fragments and Gärtner et al. [49] suggested the use of tree fragments instead of linear fragments. This idea was later extended by Mahé and Vert [93]. These kernels were again defined as the inner product between vectors counting cyclic-patterns or tree-patterns inside graphs. A subtree is a subgraph with an underlying tree structure (no cycles, but

a designated root node). However, Mahé et Vert noticed that the subtree kernel was outperformed by the MinMax kernel defined by Swamidass et al. [93] on several physicochemical prediction tasks.

3D structure-based similarity measures

As molecules are in fact 3D objects, it would be natural to define kernels based on the 3D structure of molecules. Mahé et al. [94] design a kernel focused on the detection of 3D pharmacophores. These are any three-dimensional arrangement of atoms, or a group of atoms, representing putative configurations responsible for the biological property of interest. In this study, 3D structures of molecules were described by a graph in which node labels are atom labels involved in pharmacophores, and edge labels are the Euclidean distance between the atoms they connect.

Jacob et al. [95] applied the 3D pharmacophore kernel on the virtual screening of G-Protein Coupled Receptor (GPCR), a super-family of proteins including many therapeutic targets. However, it did not perform as well as the 2D kernel based on Tanimoto measure of similarity.

Swamidass et al. [96] consider similarity measures between histograms of pairwise distances between atom types. For instance, the CO histogram is the histogram of the distances between all pairs of carbon and oxygen atoms in the molecule. Once again, 3D information is only brought by considering the 3D Euclidean distance between atoms.

Azencott et al. [97] defined 3D kernels by using graph approximation and other techniques from computational geometry to characterise the 3D surfaces of small molecules.

Both studies also concluded that Tanimoto based 2D kernels outperformed their designed 3D kernels. This observation can be either explained by the fact these kernels were not suited for the considered problems, or by the fact that the 3D information can be inferred from the 2D structures. Indeed, when the active conformation is not known, i.e. when the 3D structure that the molecule adopts in the context of the predicted property is unknown, using another conformation might be less efficient and represent a stronger hypothesis, than just encoding the molecule by its 2D structure.

Historical review of similarity measures for graph data

Deriving a similarity measure from graphs is a large and popular field. Indeed, graphs appear in a variety of situation, from social network analysis to programming language automatic debugging. The research for efficient and scalable kernels for graph data lead to several seminal works that we introduce here. Note that we refer to the following works later in the manuscript, in section 6.2 and appendix D, when comparing recent developments in graph representation learning to kernel methods.

Early work in the field of graph kernels was based on graph (random) walks or paths [90], and graph similarities were assessed based on the number of matching pairs of random walks or paths in two graphs. Remind that a walk is a sequence of consecutive nodes in a graph (i.e. consecutively connected by an edge)

and a path is a walk that consists of distinct nodes only. As mentioned earlier in this chapter, such method has been proposed for chemoinformatics by Kashima et al. [88] and Mahé et al. [91].

Others have proposed kernels based on shortest paths [98] on graphs, by assessing graphs' similarity based on the number of pairs of identical shortest paths in two graphs (i.e. starting from the same source and ending at the same labels, and having the same length).

A drawback of the random walk and shortest path approaches is their high computational cost, which corresponds to $O(n^2)$ and $O(n^4)$ respectively (with n being the maximum number of vertices of two graphs).

Later, graph kernels based on counting limited-sized subgraphs were proposed. This includes the so-called graphlet kernel [84], which represent graphs as counts of all types of subgraphs of size $k \in \{3, 4, 5\}$. Authors also provided efficient computation scheme based on sampling or exploitation of the low maximum degree of graphs, for unlabelled graphs only. Similarly, cyclic pattern kernel [99] counts pairs of matching cyclic patterns in two graphs, and subtree kernel [49] counts pairs of matching tree patterns in two graphs.

More recent developments in graph kernels have emphasised scalability. In particular, one of the most successful general graph kernels is the Weisfeiler-Lehman (WL) kernel [100] and its variants [101, 102, 103, 104].

The WL kernel is based on the well known 1-dimensional Weisfeiler-Leman graph isomorphism heuristic, used for deciding whether two graphs are isomorphic.

First, note that assessing graph similarity based on isomorphism is not a priori a trivial task because it is computationally expensive or even intractable on first hand. Indeed, the subgraph isomorphism checking problem (analogue of graph isomorphism checking for two graphs of different sizes) is an NP-complete problem, unsolvable in polynomial time. In addition, a similarity measure based on graph isomorphism must also encode inexact matching of graphs.

The principle of the WL algorithm is to generate features at each node, through an iterative relabelling, or colouring scheme, by aggregating over the multiset of labels (or equivalently colours) in its neighbourhood. At each iteration, the algorithm can terminate by concluding that the two graphs are not isomorphic by checking that the number of vertices with a specific label is different in both graphs.

The WL kernel does not use of the isomorphism checking operation at each step, but only the node relabelling procedure. Indeed, the WL kernel between two graphs is computed as the sum of (any) graph kernels between these two graphs, at each relabelling iteration. Thus, the WL kernel uses standard graph kernel, but on modified graphs, in which the neighbourhood substructures are summarised into the vertices labels.

Variants of the WL kernel proposed procedures to speed-up the computation time [102], a streaming version [103], and more importantly, an approach to handle continuous node and edge labels [104].

However, the discriminative power of the 1-WL has been well characterised [105]. The WL algorithm is known not to be able to distinguish all isomorphic graph. For instance, the WL algorithm gives the same "colour" to all nodes in a graph consisting of a triangle or a 4-cycle.

Therefore, the WL algorithm was generalised to k-set of vertices leading to a more powerful graph isomorphism heuristic. The neighbourhood of a k set $t = \{t_1, \ldots, t_k\}$ is obtained by replacing a vertex t_j from t by a vertex from $V(G) \setminus \{t_j\}$: $\mathcal{N}(t) = \{t_1, \ldots, t_{j-1}, r, t_{j+1}, \ldots, t_k\}$ such that $r \in V(G) \setminus \{t_j\}$. In fact, the k-WL algorithm is the generalisation of the 1-WL algorithm on k-sets, with the previously defined notion of the neighbourhood. Therefore, it takes more global structural properties into account instead of local properties. A kernel based on the k-WL algorithm has been recently proposed [106].

In this subsection, we discussed how to define a chemical similarity based on the chemical graph.

However, predicting biological of pharmacological properties of molecules based on their chemical structures may however not be the optimal representation.

Yamanishi et al. [107] reported that pharmacological similarity between drugs is more correlated with the drug target profiles similarities, than with chemical structure similarity. Thus, the pharmacological similarity information should be a more useful source for drug–target identification than the chemical structure. A promising approach is, therefore, to use pharmacological information to compute a similarity measure within the drug-like chemical space. Examples of such similarity measures are reviewed in the next section.

2.2.3 Other chemical similarity measures

Side effect-based similarity measures

Firstly proposed by Campillos et al. [26], the use of side-effect similarity for drug-virtual screening is based on the assumption that drugs with similar side-effects are likely to interact with similar sets of target proteins. Indeed, drugs with similar in-vitro protein binding profiles tend to cause similar side effects [26], implying a direct correlation between target binding and side-effect similarity, and hence, the possibility to predict off-targets.

The use of side-effects as descriptors of compounds has given state-of-the-art performances when applied to DTI prediction [26, 108]. Using a correlation analysis approach, Yamanishi et al. [108] observed that side effect based descriptors of drug-like molecules were more correlated with their mechanism of action than with their chemical structure. They built side-effect-based feature vectors for molecules such that each element of the vector encodes for the presence or absence of each of the considered side-effect terms.

Side-effect terms can be obtained from dedicated databases, such as the SIDER or the JAPIC databases. The adverse event reporting system (AERS) is another promising resource to obtain pharmacological information. It routinely collects adverse drug event reports from patients, clinicians and pharmaceutical companies for all approved drugs and biological products. The advantages of using the AERS-based pharmacological similarity stem from the availability of this source of information for a much larger number of drugs, compared with side-effects databases. In particular, this post-marketing safety surveillance program is intended to identify unexpected drug targets [109].

Nevertheless, such descriptions require detailed side-effect information, so that these descriptors are applicable only to marketed drugs for which this information is available. To tackle this issue, several methods have been proposed to predict side effects from chemical structures [110]. However, this is limited

by the previous observation that chemical structure and pharmacological effects are only partially related. This observation explains why there still are drug-target interactions that cannot be explained nor predicted by algorithms based on side-effect descriptors.

Drug induced phenotype-based similarity measures

Drug-induced phenotypes are also expected to be useful data for structure-independent prediction of target proteins for drug candidate compounds. Recent advances in transcriptomic technologies (e.g., DNA-chips and RNA-seq) have enabled to measure the expression profiles of all human genes at low cost, and several databases containing gene expression data have been constructed worldwide [111]. The measure of gene expression profile on several cultured human cell lines after drug exposure is one way to assess drug-induced phenotypes.

Some studies [107, 112] have shown the interest of using such data for DTI predictions. However, deriving DTI from drug-induced phenotype alone can be difficult, due to the complexity, variability and sparsity of data currently available [112]. Therefore, such pharmacological and network-based similarity measures are mostly used to improve the performance of computational methods based on physical descriptors.

Phenotypic characterisation of molecules can also be established based on the Anatomic Therapeutic Chemical (ATC) classification system. Cheng et al. [113] proposed a probabilistic model that builds an ATC semantic hierarchy-based similarity measure in order to infer interactions between molecule and proteins.

Drug-target interaction network-based similarity measures

Finally, Laarhoven et al. [114] showed that DTI network-based similarity measures could perform as well as structure-(resp. sequence)-based similarity measures for chemicals (resp. proteins) on DTI prediction. They defined interaction profiles of drugs (resp. target). These profiles are binary vectors whose elements correspond to the lists of considered proteins (resp. drugs). An element equals 1 if the corresponding protein (resp. drug) is a target for the drug (resp. protein), and 0 otherwise.

Based on the assumption that two drugs interacting with similar known targets are expected to also interact with similar new targets, they defined a kernel on interaction profiles, called the Gaussian Interaction Profile (GIP) kernel. This kernel is a radial basis function (RBF) kernel applied to the profile feature vectors. They showed that the topology of drug-target interaction networks, as a source of information, is capable of predicting interacting pairs with high accuracy.

However, Laarhoven et al. obtained the best results for DTI prediction by using a combination of structure-based (resp. sequence-based) and DTI network-based similarity measures for the molecules (resp. proteins) [114].

In a later work, they extended this similarity measure to compounds for which no interaction is known [115]. More precisely, they defined the interaction profile of a new compound as the weighted sum of the profiles

of the drug compounds in the training data. The weights are the chemical similarity between the new drug and drugs in the training set.

Before exploring in chapters 3 and 4 how various representations of molecules and proteins serve to perform drug-target interaction prediction, in the next section, we discuss another prerequisite for machine learning methods: publicly available databases and gold standard datasets. These are of great importance, because they constitute the main material that can be used to train and test models.

2.3 Data and toolkits for drug virtual screening

In this section, we introduce the databases, gold standard datasets and packages most useful for DTI prediction.

All of them are considered in this thesis.

2.3.1 Publicly available databases

Molecular and bio-activity databases

Several bio-activity databases are freely available online. Some provide quantitative measurements such as the IC50, EC50, Ki and potency (in particular the PubChem [9], ChEMBL [10] and BindingDB [116] databases), while others report binary information on activities or interactions between molecules and protein pairs, like the DrugBank [117] or KEGG [118] databases. In databases recording binary information, interactions are usually not tagged as being direct or indirect.

In particular, the largest bio-activity database is the PubChem database [9] which includes around 230 million bio-activity data for around 320 million compounds and substances. It is mainly composed of three databases, namely the PubChem Substance, PubChem Compound, and PubChem BioAssay.

The PubChem Substance displays the information (such as features, activities, cross-links to other databases) about any chemical substances provided by individual contributors such that several substances entries, and IDs may refer to the same compound.

Thus, the PubChem Compound database combines all records representing the same substance into a unique compound entry (with a unique ID) using an automated process.

Bioactivities are available in the PubChem BioAssay database by bioassays unique entries and are mainly retrieved from HTS experiments. However, the PubChem data are not curated.

The ChEMBL database [10] is the second largest bioactivity database, containing around 15 million experimentally derived bioactivities, and is manually curated by experts. Binding affinity data, selectivity, efficacy and ADMET properties of molecules are recorded in the ChEMBL database. Moreover, ChEMBL associates a confidence score ranging from 0-9 to recorded activity data reflecting the specificity of both target and assay type.

Interestingly, ChEMBL distinguishes interactions into three different types based on the curated assay results: “Single Protein”, “Protein Family” and “Protein Complex”. Indeed, compounds do not always act on individual targets and may target a protein complex.

The DrugBank database [117] is another widely considered bio-activity database. Although much smaller than PubChem and ChEMBL, it provides high-quality information regarding the approved and experimental drugs along with their targets. It contains around 17,000 curated drug-target associations with high-quality standards.

More precisely, DrugBank covers almost all aspects of drugs as a manually curated bio-medical resource with high-quality standards. In addition, it contains drugs enzymes and transporters, as well as information about their identification, pharmacology, interactions with foods and other drugs, clinical trials, pharmacoeconomics, taxonomy and databases cross-links. Targets sequences are provided, but most importantly, information about protein targets are given via the link to the UniProt database [12].

The data obtained from DrugBank is often used for novel large-scale VS methods.

The Kyoto Encyclopedia of Genes and Genomes (KEGG) database [118] is another well-maintained bio-activity database. Moreover, it also comprises functional hierarchies of biological events as well as disease information.

The KEGG database is organised into several sub-databases. More importantly, the KEGG DRUG database provides information for approved drugs, their chemical structure and their targets. The KEGG PATHWAY database is a knowledge base about biological systems, by providing an extensive network and functional hierarchies of biological events, as well as the components (molecules and proteins) involved in the pathways at all levels.

The REACTOME database [119] is another widely used curated and peer-reviewed pathway database providing intuitive tools for the visualisation, interpretation and analysis of biological pathway knowledge.

Other bio-activity databases are often mentioned in the literature: the BindingDB [116], the STITCH [120], the Chemical Entities of Biological Interest (ChEBI [121]) and the ZINC [122] databases.

BindingDB contains over 1 million experimentally measured binding affinities of protein-ligand interactions. It is built form scientific papers, patents and other biological and chemical databases.

The STITCH database has the particularity to report predicted interactions with confidence scores (obtained from text-mining of the literature). STITCH also reports data from experimentally validated interactions from ChEMBL, PDB, curated datasets from other databases, and results from HTS experiments. It contains over 1.6 billion interactions.

The ChEBI database comprises around 55,000 entries and incorporates an ontological classification to describe properties of chemical structures and to detail relationships between them.

The ZINC database comprises around 100 million entries of commercially-available compounds and has the particularity to report the 3D representations (modelled with ChemAxon’s JChem package and Omega application of OpenEye Scientific Software) that can be used in docking-based virtual screening studies.

Protein databases

The Universal Protein Resource (UniProt) database [12] is the main resource for protein data. As a central hub organising a vast amount of information from several protein resources, UniProt is a comprehensive protein knowledge base organised into four primary data sources: UniProtKB, UniRef, UniParc, and Proteomes.

Most importantly, UniProtKB is the central resource of UniProt. It includes over 500,000 non-redundant, manually curated and annotated by experts (based on the experimental information extracted from the literature) protein sequences from the original Swiss-Prot database. It also comprises a hundred million computationally unreviewed and redundant protein sequences computationally derived from high throughput sequencing data of DNA, and automatically annotated from the original TrEMBL database.

UniParc provides a complete set of known sequences, in a non-redundant manner, retrieved from publicly available sequence sources (so around 200 million sequences in total). UniRef comprises clustered sets of sequences from UniProtKB and UniParc records. Finally, the "Proteomes" database comprises the sets of expressed proteins for any species whose genome was completely sequenced. In total, around 10,000 proteomes of well-studied model organisms (and around 150,000 other proteomes) are recorded.

The Protein DataBank [123] (PDB) is another primary resource for genomic data. It results from a worldwide consortium and comprises curated and annotated three-dimensional structure of proteins, nucleic acids, and complex assemblies (currently for a total of around 140,000 structures). It has been mainly obtained by X-Ray crystallography, but also includes NMR spectroscopy data and a small amount of electron microscopy data.

The sc-PDB [124] database is derived from the PDB database and comprises atom descriptions of the proteins, their ligand and more importantly the atom descriptions of ligand binding sites. It contains around 16,000 binding sites for around 5,000 proteins and 6,000 ligands.

The Pfam [125] database and the "Structural Classification of Proteins extended" (SCOPe) [126] are both providing hierarchical classifications of proteins. The Pfam database comprises a large collection of curated protein families based on sequence alignment. The SCOPe classifies proteins based on a hierarchy of 3D-structures. The top levels ("fold" and "class') describe geometrical relationships based on specific arrangements of secondary structures and chain topologies, while lower levels (including "family" and "superfamily") are determined regarding the evolutionary relationships of protein domains.

Finally, the Interpro [127] and CATH [128] resources also provide a structural and functional grouping of proteins.

2.3.2 Gold standard datasets

Gold standard, or *benchmarking* datasets are available from the literature and used to adjust parameters, evaluate the performances and, most of all, compare the performances of computational methods.

One of the first gold-standard dataset for virtual screening was built from the Kegg Drug database by Yamanashi et al. [129]. It is composed of four classes of drug-target interactions in humans involving enzymes, ion channels, G-protein-coupled receptors (GPCRs) and nuclear receptors.

In particular, the GPCRs form a large protein family with around 800 members. They are involved in various signal transduction pathways, and are well known drug targets. Ion channels constitute the second largest group of drug targets. Enzymes are catalytic proteins accelerating biochemical reactions. There are many drugs targeting enzymes (for example, aspirin targets the COX1 and COX2 enzymes). Finally, nuclear receptors form a sub-group of transcription factors which regulate the expression of genes by binding to promoter regions in DNA.

The number of known drugs targeting enzymes, ion channels, GPCRs and nuclear receptors are 445, 210, 223 and 54 respectively. The numbers of targets in these data sets are respectively 664, 204, 95 and 26. This corresponds to a number of known interactions of 2926, 1476, 635 and 90 respectively, at the time of the original article.

This dataset has been used as benchmarking dataset for a decade [129, 130, 131, 114, 132, 133, 134, 18, 135, 136], but the number of interactions it contains is notably low comparatively the known interactions recorded in the DrugBank and most of all in the ChEMBL database.

Simultaneously, other benchmarking datasets have been considered, like the DUD-E and Merck Challenge datasets, even though they have been much less used than the Yamanishi dataset.

The DUD-E dataset [137] contains 22 886 ligands and their affinities against 102 targets retrieved from the ChEMBL database while curating challenging decoys, i.e. interactions with low experimental probabilistic evidence.

The Merck Kaggle Challenge dataset provides 164 024 compounds for 15 biologically relevant targets.

More recently, Wu et al. [138] proposed MoleculeNet, a collection of curated chemical benchmark data sets for a total of around 700 000 compounds retrieved from publicly available databases. The list of data sets and their quantitative description can be found in Table. 2.1. We introduce some of them qualitatively in the following (see the paper for a full description; see the README files of each dataset provide links to obtain further details):

- QM7/QM7b, QM8, QM9 (regression): provide quantum mechanic properties of molecules such as geometric, energetic, electronic and thermodynamic properties as well as electronic spectra and excited state energy (determined using density functional theory -DFT- and multiple quantum mechanic methods).
- ESOL, FreeSolv, Lipophilicity (regression): report respectively water solubility data, hydration free energy in water and octanol/water distribution coefficient for common organic small molecules.
- PDBbind (regression): reports the binding affinities for 3D bio-molecular complexes (the 3D representation of drug-protein interaction is reported). This dataset is directly built from the PDBbind database.

- PCBA (classification): reports measured biological activities of small molecules generated by high-throughput screening from the PubChem BioAssay database. It is The "big data" dataset in MoleculeNet collection. It contains 128 bioassays binary results for a total of more than 437,000 compounds. Interested readers can search for the assay IDs on the PubChem website.

- maximum unbiased validation -MUV- (classification): also built from PubChem BioAssay. It provides randomly distributed sets of active compounds optimised to minimise the influence of dataset bias on validation results (based on a refined nearest neighbour analysis). We introduced this bias in Section 1.3.

- HIV (classification): includes binary bioactivity measurements of 41 913 compounds against the inhibition of HIV replication. Initially, the results were placed into three categories: confirmed inactive (CI), confirmed active (CA) and confirmed moderately active (CM). Authors combined the last two labels, making a binary classification task between inactive (CI) and active (CA and CM). See AIDS Antiviral Screen Data website for more details.

- Tox21 (classification): generated by the Tox21 Data Challenge community in 2014 to evaluate the performances of different computational methods for toxicity prediction. This dataset contains qualitative toxicity measurements for around 8000 compounds on 12 different targets, including nuclear receptors and stress response pathways. Refer to the website for more information about the tasks.

- ToxCast: toxicology data including around 600 in vitro high-throughput screening experiments for more than 8,000 compounds from the same initiative as Tox21. Refer to the website for more information about the bioassays.

- SIDER: the SIDER database links 1427 marketed drugs with adverse drug reactions (ADR), grouped into 27 system organ classes. Refer to the website for details on ADRs.

Category	Dataset	Data Type	Task Type	# Tasks	# Compounds
Quantum Mechanics	QM7	SMILES, 3D coordinates	Regression	1	7160
	QM7b	3D coordinates	Regression	14	7210
	QM8	SMILES, 3D coordinates	Regression	12	21786
	QM9	SMILES, 3D coordinates	Regression	12	133885
Physical Chemistry	ESOL	SMILES	Regression	1	1128
	FreeSolv	SMILES	Regression	1	642
	Lipophilicity	SMILES	Regression	1	4200
Biophysicics	PCBA	SMILES	Classification	128	437929
	MUV	SMILES	Classification	17	93087
	HIV	SMILES	Classification	1	41127
	PDBBind	SMILES, 3D coordinates	Regression	1	11908
	BACE	SMILES	Classification	1	1513
Physiology	BBBP	SMILES	Classification	1	2039
	Tox21	SMILES	Classification	12	7831
	ToxCast	SMILES	Classification	617	8575
	SIDER	SMILES	Classification	27	1427
	ClinTox	SMILES	Classification	2	1478

Table 2.1. MoleculeNet dataset collection description.

Due to the large amount of data recorded in this dataset collection, MoleculeNet is the benchmark dataset of choice for chemoinformatics. Note that it has also been integrated into the DeepChem open-source library, which implements a variety of deep learning approaches and functionalities for chemoinformatics.

2.3.3 Freely available libraries and toolkits

Chemoinformatics requires to numerically handle physical objects, namely proteins and molecules. In particular, we need to inter-translate molecules and proteins between different representations, construct descriptors, compute pairwise similarities, and apply the various statistical model. To this end, available python libraries and toolkits are very useful.

In this subsection, we focus on the libraries we used for this work, and we also introduce the main libraries and toolkits considered in the literature.

For numerically manipulating chemical molecules, we mainly used the RDkit library [139] in Python. Generally speaking, RDkit is an open source chemoinformatics toolkit available for Python, Java, C++ and C#. RDKit can handle and inter-convert a wide range of molecular representations, mol2 and SMILES representations. Most of all, RDkit can generate a wide range of atomic and molecular descriptors: for instance, a variety of structural fingerprints, topological and physicochemical descriptors (like for instance, solubility, hydrophobicity, partial charges). Interestingly, it can also generate 3D conformers from a 2D

molecular representation.

We also used the ChempCPP [140] toolkit which provides practical command line functions to compute similarities between molecules. In particular, the kernel functions that can be computed include the marginalised graph kernel and its extension between labelled graphs [88, 91], the tree subgraph based kernel [93] and pharmacophore kernel [94].

Furthermore, the OpenBabel, Dragon, Daylight, CDK (Chemistry Development Kit) and OpenEye toolkits are widely encountered in the literature. Only OpenBabel, OpenEye and CDK are open sources. They all mostly provide the same functionalities as RDKit, such as the generation of several types of molecular attributes, the conversion between various chemical formats, the generation of 3D conformers and substructure searching. Moreover, note they can be used in Python and C++.

To compute protein similarities based on sequence local alignment, we considered the LAkernel software [141] (see section 2.2). We also considered BLAST (Basic Local Alignment Search Tool) which is a prevalent sequence alignment tool that finds local similarities between protein sequences.

The main toolkit used for deriving protein attributes from protein sequence which is widely mentioned in the literature is the PROFEAT (Protein Feature Server) web server. However, when we intended to use it, we noticed that the PROFEAT package and web server seem not fully working. We considered instead the iFeature [142] and ProtR [143] web servers and packages.

Such toolkits provide feature analysis algorithms, but more interestingly, feature extraction algorithms. They can extract protein descriptors from protein sequences based on commonly used structural and physicochemical features of amino acids. In particular, they can generate 11 protein level feature groups including amino acid composition (AAC), dipeptide composition (DC), autocorrelation descriptors, "composition, transition and distribution" descriptors (CTD), quasi-sequence-order descriptors (QSO), pseudo-amino acid composition descriptors (Pse-AAC), amphiphilic pseudo-amino acid composition descriptors (Am-Pse-AAC), topological descriptors at atomic level and total amino acid properties. In general, these descriptors encode the fraction of each amino acid type and amino acid pair type within a protein as well as the distribution of amino acid properties along the sequence. They also encode the composition, transition and distribution of attributes (including hydrophobicity, normalised van der Waals volume, polarity, and polarisability) along the sequence. In addition, the ProtR package enables to derive descriptors for proteochemometric modelling, which includes more than twenty classes of 2D and 3D molecular descriptors, and BLOSUM matrix-derived descriptors.

See the PROFEAT manual and ProtR documentation for more details.

For machine learning methods, we mainly used the scikit-learn library [144] which is a well-maintained and viral free software machine learning library for Python. We broadly used machine learning shallow algorithms and auxiliary functions (for instance for computing performance scores) implemented in the scikit-learn library.

Also, we used the Tensorflow library [145] and the Keras library [146] (itself implementing on top of Tensorflow) to implement deep learning algorithms. Tensorflow is a cross-platform open source software

library for numerical computation using data-flow graphs. Its functioning is efficient for distributed training.

Finally, let us introduce the PyDTI python package. This package implements state-of-the-art methods [114, 134] for DTI prediction and provide a command line interface to compare methods to each other with the gold standard Yamanishi datasets. As presented in chapter 4, we incremented on this package by adding a benchmark dataset based on the Drugbank database and the approach we developed [50].

In the next chapter, we investigate machine learning approaches for drug-target interaction prediction with expertise-based extracted representations.

Part II

Drug virtual screening from expert-based representations for molecules and proteins

Chapter 3

Drug virtual screening with expert-based representations

Abstract: *Drug virtual screening has focused many research efforts from many points of view, indicating its important role in drug discovery. It benefited mostly from methodological advances in machine-learning algorithms, and from considerable efforts to design relevant and efficient numerical molecular and protein descriptors, as well as similarity measures. Indeed, most of the methods that predict molecular bio-activities rely on the assumption that similar molecular structures share similar biological properties. In this chapter, we recall the main virtual screening frameworks, and review state-of-the-art feature-based and similarity-based approaches for chemogenomics. This critical appraisal is to guide the development of SVM-based chemogenomics approaches in the next chapter.*

Résumé: *Le criblage virtuel de médicaments a été abordé à partir de nombreux points de vue et a concentré de nombreux efforts, ce qui témoigne de son importance. Ce domaine a principalement bénéficié des avancées méthodologiques en matière d'algorithmes d'apprentissage statistique, mais surtout des efforts considérables déployés pour concevoir des descripteurs numériques chimiques et protéiques pertinents et efficaces, ainsi que des mesures de similarité. En effet, la plupart des méthodes permettant de prédire les bio-activités moléculaires reposent sur l'hypothèse que des structures moléculaires similaires partagent des propriétés biologiques similaires. Dans ce chapitre, nous rappelons les principaux cadres qui structurent les approches de criblage virtuel, ainsi que les meilleures approches en chimiogénomique basées sur des descripteurs numériques ou basées sur des mesures de similarité. Cette revue critique guidera le développement d'approches de chimiogénomique dans le chapitre suivant.*

3.1 Drug virtual screening frameworks

One of the pioneer drug virtual screening approaches is the Similarity Ensemble Approach methodology [147] (SEA). Ligands (resp. targets) bio-activities are directly predicted based on their similarity to other ligands (resp. targets) with the desired bio-activity.

Interestingly, most of current approaches are still based on the same intuition. Very recent methods [66, 148] (with the term "new approach" in their title) still rank drug targets of a query compound according to the sum of similarities with the target's ligands. Such a naive approach reaches high predictive performances by exploiting, in a consensus-like approach, a large collection of high quality experimental bio-activity data (notably in the ChEMBL database). However, this may be due, at least in part, to the inductive bias mentioned in Section 1.3.

More sophisticated ML-based approaches have been successfully applied to drug virtual screening.

Currently, machine learning methods for virtual screening can be categorised into two global frameworks: ligand-based and chemogenomics. We will not consider docking, a receptor-based approach that uses the 3D structure of the protein, because it does not rely on machine learning approaches. It is in fact related to vast domain of molecular modelling approaches, that also include molecular mechanics and dynamics.

3.1.1 Ligand-based and chemogenomic frameworks

First, *ligand-based approaches*, such as Quantitative Structure Activity Relationship (QSAR) (refer to [149] for a recent review on QSAR), build a model function that predicts the affinity of any molecule for a given target, based on the affinities of known ligands for this target. They are efficient to study the affinity of molecules against a given protein target, but they are not suitable to study the specificity of a molecule against a large panel of proteins. This would indeed require, for each of the considered proteins, that the binding affinities of multiple ligands were available.

Second, *chemogenomic approaches* [18] can be viewed as an attempt to fill a large binary interaction matrix where rows are molecules and columns are proteins, partially filled with the known protein-ligand interactions available in public databases. In the chemogenomic framework, drug specificity prediction is formulated as a classification problem, where the goal is to distinguish protein-ligand pairs that bind from those that do not. In other words, the aim is to predict “interacting” or “not interacting” labels for all pairs, but not to predict the strength of the interaction, which would correspond to a regression problem.

Furthermore, while ligand-based (resp. target-based) virtual screening predicts targets (resp. ligands) for individual molecules (resp. proteins), the chemogenomics approach adopts a cross-target view, and attempts to simultaneously screen the chemical space against whole families of proteins. In other words, chemogenomics mines the entire space of small molecules for interactions with the biological space, i.e. the set of all proteins or at least of protein families. The lack of known ligands for a given target can then be leveraged by known ligands for similar targets.

From a broader perspective, ligand-based and chemogenomic frameworks refer respectively to singletask and multitask frameworks that we introduce in the next section.

3.1.2 Singletask and multitask frameworks

In the context of DTI prediction, a singletask method consists in predicting protein targets for a given molecule m. In this setting, the specificity of m is studied by learning a model function $f_m(p)$ that predicts whether molecule m interacts with protein p, based on known protein targets for m. This means that a new model function is learned for each molecule. We refer to this setting as *ligand-based ST*. Conversely, a singletask method could learn a model function $f_p(m)$ that predicts whether protein p interacts with molecule m, based on all ligands known for protein p. We call this the *target-based ST* setting.

In contrast, multitask methods predict whether p and m interact by training a model based on all known interactions, including those involving neither m nor p. The task of predicting ligands for protein p is then solved not only based on the data available for this task (i.e. known ligands for this protein), but also based on the data available for the other tasks (i.e. known ligands of any other proteins). The main issue is to define how the data available in all tasks can be used to make the predictions for a given task.

Therefore, chemogenomics can be viewed as a multitask prediction problem, where the tasks correspond to the predictions of ligands for individual proteins. Thus, we can measure similarity between tasks based on protein similarity measures.

Such approaches are of particular interest when few interactions are known for a given ligand or protein, as is often the case when looking for secondary targets at the size of the human proteome. However, even within the multitask framework, the prediction on orphan proteins (for which no ligand is known) and proteins for which the only known ligands are very dissimilar to the molecule in the databases, remains difficult. Therefore, our studies focused more particularly on these critical cases.

3.2 Feature-based and similarity-based approaches for chemogenomics

Chemogenomics can be categorised into three main approaches depending on the description of the input data: feature-based, similarity-based and representation learning-based approaches. In this section, we first introduce the feature-based and similarity-based frameworks and then discuss related state-of-the-art approaches for DTI prediction. Representation learning based approaches are the topic of Chapter 6 and are therefore not introduced here.

3.2.1 Definitions of feature-based and similarity-based approaches

In feature-based approaches, instances (i.e. compounds, or targets, or compound-target pairs) are represented by a numerical "data-blinded" features vector (as opposed to "data-driven" features that are end-to-end calculated from molecular graphs and protein sequences). Data-blinded features are either handcrafted

or extracted automatically based on human expertise. These descriptors reflect non exhaustively various types of physicochemical, structural, topological, geometrical. properties of the corresponding instance. The feature vectors of labelled (active or inactive) instances are then fed into a machine learning algorithm to build a predictive model for the drug-target interactions.

As previously discussed in Section 2.1, molecules are often described with structural fingerprint vectors that encode the presence and absence of structural properties. Proteins are usually described by their physical and chemical properties, and by sub-sequence distributions or functional attributes.

An advantage of feature-based methods is that they can lead to interpretable results by associating the prediction to some of the input features.

Similarity-based methods consider as input data the similarity between instances. We introduce these models in appendix A.5. They rely on the assumption that similar compounds (resp. protein), regarding structure, topology or physical properties etc., have similar functions and bio-activities and, therefore, have similar targets (resp. ligands). This assumption can however be wrong in the context of virtual screening, because a tiny change in the molecular structure or protein sequence can sometimes lead to complete loss of interaction.

As discussed in Section 2.2, molecular similarities are usually calculated by searching for common molecular substructure and isomorphism of their molecular graphs, while protein similarities are mainly calculated by sequence alignment.

One advantage of similarity-based methods (also called kernel methods), is to project input instances into a new feature space with an arbitrarily high number of dimensions, according to the Mercer theorem (see appendix A.5), which increases the predictive power of classical linear algorithms.

This also allows integrating heterogeneous data, by combining, in the same model, different similarity matrices associated with different types of data. Since it is a valuable direction of research, we discuss multi-kernel learning for DTI prediction in appendix B.

3.2.2 State-of-the-art in feature-based and similarity-based approaches for chemogenomics

Various chemogenomics methods have been proposed in the last decade [129, 135, 131, 130, 107, 109, 150, 151, 114, 115, 132, 152, 153, 133, 136]. They mostly rely on the assumption that "similar" proteins are expected to bind "similar" ligands. They differ by (i) the descriptors used to encode proteins and ligands or the similarities between these objects, (ii) the ML algorithm that is used to learn the model and make the predictions.

Pairwise kernel approach

Jacob and Vert [18] first used the powerful kernel framework to provide efficient ways to combine protein and ligand representations into a joint chemical and biological space. The most simple way to represent a (protein,molecule) pair is to concatenate both descriptor vectors into a single vector. Then, it is easy to

show that the kernel, defined on the joint space obtained by concatenating the vectors, is the sum of both kernels [18].

However, a better integration of the protein and the molecule descriptors can be introduced by considering, instead of the simple concatenation of vectors, the tensor product between the protein and ligand vectors. The descriptors of the resulting vectors are all possible products between protein descriptors and molecule descriptors. Hence, this representation allows encoding the joint presence of protein and ligand features, which encodes richer information about the interaction mechanism. It can be shown with classical algebra, that the kernel associated with a tensor product of the chemical and the protein spaces is the Kronecker product of individual kernels, which is usually called the tensor product kernel [18].

Any machine learning algorithm performing prediction in the joint space allows targets with few known ligands to benefit from the ligands known for similar proteins, which at the extreme, allows to make predictions for orphan targets (proteins with no known ligand). The information-sharing process depends strongly on ligands and protein similarity measure. Such approaches have been successfully applied to DTI prediction [135, 95].

Although SVM-based and kernel RLS-based methods yield to accurate predictors, they are, however, black boxes. All the data modelling power is used to learn the decision boundary, although the posterior probability distribution of the data is not directly learned. In addition, the SVM decision function based on support vectors ignores representative examples that are far from the decision boundary. Consequently, the output of an SVM, especially for data points that are far from the decision boundary, is not always accurate. This may be problematic in drug virtual screening, where ranking active compounds at the top of the prediction list can be more important than learning a single decision boundary between two classes.

Note however that a heuristic has been developed to compute probabilities from SVM prediction [154].

Bipartite Local Model approach

Another useful approach, known as the *bipartite local model* (BLM), has been developed by Beackley and Yamanishi [131]. In a first step, for each drug-target pair, they compute two independent singletask predictions, one from the ligand-based point of view, and a second from the target-based point of view. In a second step, they combine the two predictions to compute a final prediction for the considered pair. The originality of this method lies in the formalisation of bipartite graph inference as a set of independent local supervised learning problems.

This bipartite network approach was extended by Mei et al. [132] to orphan molecules and proteins. Their method is called BLM-NII in the original paper. Indeed, since orphan compounds have no known interactions, all the elements of their interaction profile vector, which is the binary vector specifying the presence or absence of interaction with every targets, equals 0. The authors addressed this issue using the weighted profile methods to derive an interaction profile that is used as label information to train the local model. More precisely, they defined the interaction profile of a new drug compound as the weighted sum of the profiles of the drug compounds in the training data. The weights correspond to the chemical similarity

between the new drug and drugs in the training set.

Within the bipartite local approaches, Laarhoven et al. [114] defined a kernel on interaction profiles, called the Gaussian Interaction Profile (GIP) kernel. This kernel is a radial basis function (RBF) kernel applied to the interaction profile feature vectors of drugs (resp. targets). They reported that the best results for DTI prediction on the gold-standard Yamanishi dataset are obtained by using a combination of structure-based (resp. sequence-based) and interaction profile-based similarity measures for the molecules (resp. proteins).

In a later work [115], they extended this similarity measure to compounds for which no interaction is know via the procedure of Mei et al. [132] to build interaction profiles of compounds and proteins with no known interactions, and named this method WNN-GIP [115].

The bipartite network approach addresses the central issue of the Kronecker product approach, that is to say, the computational burden triggered by the very high dimension of the Kronecker product of the two spaces. However, the information-sharing process of the bipartite network approach may be less effective. Bleackley and Yamanishi have however shown that both approaches lead to similar performances on the Yamanishi dataset [131].

Matrix factorisation

Matrix factorisation (MF) approaches decompose the interaction matrix that lives in the (protein, molecule) space into the product of matrices of lower rank, living in the two latent spaces of proteins and molecules.

The main idea of Matrix Factorisation is to project the data into a low-dimensional representation space, named 'pharmacological space', that is used to make predictions. The low-dimensional representations of drug compounds and target proteins in the pharmacological space are used to calculate the interaction scores. Yamanishi et al. [129] first proposed this kind of approach by projecting drugs and targets into a pharmacological space based on the eigenvalue decomposition of the graph-based similarity matrix.

The most recent and efficient MF-based approach, named Neighbour Regularised Logistic Matrix Factorisation [136] (*NRLMF*), consider more specifically Logistic Matrix Factorisation [155]. A specific feature of the *NRLMF* method is that it integrates a neighbourhood regularisation method, which allows taking into account only the nearest neighbours to predict a given (protein, ligand) interaction. They display excellent performances and are also computationally efficient. Note that the *NRLMF* approach is also generalised to orphan molecules and proteins, by computing latent representations of orphan molecules and proteins as a weighted sum of the latent representation of their neighbours.

As being one of the most recent papers considering similarity-based predictor for DTI prediction while providing a comparison with state-of-the-art methods implemented into a python library, we display in Table 3.1 the results reported in the original paper [136]. We observe that the proposed method *NRLMF* outperforms some of the other state-of-the-art methods on the Yamanishi benchmark dataset. Therefore, this method is used in the next chapters as "state-of-the-art", to compare the proposed approaches for drug

virtual screening.

However, this gold-standard dataset is a relatively small dataset (see Section 2.3) and does not allow to truly compare the proposed approaches, as the performances are extremely good, almost reaching the upper limit. Thus, we build a larger gold-standard dataset that is more suitable to compare methods in more realistic situations, as presented in chapter 4.

ROCAUC				
Dataset	BLM-NII	WNN-GIP	KBMF2K	*NRLMF*
Nuclear Receptor	0.905 ± 0.023	0.901 ± 0.017	0.877 ± 0.023	0.950 ± 0.011
GPCR	0.950 ± 0.006	0.944 ± 0.005	0.926 ± 0.006	0.969 ± 0.004
Ion Channel	0.981 ± 0.002	0.959 ± 0.003	0.961 ± 0.003	0.989 ± 0.001
Enzyme	0.978 ± 0.002	0.964 ± 0.003	0.905 ± 0.003	0.987 ± 0.001
Avg.	0.942	0.917	0.974	
AUPR				
Dataset	BLM-NII	WNN-GIP	KBMF2K	*NRLMF*
Nuclear Receptor	0.659 ± 0.039	0.589 ± 0.034	0.534 ± 0.050	0.728 ± 0.041
GPCR	0.524 ± 0.024	0.520 ± 0.021	0.578 ± 0.018	0.749 ± 0.015
Ion Channel	0.821 ± 0.012	0.717 ± 0.020	0.771 ± 0.009	0.906 ± 0.008
Enzyme	0.752 ± 0.011	0.706 ± 0.017	0.654 ± 0.008	0.892 ± 0.006
Avg.	0.689	0.633	0.634	0.819

Table 3.1. The AUC and AUPR obtained on the gold-standard Yamanishi dataset (results from [136]). BLM-NII [132] is the extension of the bipartite local model introduced in the previous paragraph. WNN-GIP [115] is a very similar methods that makes use of the Gaussian interaction kernel introduced in the previous paragraph. KBMF2K [133] is a probabilistic matrix factorisation approach (we introduce in the next paragraph).

Probabilistic Matrix Factorisation

The Bayesian equivalent of matrix factorisation, namely "Probabilistic Matrix Factorisation" (PMF), has also been successfully applied to DTI prediction.

In particular, Gonen et al. [133] projected drug compounds and target proteins into a unified subspace, using the kernels calculated from chemical and genomic data, respectively. The contribution of Gonen. et al. is that matrices are defined as random variables, which leads to a probabilistic inference. Later work by Gonen and Kaski extends the method to multiple side information sources expressed as different kernels [134].

Another study from Bahar et al. [156] applies PMF directly to the interaction matrix describing the bipartite interaction network. Such drug-target pair inferences, independent of 2D/3D structure comparison, implies that the latent variables defined by the factorisation capture some patterns of drugs that are relevant for the DTI prediction problem. Following Laarhoven et al. [114], this work shows that DTI network-based similarity measures can perform as well as structure-base (resp. sequence-based) similarity measures for chemicals (resp. proteins) on DTI prediction.

Random Forest

Some studies [157, 158] considered the Random Forest (RF) algorithm [159] (see appendix A.4).

RF was found slightly less efficient than SVM for DTI prediction [157]. However, it is difficult to draw a final conclusion, because this work does not use molecule and protein descriptors that could be considered entirely relevant by the community.

Shallow artificial networks

Shallow artificial networks are artificial neuron networks with a maximum of one hidden layer. One of the most interesting works in shallow neuron networks for DTI prediction is the study of Swamidass et al. [160]. They developed the Influence Relevance Voter (IRV) method, a low-parameter neuron network that refines a k-nearest neighbour classifier, by non-linearly combining the influences of chemical's neighbours in the training set.

IRV uses a pre-processing step during which all the neighbours of a test molecule are identified, and a processing step during which information from each neighbour is fed into a neuron network to produce a prediction.

The IRV method outperforms basic SVM regarding ranking metrics. In addition, its output predictions have a probabilistic semantic.

More recently, another shallow neuron network type, named Restricted Boltzmann machines [161] (RBM), have been successfully applied to DTI detection. An RBM is a two-layer graphical model that can be used to learn a probability distribution over input data [161].

When applied to DTI prediction, Wang and Zeng [42] built a specific RBM for each target, with the same number of hidden units and the same definitions of visible units. This means that the RBMs for all targets use the same parameters between hidden and visible layers. Moreover, they segregated direct and indirect interactions by adding edge properties to their network, and by using data of the Matador database[43]. The proposed method was compared only with a simple logic-based approach, and it performed better.

Ensemble methods and meta-classifiers

Recently, He et al. [162] presented a gradient boosting machine learning method for DTI prediction which outperformed the KronRLS approach on the Davis dataset (68 drugs for 442 targets) and the Kiba dataset (2116 drugs for 229 targets). Although they said to use various sources of features, they only considered similarity based descriptors for both molecules and targets. Such boosting algorithm is an ensemble model, which trains a ensemble of "weak learners" (trainable in parallel to reduce the training time) to achieve a good accuracy.

From a general perspective, the machine learning Kaggle Challenges winners are mainly ensemble of predictors. For many prediction problems, ensemble approaches were reported to outperform any of their

individual predictors, in particular for DTI prediction [151]. Ensembles of predictors make their final prediction by combining the prediction of several predictors. Combining predictions refers to averaging in the case of a regression problem, and "majority voting" or "highest probability selection" in the case of a classification problem.

A general solution for combining several predictors is to consider a meta-predictor performing the final prediction, based on the prediction scores of individual predictors. For instance, Zhang et al. [163] considered an SVM as meta-classifier for DTI prediction, i.e. a SVM classifier making its final prediction by processing as input the prediction scores of bases classifiers working on the input data features (weighted profile, GIP, LapRLS et Network Inference in this study).

Sparse Canonical Correspondence Analysis

Yamanishi et al. [41] investigated the problem of prediction interpretability by considering the Sparse Canonical Correspondence Analysis (SCCA) for DTI prediction.

It is the sparse extension of CCA, obtained by adding a constraint on the learned weights. CCA considers weighted linear combinations of molecule and protein descriptors and seeks the weights maximising the correlation between molecule and proteins descriptors and the output (interacting/ non interacting). This method did not outperform SVM with the Kronecker product of the genomic and chemical spaces, but it can identify the underlying associations between drug chemical substructures and protein functional sites which are involved in the predicted drug-target interactions, and therefore, provides interpretable results.

Multi-kernel learning

To conclude, we mention the "Multi-Kernel Learning" (MKL) framework. Although promising, it has not been widely exploited in the context of virtual screening [164, 165, 166, 167].

The interested reader can find a brief survey of multi kernel learning for DTI prediction in appendix B. Note however that we do not use such approaches later in this thesis. Nevertheless, we refer to multi-kernel learning all along the remaining of the manuscript, since we think it it is a valuable direction of research to enhance drug virtual screening.

Conclusions on previous contributions in DTI prediction

Drug specificity prediction enables to address the main bottleneck of the drug design process which is the toxicity of promising compounds. Indeed, adverse drug reactions, also called side effects, range from mild to fatal clinical events and significantly affect the quality of care. Among other causes, side effects occur when drugs bind to proteins other than their intended target. Moreover, experimentally testing drug specificity against the entire proteome is out of reach so that there is a need for chemogenomics approaches, i.e. proteome-wide drug-target interaction prediction. Chemogenomics is also a very valuable tool for drug

repositioning, since a side-effect in a pathology can be a therapeutic effect in another, as will be discussed in chapter 5.

To this end, various chemogenomics methods have been proposed in the last decade [129, 135, 131, 130, 107, 109, 150, 151, 114, 115, 132, 152, 153, 133, 136].

However, most of previous studies in the field of predicting interactions between proteins and molecules were restricted to proteins belonging to the same family, such as kinases or GPCRs [168, 129, 95, 114, 115, 132, 152, 153, 133, 136]. A few studies were devoted to larger scales in the protein space, such as [151] which however did not consider settings relevant to the prediction of drug specificity, that is to say, druggable proteome-wide prediction.

In this perspectives, we formalise and solve, in the next chapter, the problem of proteome-wide drug-target interaction prediction with Support Vector Machines (SVM), a successful algorithm for learning a classification or a regression rule from labelled examples [169].

Moreover, we build several benchmark datasets for chemogenomics, and propose *NN-MT*, a multitask Support Vector Machine (SVM) algorithm that is trained on a limited number of data points, in order to solve the computational issues or proteome-wide SVM for chemogenomics.

Chapter 4

An efficient kernel-based approach for proteome-wide drug virtual screening

Abstract: *Since we are interested in drug specificity prediction, we formulate the study of drug-target interaction prediction at the proteome scale. We build several benchmark datasets, and propose NN-MT, a multi-task Support Vector Machine algorithm that is trained on a limited number of data points, in order to solve the computational issues of proteome-wide SVM for chemogenomics. We compare NN-MT to different state-of-the-art methods, and show that its prediction performances are similar or better, at an efficient calculation cost. Compared to its competitors, the proposed method is particularly efficient to predict (protein, ligand) interactions in the difficult double-orphan case, i.e. when no interactions are previously known for the protein nor for the ligand. The NN-MT algorithm appears to be a good default method providing state-of-the-art or better performances, in a wide range of prediction scenario that are considered in this chapter: proteome-wide prediction, protein family prediction, test (protein, ligand) pairs dissimilar to pairs in the train set, and orphan cases. The work described in this chapter has been published as joint study [50] with Chloé-Agathe Azencott and Véronique Stoven in the PlosOne Computational Biology journal.*

Résumé: *Puisque nous nous intéressons à la prédiction de la spécificité des médicaments, nous formulons l'étude de la prédiction de l'interaction médicament-cible à l'échelle du protéome. Dans ce chapitre, nous construisons plusieurs jeux de données de référence et proposons NN-MT, un algorithme multi-tache basé sur un nombre limité de points de données, afin de résoudre les problèmes de calcul de l'application de SVM à l'échelle du protéome pour la chimiogénomique. Nous montrons que NN-MT est une bonne méthode par défaut offrant des performances de pointe ou meilleures, dans un large éventail de scénarios de prédiction pris en compte dans ce chaptire : la prédiction à l'échelle du protéome, de la famille de protéines, et dans le case où les paires (protéine, ligand) de test sont dissemblables des paires dans le jeux de donnée d'entrainement. Les travaux présentés dans ce chapitre ont été publiés sous la forme d'un article [50] co-signé avec Chloé-Agathe Azencott et Véronique Stoven.*

The goal of this chapter is to investigate the application of multitask Support Vector Machines (SVM) to the prediction of drug specificity, by predicting interactions between the drug of interest and the entire druggable proteome. To this end, we evaluate our multitask SVMs based methods and state-of-the-art approaches in several key scenario that explore the impact of the similarity between the query (protein, molecule) pair and the training data on the prediction performance, a point that is rarely discussed in the literature. We also explore their applicability to orphan settings, a situation often encountered large scale studies, and where single-task methods are not applicable.

Predecessors of our approach include Support Vector Machine (SVM) [135], kernel Ridge Linear Regression (kernelRLS) [129, 114, 115, 131, 132], and matrix factorisation (MF) [133, 153, 136].

We first discuss how to generate negative training examples, and we optimise protein and molecule kernels in the spaces of drug-like molecules and druggable proteins. Our observations lead us to propose the *NN-MT* algorithm, a multitask SVM for chemogenomics that is trained on a limited number of data points: for a query (protein, molecule) pair (p^*, m^*), the training data is composed of (1) all *intra-task* (protein, ligand) pairs defined by pairs (p, m) with either $p = p^*$ or $m = m^*$; (2) a limited number of *extra-task* (protein, ligand) pairs, defined by pairs (p, m) with $p \neq p^*$ and $m \neq m^*$, chosen based on the similarity of p and m to p^* and m^*, respectively; and (3) randomly picked negative examples (about ten times more than positive training pairs).

While the applicability of multitask approaches can be limited in practice by computational times, our approach only requires training on a dataset of size similar to those used by single-task methods. We evaluate the performance on various assembled datasets in which the protein and/or the ligand are orphan.

In a second phase, we compare the *NN-MT* algorithm to state-of-the-art approaches in drug-target interaction prediction [114, 115, 132, 152, 153, 133, 136]. We used and updated the PyDTI package [136], adding an implementation of *NN-MT* together with key cross-validation schemes and a DrugBank-based dataset built in the present study. Based on all other experiments performed in the present chapter, this benchmark study concludes that *NN-MT* is a good default method providing state-of-the-art or better performances, in a wide range of prediction scenario that can be encountered in real-life studies: proteome-wide prediction, protein family prediction, test (protein, ligand) pairs dissimilar to pairs in the train set, and orphan cases.

Finally, we illustrate the suitability of *NN-MT* to study the specificity of drug-like molecules in real-life situations by applying it to suggest secondary targets for 36 recently withdrawn drugs. In 9 cases, we were able to link the first ranked predicted secondary target to the side effect responsible for the withdrawal of the drug.

All codes and all the datasets that were built are available at `https://github.com/bplaye/efficient_MultiTask_SVM_for_chemogenomics/`. The updated PyDTI package is available at `https://github.com/bplaye/PyDTI/`.

Furthermore, the numerical values of performance associated to the performances plots on this chapter can be found in Supporting Information of the original paper[50].

4.1 Materials and methods

Our contributions in the present study belong to the category of methods that we briefly review in the following.

4.1.1 Kernel methods for chemogenomics

Intuitively, SVMs seeks to find the optimal hyperplane separating two classes of data points. In addition to the choice of kernels, SVMs requires an important regularisation parameter classically called C. This parameter controls the trade-off between maximising the margin (i.e. the distance separating the hyperplane and the classes distributions) and classification errors on the training points. A deeper intuition on how SVM works and how this regularisation parameters plays its role is given in appendix A.5. Additionally, although SVMs can be solved from vector representations of the data, they can also be solved using the "kernel trick", based only on the definition of a kernel function K which gives the similarity value $K(x, x')$ between all pairs of data points x and x', without needing an explicit representation of the data. Many kernels have been proposed for molecules and for proteins, and an overview of such kernels is presented in the Material and Methods section.

In chemogenomics, our goal can be viewed as finding the optimal hyperplane that separates the pairs (m, p) of molecules and proteins that interact from those that do not interact. This classification task can be solved using an SVM with a kernel K_{pair} defined on (ligand, protein) pairs. Given N example pairs, solving the SVM in the space of (m, p) pairs using the K_{pair} kernel corresponds to finding the optimal α_i coefficients such that (cf. appendix A.5):

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K_{pair}((m_i, p_i), (m_j, p_j)) \tag{4.1a}$$

$$\text{subject to } \alpha_i \geq 0, \forall i = 1, \ldots, N \tag{4.1b}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{4.1c}$$

A general method to build a kernel on such pairs is to use the Kronecker product of the molecule and protein kernels [170]. Given a molecule kernel $K_{molecule}$ and a protein kernel $K_{protein}$, the Kronecker kernel K_{pair} is defined by:

$$K_{pair}((m, p), (m', p')) = K_{molecule}(m, m') \times K_{protein}(p, p')$$

Thus, the Kronecker kernel K_{pair} captures interactions between features of the molecule and features of the protein that govern their interactions.If $K_{molecule}$ is a $n \times n$ matrix and $K_{protein}$ is a $p \times p$ matrix, their

Kronecker product K_{pair} has size $np \times np$. In the context of chemogenomics, this can correspond to a very large size, leading to intractable computations. However, one interesting property of the Kronecker kernel is that calculating its values on a data set of (m, p) pairs does not require storing this entire matrix since it is sufficient to store $K_{molecule}(m, m')$ and $K_{protein}(p, p')$.

Therefore, solving the SVM (equation 4.1) only requires calculation of the $K_{molecule}$ and $K_{protein}$ kernels according to equation 4.1.1.

Once the α_i coefficients have been determined, the ability of a given (m, p) pair to interact is predicted based on:

$$f((m,p)) = \text{sign}\left(\sum_{i=1}^{np} \alpha_i y_i K_{molecule}(m, m_i).K_{protein}(p, p_i) + b\right)$$

This equation illustrates why the use of such a kernel can be viewed as a multitask method. Indeed, in a single-task approach where one task corresponds to the prediction of ligands for a given protein p, the ability of molecule m to bind protein p would be estimated by:

$$f(m) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K_{molecule}(m, m_i) + b\right)$$

where the m_i molecules are ligand and non-ligand molecules known for protein p.

In the multitask framework, $f((m, p))$ evaluates the ability of m to bind to protein p using m_i molecules that are all ligand or non-ligand molecules known for all proteins p_i. However, the contribution of the labels y_i of ligands for p_i proteins that are different from p to calculate $f((m, p))$ is weighted by $K_{protein}(p, p_i)$. In other words, the more similar two tasks (i.e. the corresponding proteins) are, the more known instances for one of the task will be taken into account to make predictions for the other task.

Such kernel-based multitask approaches have been successfully applied to biological problems, including the prediction of protein-ligand interactions[171, 18, 135, 112].

4.1.2 Protein kernels

We used sequence-based kernels since they are suitable for proteome-wide approaches, unlike kernels relying on the 3D structure of the proteins or on binding pocket descriptions. Numerous studies have already been devoted defining descriptors of proteins based on amino-acid sequence [172, 71, 72, 80, 73]. We considered three sequence-based kernels: the *Profile kernel* [73], the *SWkernel*, and the *LAkernel*.

The *Profile kernel* uses as protein descriptors the set of all possible subsequences of amino acids of a fixed length k, and considers their position-dependent mutation probability. This kernel is available at `http://cbio.mskcc.org/leslielab/software/string_kernels.html`.

We also used two kernels that rely on local alignment scores. The first one is directly based on the Smith-Waterman (SW) alignment score between two proteins [76] and is called the *SWkernel* in the present paper. SW scores were calculated with the EMBOSS Water tool available at `http://www.ebi.ac.uk/Tools/psa/`

`emboss_water/`. We built a kernel based on the SW score matrix by subtracting its most negative eigenvalue from all diagonal values.

We also used the Local Alignment kernel (*LAkernel*) [80] which mimics the behavior of the SW score. However, while the SW score only keeps the contribution of the best local alignment between two sequences to quantify their similarity, the LAkernel sums up the contributions of all possible local alignments, which proved to be efficient for detecting remote homology [80]. This kernel is available at `http://members.cbio.mines-paristech.fr/~jvert/software/`.

Kernel hyperparameters values. The Profile kernel has two hyperparameters: the size k of the amino acid subsequences that are searched and compared, and the threshold t used to define the probabilistic mutation neighbourhoods. We considered $k \in \{4, 5, 6, 7\}$ and $t \in \{6, 7.5, 9, 10.5\}$. The SWkernel also has two hyperparameters: the penalties for opening a gap (o) and for extending a gap (e). We considered $o \in \{1, 10, 50, 100\}$ and $e \in \{0.01, 0.1, 0.5, 1, 10\}$. The LAkernel has three hyperparameters: the penalties for opening (o) and extending (e) a gap, and the β parameter which controls the importance of the contribution of non-optimal local alignments in the final score. We considered $o \in \{1, 20, 50, 100\}, e \in \{0.01, 0.1, 1, 10\}$, and $\beta \in \{0.01, 0.5, 0.05, 0.1, 1\}$. All kernels hyperparameters were optimised by cross-validation (see Section 4.1.4). All protein kernels were centered and normalised.

In the last part of the chapter, we also consider kernels on proteins based on their family hierarchy. Indeed, the most important classes of drug targets have been organised into hierarchies established on the sequence and the function of the proteins within these families (GPCR [173], kinases [174] and ion channels [175]). As in [135], the hierarchy kernel is built based on the number of common ancestors shared by two proteins in the hierarchy. More precisely, $K_{hierarchy}(t, t') = \langle \phi(t), \phi(t') \rangle$, where $\phi(t)$ is a binary vector for which each entry corresponds to a node in the hierarchy and is set to 1 if the corresponding node is part of t's hierarchy and 0 otherwise.

4.1.3 Molecule kernels

Many descriptors have been proposed for molecules, based on physico-chemical and structural properties [87, 88, 91, 97]. To measure the similarity between molecules, we considered two state-of-the-art kernels based on molecular graphs that represent the 2D structure of the molecules, with atoms as vertices and covalent bonds as edges. Both kernels compute similarities between molecules via the comparison of linear fragments found in their molecular graphs. They are available at `http://chemcpp.sourceforge.net/`.

The first one, called the Marginalized kernel[91], calculates the similarity between two molecules based on the infinite sets of random walks over their molecular graphs.

The second kernel, called the Tanimoto kernel, uses a description of molecules by vectors whose elements count the number of fragments of a given length. The similarity between molecules is based on the Tanimoto metric [87].

Kernel hyperparameters values. The Marginalized kernel has two hyperparameters: the stopping probability (while building a path) q in {0.01, 0.05, 0.1, 0.5}, and the Morgan Index (MI) in {2, 3, 4}. For both kernels, hyperparameters were selected by cross-validation (see Section 4.1.4). The Tanimoto kernel has one hyper-parameter: the length d of the paths, which we considered in {2, 4, 6, 8, 10, 12, 14}. All molecule kernels were centered and normalised.

4.1.4 Evaluation of prediction performance

When hyperparameters had to be selected, we used a nested cross validation (*Nested-CV*) scheme [176]. It consists in a (K-1) folds cross validation (*inner-CV*) nested in a K folds cross validation (*outer-CV*). At each step of the outer-CV, the inner-CV is repeated for all considered values of the hyperparameters. The values leading to the best prediction performance are retained as optimal. We used K=5, a classical value in CV.

We also considered leave-one-out cross-validation (*LOO-CV*), for which the number of folds is the number of available points in the dataset. The *LOO-CV* scheme is particularly useful when the number of samples is small. It was used in the present paper when the size of the considered dataset was too small to perform *5-fold-CV* .

We estimated prediction performance using two scores that are classically employed to judge the quality of a classifier in case of drug-target interaction prediction. The first one is the area under the Receiver Operating Characteristic curve [177] (ROCAUC). The ROC curve plots true positive rate as a function of false positive rate, for all possible thresholds on the prediction score. Intuitively, the ROCAUC score of a classifier represents the probability that if a positive and a negative interaction are each picked at random from the dataset, the positive one will have a higher positive score than the negative one. The second one is the area under the Precision-Recall curve [178] (AUPR). It indicates how far the prediction scores of true positive interactions are from false positive interactions, on average. Although we used both the ROCAUC and AUPR scores, since negative interactions are actually unknown interactions in protein-ligand interaction datasets, the AUPR is considered a more significant quality measure of the prediction method than the ROCAUC. Indeed, it emphasises the recovery of the positive samples and penalises the presence of false positive examples among the best ranked points.

4.1.5 Datasets

Many publicly available databases such as KEGG Drug [118], DrugBank [11], or ChEMBL [10] can be used to build a learning dataset of protein-ligand interactions. We chose to build all the datasets used in the present chapter from the DrugBank database v4.3, because it contains FDA-approved drugs, or drug candidate molecules. This allowed optimise and test our models on drug-like molecules, on which they intend be applied. In addition, we assumed that the list of human proteins appearing as targets for molecules of DrugBank can represent a relevant "druggable" human proteome on which we could train models that predicting the specificity of drug-like molecules.

We built a first learning dataset called S, based on Version 4.3 of the DrugBank [11]. We selected all molecules targeting at least one human protein, and having a molecular weight between 100 and 600 g.mol^{-1}, a range in which most small molecule marketed drugs are found [23]. This leads to a dataset composed of 3 980 molecules targeting 1 821 proteins, and including 9 536 protein-ligand interactions that correspond to the positive training pairs. All other protein-ligand pairs are unlabeled because no interactions were recorded for them in the database. Most of these pairs are expected not to interact, but a small number of them are in fact missing interactions. However, we considered that all unlabeled pairs as negative examples, allowing the predictor to re-classify some of these pairs as positive examples.

We built several other datasets using exactly the same training pairs as those in S, but 5-folded in various ways. Datasets S_1, S_2, S_3, and S_4 are folded so as to correspond to random, orphan protein, orphan ligand, and double orphan prediction situations. The construction of these four datasets is detailed in Section 4.2.2, where they are used. Datasets S'_1, S'_2, S'_3, and S'_4 are also folded to mimic the same situations, but with the additional constraint that proteins and ligands were clustered based on their similarities, and each fold contains only one cluster of proteins and of ligands. The goal is to test the performance of the method in situations similar to those of S_1, S_2, S_3, and S_4, but with the added difficulty that the test set (one fold) and the train set (the 4 other folds) contain pairs that have low similarities. This setting is relevant when considering proteome-wide predictions: many of the proteins to consider may not have close neighbours among the proteins for which the most information (i.e. ligands) are known. The construction of these four datasets is detailed in Section 4.2.3, where they are used.

We also built a dataset called S_0 by keeping only molecules and proteins in S which are involved at least in two interactions, in order to compare the prediction performance of the proposed methods with those of ligand-based and target-based approaches. Indeed, these two single-task approaches require at least two data points, one used as train, and one as test. Consequently, when a *LOO-CV* scheme is used, no ligand and no protein are orphans in S_0. S_0 contains 5 908 positive interactions and was used in Sections 4.2.4 and 4.2.5. In addition, we randomly generated four sets of 5 908 negative interactions involving proteins and ligands found in the positive interactions, while ensuring that each protein and each ligand are present in the same number of positive and negative interactions. Then, we assessed performance by computing the mean and standard deviation of the AUPR scores over test sets including the positive interactions set and one of the negative interactions sets.

Finally, we built three protein family datasets by extracting from S_0 all protein-ligand interactions involving respectively only G-Protein Coupled Receptors (GPCR set), ion channels (IC set), and kinases (Kinases set). These datasets were used to evaluate performance of our method within a family of proteins, and compare it to those of single-task approaches. They were extracted from S_0 (and not from the larger dataset S) since again, these comparisons used the *LOO-CV* scheme, which requires at least two data points per protein and per molecule.

Table 4.1 gives some statistics about the datasets, including the distribution of targets per drug and the distribution of ligands per protein.

	S	S_0	GPCR	IC	Kinases
number of interactions	9 536	5 908	1 735	1 603	847
number of proteins	1 821	788	85	140	143
number of molecules	3 980	1 180	482	295	577
number of targets per drug (mean/median)	5.2/2	7.5/3	20.3/6	5.9/3	11.5/4
number of targets per drug (min – max)	$1-136$	$2-82$	$1-86$	$1-67$	$1-136$
number of ligands per protein (mean/median)	2.4/1	5.0/3	3.6/3	5.4/3	1.5/7
number of ligands per protein (min – max)	$1-70$	$2-48$	$1-31$	$1-26$	$1-18$

Table 4.1. Dataset statistics

4.2 Results and discussion

4.2.1 Kernel selection and parametrisation

The goal of this section is to choose a protein kernel and a molecule kernel that we will use throughout the remainder of this study. We assumed that kernels optimised on a large dataset of interactions between drug-like molecules and druggable human proteins such as dataset S would be good default kernels for the prediction of drug candidates specificity. Therefore, we optimised kernels on dataset S (the largest dataset built in the present study), and used the best-performing couple of kernels in the remainder of the paper.

The set of (protein, ligand) pairs in S were randomly 5-folded, and we performed a *nested 5-fold-CV* experiment in order to evaluate the six possible kernel combinations and their best hyperparameters.

Kernels	Tanimoto	Marginalized
LAkernel	AUPR $= 0.938 \pm 0.001$ Tanimoto: $d = 14$ LAkernel: $o = 20$, $e = 1$, $\beta = 1$	AUPR $= 0.930 \pm 0.001$ Marginalized: $q = 0.1$, MI=4 LAkernel: $o = 20$, $e = 1$, $\beta = 1$
SWkernel	AUPR $= 0.878 \pm 0.002$ Tanimoto: $d = 6$ SWkernel: $o = 100$, $e = 0.01$	AUPR $= 0.868 \pm 0.002$ Marginalized: $q = 0.1$, MI=2 SWkernel: $o = 50$, $e = 10$
Profile	AUPR $= \mathbf{0.941 \pm 0.001}$ Tanimoto: $d = 8$ Profile: $k = 5$, $t = 7.5$	AUPR $= 0.935 \pm 0.001$ Marginalized: $q = 0.1$, MI=2 Profile: $k = 4$, $t = 6$

Table 4.2. Best *nested 5-fold-CV* AUPR for each kernel combination, together with optimal hyperparameters.

Table 4.2 gives the best prediction performance for the six combinations of protein and molecule kernels, together with the corresponding hyperparameters. All protein kernels gave the best AUPR when coupled to the Tanimoto kernel. The Marginalized kernel obtained good performance only when coupled to the Profile kernel. Overall, the Profile kernel ($k = 5, t = 7.5$) associated to the Tanimoto kernel ($d = 8$) gave the best

performance. Therefore, in what follows, we only consider these two kernels, and call MT the MultiTask SVM that uses their Kronecker product.

We also considered one-class SVM using the same kernels [33]. However, the performance of one-class SVM were clearly lower than those of KronSVM. The AUPR scores of one-class SVM were in the range of 0.6 for all considered kernels when those of KronSVM were in the range of 0.9. Therefore, we did not further consider one-class SVM.

It is worth noting that the SW-kernel gave the worst performance, although it is used in many studies [129, 114, 132, 109]. Overall, the good performance of the six multitask methods observed on S is consistent with previously reported results [132, 151]. However, S was built from the DrugBank, which is mostly fuelled by application-specific screens of either related proteins or related small molecules. Therefore, (protein, ligand) pairs of the test sets will usually have close pairs in the train set (i.e. pairs involving the same or similar proteins and ligands), which will facilitate the prediction. The performance in real-case prediction of drug specificity is expected to be lower than that obtained on S, since at the proteome scale, some of the test (protein, ligand) pairs will be far from all pairs of the train set. This will be particularly true in the case of new drugs and therapeutic targets, as already pointed by [168].

The question of interest is now to which extent the proposed MT method is effective to make predictions on more challenging situations that are relevant in the context of drug specificity prediction. Therefore, in the following, we study the evolution of MT's performance in more realistic settings where the protein, the molecule, or both, are orphan, or where the tested (protein, ligand) pair has low similarity with the pairs belonging to the train set.

4.2.2 Performance of multitask approaches in orphan situations

The goal of this section is to evaluate the performance of MT in cases where the queried (protein, molecule) pairs contain proteins and/or molecules that are *not* in the training set, as proposed by [168]. For that purpose, all the pairs of dataset S were used and 5-folded as follows in order to create four cross-validation data sets :

- S_1: randomly and balanced in positive and negative pairs;
- S_2 (corresponding to the "orphan ligand" case): (protein, molecule) pairs in one fold only contain molecules that are absent from all other folds; prediction on each test set (each fold) is performed using train sets (the four other folds) in which no the ligands of the test set are absent.
- S_3 (corresponding to the"orphan protein" case): (protein, molecule) pairs in one fold only contain proteins that are absent from all other folds; prediction on each test set is performed using train sets in which no the proteins of the test set are absent.
- S_4 (corresponding to the"double orphan" case): (protein, molecule) pairs in one fold only contain proteins *and* molecules that are both absent from all other folds. Prediction on each test set is performed

using train sets in which no the proteins and the ligands of the test set are absent. The folds of S_4 were built by intersecting those of S_2 and S_3. Thus, S_4 contains 25 folds and not 5.

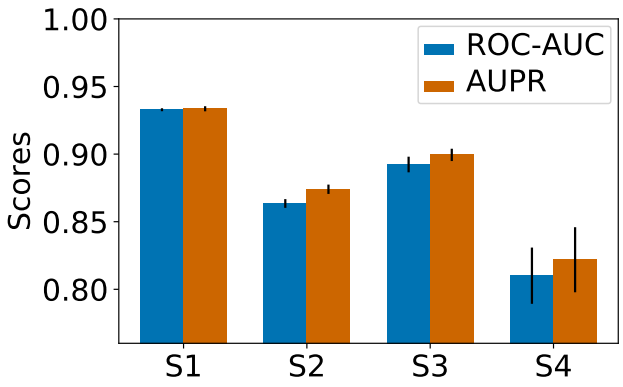


Figure 4.1. *nested 5-fold-CV* **performance of the** MT **method on the** $S_1 - S_4$ **datasets.**

Fig 4.1 shows the nested-CV ROCAUC and AUPR scores obtained by the MT method on the S_1-S_4 datasets. As expected, the best scores are obtained for S_1, and the worst for S_4, since in S_4, no pairs of the train set contain the protein or the ligand of the tested pair to guide the predictions. The loss of performance between the random and the double orphan settings is about 0.12 both in ROCAUC and AUPR. However, the performance on the S_4 dataset remains well above those of a random predictor. These results confirm that MT chemogenomics can make predictions for (protein, ligands) pairs made of unseen proteins and unseen ligands, even in datasets containing very diverse types of proteins. This confirms previous observations made on less diverse datasets [168]. It is important to point that single-task approaches would not be able to provide any prediction on the S_4 dataset.

The scores obtained in the S_2 and S_3 datasets are intermediate between those observed on S_1 and S_4. This was to be expected, as in these datasets, the algorithm can rely on training pairs containing either the same proteins (S_2) or the same ligands (S_3) as the test set. The ROCAUC and AUPR scores are both slightly better for S_3 than for S_2, which suggests that predicting ligand for new protein targets is easier than predicting targets for new compounds, as already noticed in [168]. We also observed similar behaviours when replacing the SVM with a kernel ridge regression (see Fig 4.2) and hence did not further consider this algorithm.

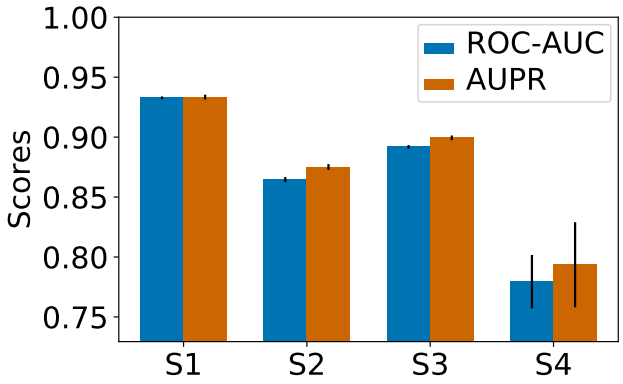


Figure 4.2. Scores of *MT* SVM chemogenomics on datasets $S_1 - S_4$, obtained with the 5-fold CV scheme.

Overall, our results suggest that the performance of *MT* is driven by known (protein, molecule) pairs that are similar to the query pair, in the sense that they share either their protein or their molecule. In the next section, we will evaluate how the actual similarity between query and train pairs influences the prediction performance of this multitask algorithm.

4.2.3 Impact of the similarity of the training examples to the test set

To evaluate the impact on performance of the dissimilarity between training and test pairs, we re-folded the pairs of S following the "clustered cross-validation" approach [179]. More precisely, we clustered proteins (resp. ligands) into 5 clusters by hierarchical clustering [180]. We then built four cross-validation datasets, $S'_1 - S'_4$, generated based on folds similarly as $S_1 - S_4$, but with the added constraint that all pairs in a given fold are made of proteins from a single protein cluster and ligands from a single ligand cluster. Therefore, test pairs are more dissimilar from train pairs than in the $S_1 - S_4$ datasets, which makes the problem more difficult.

Overall, all the pairs of dataset S were 5-folded as follows in order to create four cross-validation data sets :

- S'_1: randomly and balanced in positive and negative pairs, each fold containing proteins and ligands belonging to the same cluster;
- S_2' (corresponding to the “orphan ligand” case): (protein, molecule) pairs in one fold only contain molecules that are absent from all other folds; prediction on each test set (each fold) is performed using train sets (the four other folds) in which no the ligands of the test set are absent, with the additional constraint that each fold contains proteins and ligands belonging to the same cluster.
- S'_3 (corresponding to the “orphan protein” case): (protein, molecule) pairs in one fold only contain proteins that are absent from all other folds; prediction on each test set is performed using train sets in

which no the proteins of the test set are absent, with the additional constraint that each fold contains proteins and ligands belonging to the same cluster.

- S'_4 (corresponding to the "double orphan" case): (protein, molecule) pairs in one fold only contain proteins *and* molecules that are both absent from all other folds. Prediction on each test set is performed using train sets in which no the proteins and the ligands of the test set are absent, with the additional constraint that each fold contains proteins and ligands belonging to the same cluster. The folds of S_4 were built by intersecting those of S_2 and S_3 and S_4. Thus, S_4 contains 25 folds and not 5.

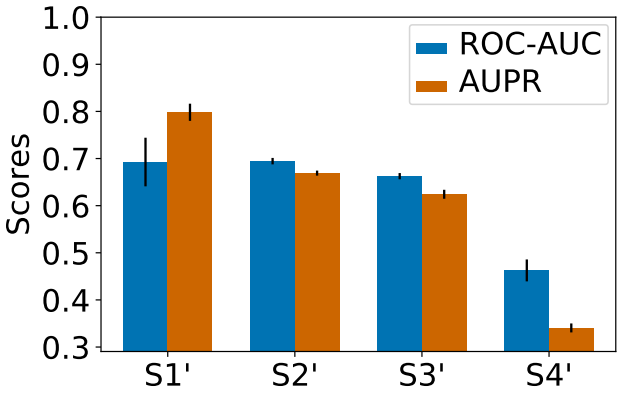


Figure 4.3. *nested 5-fold-CV* **performance of the** MT **method on the** $S'_1 - S'_4$ **datasets.**

We used the same kernels as for the MT method. Fig 4.3 shows the prediction performance of MT on these new cross-validation folds. For all the datasets, we observed a strong decrease in prediction scores with respect to those obtained on the corresponding $S_1 - S_4$ datasets.

This suggests that good performance on a query pair (p^*, m^*) is driven by the presence in the training set of pairs made *both* of proteins similar to p^* and of molecules similar to m^*, even if the query pair (p^*, m^*) is a double orphan, as in S_4. Our results also suggest that it is more important to train on pairs similar to the double orphan query pair (p^*, m^*), as in S_4, than on data containing, for example, p^* itself, but paired only with molecules quite dissimilar to m^*, as in S'_2.

In the most difficult case of S'_4, the performance is even lower than that of a random predictor, which would display an ROCAUC of 0.5. This is somewhat intriguing. One explanation could be that, when the data points are randomly dispatched in the folds used to build the train and test sets, this value is of 0.5 can be considered as a baseline.

In the case of S'4, the folds are built using clustered protein-ligand pairs, so that the data distribution in the test set (one of the clusters/folds) may be different from the data distribution in the train set (the remaining samples). This explains why a machine-learning algorithm like MT is unable to learn a model that is relevant for the test set. In other words, the baseline performance expected when systematically learning on differently distributed data might be worse than that observed by a random predictor.

These results suggest that pairs in the training set that are very dissimilar to the query pair do not help making more accurate predictions. In other words, although the kernels used in multitask approaches modulates how information available in one task is shared for training other tasks (the further the tasks are, the less information is shared), using information from distant tasks seems to degrade performance. This insight is interesting since the *MT* algorithm requires calculating the Kronecker kernel on all (protein, molecule) pairs, which is computationally demanding. Therefore, the next two sections evaluate whether removing distant pairs from the training set can improve computational efficiency without degrading performance.

4.2.4 Multitask approaches on reduced training sets

Based on the insight that *MT* prediction is driven by training examples that are close to the query data, we propose to build training tests of reduced sizes by removing training examples distant from the query. The goal of this section is also to compare the prediction performance of the *MT* method trained on these reduced data sets to that of the simpler and faster single-task method, since there would be no point in using the more complex *MT* method if a single-task method performs better. Because this study is motivated by ligand specificity prediction, we chose to focus on comparisons with the *ligand-based ST* method rather than *target-based ST*.

In what follows, n^+ (resp. n^-) will refer to the number of positive (resp. negative) examples in the train set.

In all the following experiments, we used the *LOO-CV* scheme because intra-task and extra-task pairs can only be defined for each pair separately, which prevents from using *K-fold-CV* schemes. In addition, in single-task approaches, the size of the training set was relatively small in most cases (see datasets statistics in Section 4.1), which does not allow to fold the data. We checked that the *LOO-CV* scheme did not trigger a bias, as sometimes observed [168] (results not shown).

Because prediction of a given (protein, ligand) interaction can only be made by single-task if the pair partners are present in at least another pair of the train set, in the following experiments, we used the S_0 dataset in which all ligands and all proteins are involved in at least two known interactions, as explained in Section 4.1.5.

Training on intra-task positive examples

The goal of this section is to compare the prediction performance of the *MT* method trained on a reduced data set (of size similar to that employed in single-task methods) to those of single-task methods. Since *ligand-based ST* can only use intra-task positive examples, the only positive training pairs we use for the *MT* method are the intra-task pairs as well. Note that *MT* still gets more training examples than *ligand-based ST*, since pairs formed with the query protein and a different ligand are also included. By reducing the training set size, the computational times required by the *MT* method are now similar to those of the single-task method. In the following, we call *MT-intra* this variant of *MT*. For each test ligand, we build

the negative training examples by randomly selecting a number n^- of proteins that do not interact with the ligand in S_0. We vary n^- from 1 to $100 \times n^+$.

Fig 4.4 shows the *LOO-CV* AUPR of *MT-intra* and *ligand-based ST* on S_0, for increasing values of the n^-/n^+ ratio. For both methods, the AUPR score increases with the number of negative pairs in the train set, before decreasing for large numbers of negative pairs. A good trade-off for both computational and predictive performance seems to be in the range of 10 times more negative points than positive points. We therefore set n^-/n^+ to 10 for the remaining experiments of this section.

The AUPR scores of *MT-intra* outperform those of the *ligand-based ST* method. Interestingly, the performance of the *MT-intra* with a n^-/n^+ ratio of 10 is close to 0.96 which outperforms the AUPR score of 0.93 obtained with *MT* (see Section 4.2.2). This indicates that including in the train set pairs displaying low similarity to the tested pair degrades both the computational time and the quality of the prediction of *MT*.

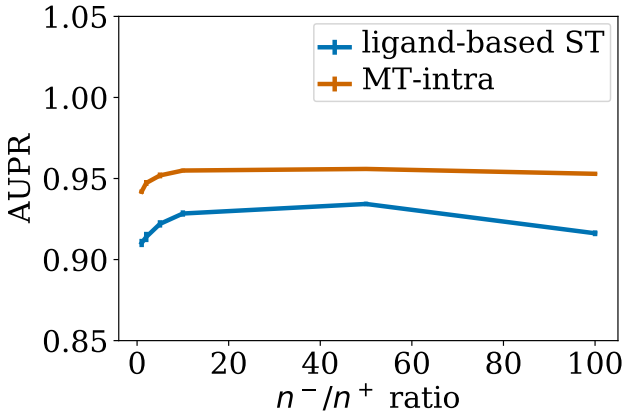


Figure 4.4. Performance of single-task and *MT-intra* as a function of the n^-/n^+ ratio.

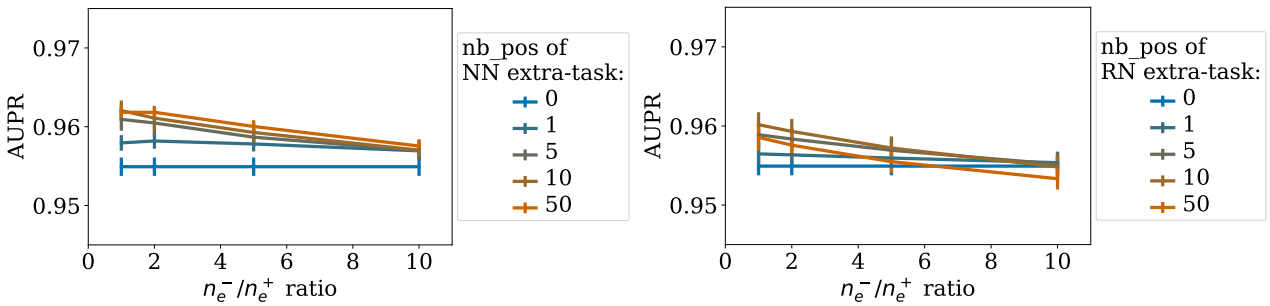
Adding similar extra-task positive examples

The results from Section 4.2.3 suggest to explore the performance of the *MT-intra* method when trained on various datasets including extra-task pairs close to the tested pair, in addition to the intra-task pairs. Therefore, we built train sets made of:

- the train set of *MT-intra*
- n_e^+ closest extra-task positive pairs with respect to the tested pair ($n_e^+ \in \{1, 5, 10, 50\}$).
- n_e^- closest extra-task negative pairs with respect to the tested pair, so that the n_e^-/n_e^+ ratio varies from 1 to 10.

We call *NN-MT* (for Nearest Neighbour *MT*) the resulting variant of *MT*. We also considered a similar approach in which the extra-task pairs were chosen at random rather than according to their similarity to the test pair. We refer to this method as *RN-MT* (for Random Neighbour *MT*).

We report the *LOO-CV* performance of *NN-MT* and *RN-MT* on Fig 4.5. Fig 4.5(a) shows that, while adding to the train set 0 to 50 nearest neighbour extra-task positive pairs with respect to the tested pair, the prediction performance of *NN-MT* slightly and monotonously increases. Fig 4.5(b) shows that the performance of *RN-MT* also slightly increases (although not monotonously) when random extra-task pairs are added. However, its best performance remains under that of *NN-MT*. Finally, using a high n_e^-/n_e^+ ratio did not improve the performance. This is an interesting observation, since limiting the size of the train set is computationally favourable.



(a) *NN-MT* (b) *RN-MT*

Figure 4.5. AUPR as a function of the n_e^-/n_e^+ ratio for increasing numbers random extra-task points in the train set. (a): *NN-MT*. (b): *RN-MT*. The blue horizontal line corresponds to *MT-intra* (which is trained only on intra-task pairs).

Taken together, these results show that *NN-MT* outperforms not only *MT-intra*, but also the more computationally demanding *MT* method trained in the *LOO-CV* setting in Section 4.2.2. AUPR scores for (protein, ligand) pairs involving non-orphan ligands and non-orphan proteins are expected to be very high (around 0.96).

However, predicting the specificity of a given ligand requires the ability to make predictions for proteins that are far from the known targets. In these cases, the high prediction scores obtained in this section might not hold. Therefore, in the next section, we study the performance of *NN-MT* when the test pairs are far from the train set.

4.2.5 Impact of the distance of the intra-task examples to the query pair

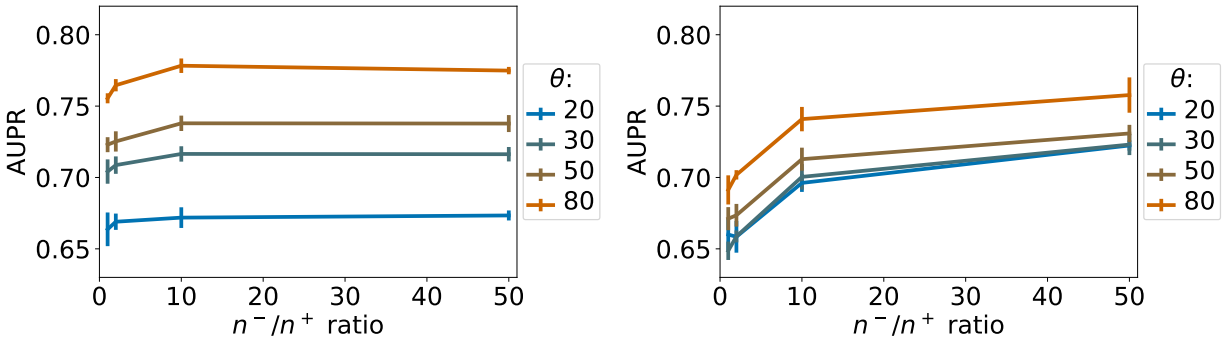
The goal of this section is to evaluate the performance of the *MT-intra* and *NN-MT* proposed methods, and to compare them to those of the *ligand-based ST* method, when the similarity between the test pair and the training data varies.

Training on dissimilar intra-task positive examples

We first evaluated the performance of *ligand-based ST* and *MT-intra* when the similarity between the test pair and the training data varies. To do so, we computed the percentiles of the molecules (respectively proteins) similarity distribution in S_0.

For each test pair (p^*, m^*), the training set only included the positive intra-task pairs (p, m) such that $K_{protein}(p, p^*)$ and $K_{molecule}(m, m^*)$ is lower than a percentile-based threshold θ. We then added n^- random intra-task negative pairs. We did not apply a similarity constraint to negative pairs, since, unlike the positive pairs, they are available in large numbers and at all distances from the tested pairs.

Fig 4.6 reports the *LOO-CV* AUPR scores of *ligand-based ST* and *MT-intra* for varying values of θ ($20^{th}, 30^{th}, 50^{th}$, and 80^{th} percentiles) and of the n^-/n^+ ratio (from 1 to 50).



(a) *MT-intra* method

(b) *ligand-based ST* method

Figure 4.6. AUPR scores as a function of the n^-/n^+ ratio, for percentile-based threshold θ ranging from 20% to 80%. (a): *MT-intra* method. (b): *ligand-based ST* method.

Fig 4.6(a) shows that, as expected, the performance of *MT-intra* increases when the similarity of the tested pair to the train set increases from the 20th to the 80th percentiles (AUPR of 0.67 to 0.77). However, the performance is still much lower than when the closest pairs are allowed in the training set (AUPR of 0.96, see Section 4.2.4). Fig 4.6(a) also suggests that a n^-/n^+ ratio of 10 is again an appropriate choice, as when all intra-task positive example are used (see Section 4.2.4). We therefore set n^-/n^+ to this value for the remaining experiments of this section.

Fig 4.6(b) shows that *ligand-based ST* behaves similarly to *MT-intra*: the AUPR score increases from 0.70 to 0.75 for *ligand-based ST* when threshold θ increases from the 20th to the 80th percentiles. These values again remain much under the AUPR score of 0.93 observed when all intra-task pairs are used. Although modest, the performance of *MT-intra* remains above those of *ligand-based ST* for all tested thresholds of similarity (except $\theta = 20$) between the tested pair and pairs of the train set.

Adding similar extra-task positive examples

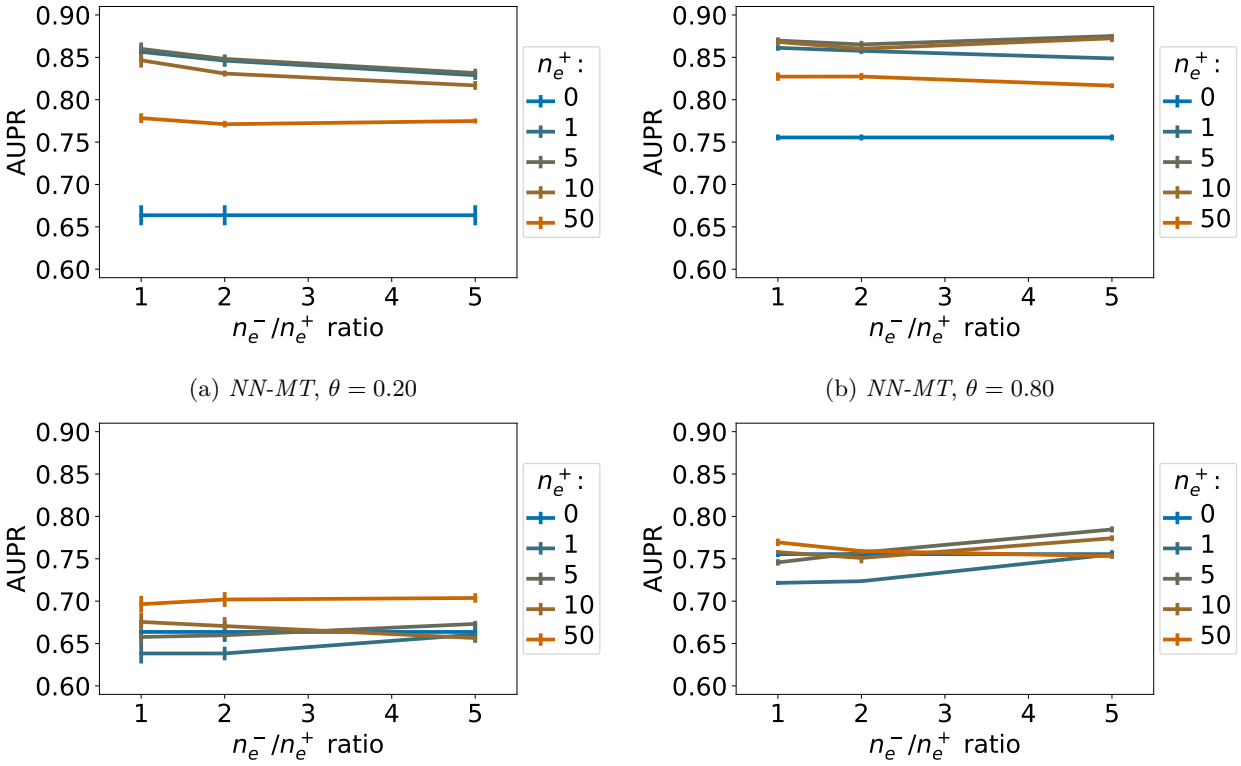
We then explored to which extent adding extra-task (protein, ligand) pairs to the training set of the *MT-intra* method improves the prediction scores.

Applying the same percentile-based similarity constraint to the intra-task positive pairs, we compared the performance of *NN-MT* and *RN-MT* when respectively adding n_e^+ nearest neighbours or random extra-task positive to the training set. We did not apply a similarity constraint to the extra-task pairs, since the principle underlying multitask methods is precisely to learn from extra-task data, which is particularly critical when the intra-task pairs of the train set are scarce or far from the tested pair, as illustrated by the poor performance of *ligand-based ST* in the previous section. A number n_e^- of nearest neighbours (respectively random) negative extra-task pairs were also added for *NN-MT* (respectively *RN-MT*).

Figs 4.7(a) and (b) report the *LOO-CV* AUPR of *NN-MT*, as a function of n_e^-/n_e^+ (ratio of negative over positive extra-task pairs) and for a number of extra-task positive pairs varying from 0 to 50, respectively for percentile similarity constraints θ of 20 and 80. The blue horizontal line (for $n_e^+ = 0$) corresponds to the performance of the *MT-intra* methods. Fig 4.7(a) and (b) show that adding extra-task pairs to the train set dramatically improves performance. The AUPR score reaches values above 0.85, independently of θ, suggesting that when no close intra-task pairs are available, performance is driven mainly by extra-task training pairs, confirming our observations in Section 4.2.3. Moreover, when the number of extra-task pairs increases, the performance of *NN-MT* increases, then tends towards that of *RN-MT*, and then degrades at larger values of n_e^+ because too many dissimilar extra-task pairs are included in the training set. This implies that only a limited number of the closest extra-task pairs is required to reach optimal performance. Adding the same number of negative extra-task pairs (n_e^-/n_e^+=1) provides the best AUPR, which again limits the size of the required training set. Unsurprisingly, the best AUPR in the absence of the closest intra-task pairs (around 0.87 for $\theta = 0.80$) is still lower than when all available intra-task pairs are used (AUPR=0.93, see Section 4.2.4). Note that, although the performance of *MT-intra* can be biased when considering similarity thresholds of 20th and 80th percentile, because the corresponding sizes of the train sets might be different, this is not the case for the *NN-MT* method because the prediction is driven by extra-task pairs.

On the contrary, Fig 4.7(c) and (d) show that the performance does not improve when the extra-task training pairs are chosen at random, and therefore, are on average further from the test pair. It might even degrade when the number of extra-task pairs becomes large. Finally, Fig 4.8 shows that, for all similarity thresholds, the performance of the *MT-intra* and *RN-MT* methods are similar, and far beyond that of the *NN-MT* method.

In conclusion, in settings where (protein, ligand) pairs similar to the test pair are available, our results suggest the best prediction performance are obtained using the *NN-MT* method trained with 10 times more negative intra-task pairs than positive ones, 1 to 5 extra-task nearest neighbour positive pairs, and the same number of extra-task negative pairs. The computational time will be reasonably comparable to that of *ligand-based ST*, and performance should be high enough (AUPR above 0.85) to guide experimental evaluations for drug specificity prediction.



(a) *NN-MT*, $\theta = 0.20$

(b) *NN-MT*, $\theta = 0.80$

(c) *RN-MT*, $\theta = 0.20$

(d) *RN-MT*, $\theta = 0.80$

Figure 4.7. AUPR score of *NN-MT* and *RN-MT* as a function of the n_e^-/n_e^+ ratio, for a number of extra-task positive pairs n_e^+ varying from 0 to 50, and for percentile-based similarity threshold θ of 20 and 80 applied to the intra-task positive pairs. (a): *NN-MT*, $\theta = 0.20$. (b): *NN-MT*, $\theta = 0.80$. (c): *RN-MT*, $\theta = 0.20$. (d): *RN-MT*, $\theta = 0.80$.

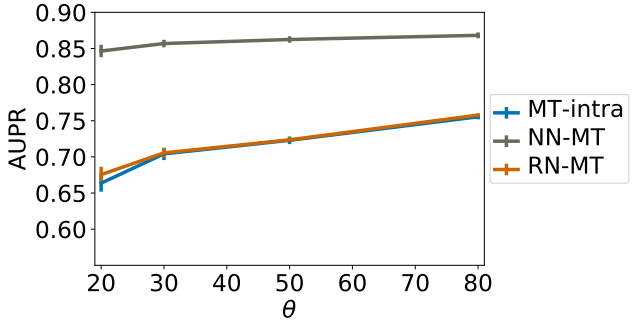


Figure 4.8. AUPR score as a function of percentile-based similarity θ, for n^-/n^+=10, a number of extra-task positive pairs n_e^+=10 and a ratio of n_e^-/n_e^+=1 for extra-task pairs.

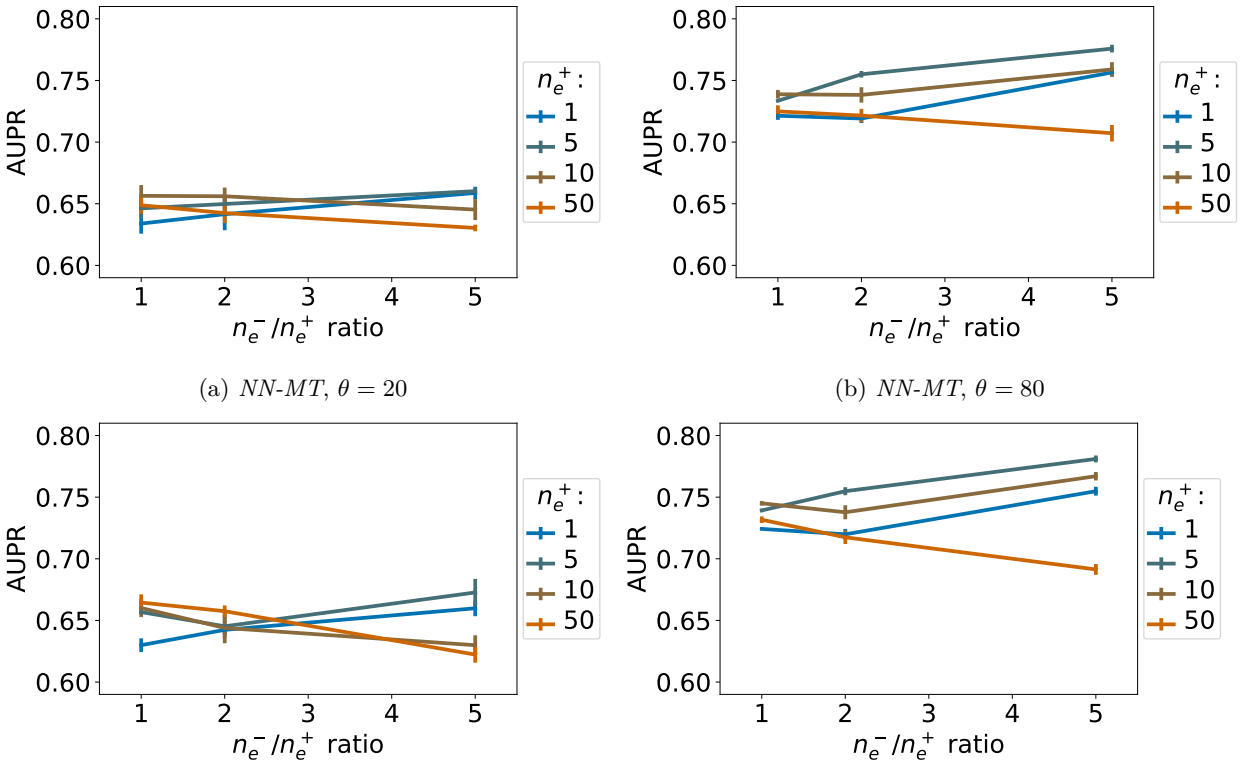
Adding dissimilar extra-task positive examples.

While we previously argued that the point of multitask approaches is to leverage similar extra-task data to improve prediction performance, ligand specificity studies can require the prediction of interactions between

proteins and ligands for which very little similar extra-task data is available. We therefore repeated the experiments from the previous section, but this time applying the percentile-based similarity constraint to both intra-task and extra-task positive pairs of the train set. We report the corresponding *LOO-CV* AUPR on Fig 4.9.

We observe that the performance of *NN-MT* remains overall low (best AUPR score of 0.75 for $\theta = 0.80$). Adding dissimilar extra-task positive pairs fails to improve the scores obtained when only intra-task positive pairs are included in the training set. Hence, if neither close intra-task nor close extra-task positive pairs are available, no method can provide performance good enough for the purpose of drug sensitivity prediction. These interactions would have to be experimentally tested if they are critical in the context of a drug's development program.

These observations were expected given that adding random extra-task training pairs, possibly far from the tested pair, did not improve performance of the *MT-intra* method (see Section 4.2.5).



(a) *NN-MT*, $\theta = 20$

(b) *NN-MT*, $\theta = 80$

(c) *RN-MT*, $\theta = 20$

(d) *RN-MT*, $\theta = 80$

Figure 4.9. AUPR scores of the *NN-MT* and *RN-MT* multi-task methods as a function of the n_e^-/n_e^+ ratio, for a number of extra-task positive pairs n_e^+ varying from 1 to 50. (a): *NN-MT*, $\theta = 0.20$. (b): *NN-MT*, $\theta = 0.80$. (c): *RN-MT*, $\theta = 0.20$. (d): *RN-MT*, $\theta = 0.80$. The two methods are trained with intra-task and extra-task examples that are both dissimilar to the tested pair (percentile-based similarity thresholds θ of 20 and 80).

Taken together, our results show that the proposed *NN-MT* method is the most appropriate for predicting the specificity of a molecule. Indeed, it outperforms all its comparison partners independently of the number of known (protein, ligand) interacting pairs involving the same or similar ligands or proteins as the query pair. In addition, it requires much fewer training pairs than the classical *MT* approach, and its computational time is therefore close to that of a single-task method. Finally, in the most challenging setting where no similar intra-task nor extra-task training data is available, it performs significantly better than random, in a context where *ligand-based ST* could not make any prediction.

The results we have presented so far address the issue of using kernel methods with SVM in the context of proteome-wide specificity prediction, at a tractable computational cost thanks to the choice of a reduced learning dataset, without loss in prediction performance. Reducing datasets to the most informative data points is also the underlying idea of active-learning methods [181]. In the latter, the goal is to guide step by step which data points are needed (i.e. have to be observed and labelled) to best improve the prediction performance. These methods are used when acquiring data is the limiting factor. By essence, their goal is also to reduce the size of the data sets that are used.

However, another key issue corresponds to study the specificity of a molecule within a family of related proteins. Indeed, when a new drug candidate is identified against a given therapeutic target, proteins belonging to the same family are important off-target candidates. This corresponds to the setting where similar training pairs are available, since proteins of the same family are similar in terms of sequence.

In the next section, we therefore assess whether the proposed *NN-MT* method, initially dedicated and tuned in proteome-wide prediction problems, also provides good performance for molecule specificity prediction within a family of proteins.

4.2.6 Specificity prediction within families of proteins

We considered three families of proteins because they gather a wide range of therapeutic targets, and have also been considered in other chemogenomics studies, thus providing reference prediction scores: G-Protein Coupled Receptors (GPCRs), ion channels (IC), and kinases. All the (protein, molecule) pairs involving GPCRs, ICs, or kinases that were present in the dataset S described in Section 4.1.5 were used to build the three corresponding family datasets.

We compared the performance of the *MT-intra* method (trained using only positive pairs involving the protein or the ligand of the tested pair) to those of the *NN-MT* and *RN-MT* methods, in order to evaluate the interest of the multitask approach in family studies. We considered two versions: one in which the Profile protein kernel is used, as in the above sections, and another in which a family-based hierarchy kernel is used (Section 4.1.2), because a sequence-based kernel may not be optimal to study the specificity of the molecule within a family of proteins [135, 95]. The corresponding methods are called *MT-intra-family*, *NN-MT-family*, and *RN-MT-family*.

As in the above section, each (protein, ligand) test pair is considered in turn in a *LOO-CV* scheme. We used a learning dataset containing: all positive intra-task positive pairs, ten times more random negative intra-task pairs (this value was found adequate in previous sections), a varying number of positive extra-task pairs (nearest neighbours for *NN-MT* or *NN-MT-family*, random for *RN-MT* or *RN-MT-family*), and a number of negative extra-task pairs so that the ratio of n_e^-/n_e^+ varies from 1 to 20.

G-Protein Coupled Receptor family

Fig 4.10 shows that all methods perform very well, with AUPR scores above 0.95. Including extra-task positive pairs in the train set improves the AUPR score, even when added randomly. This indicates that, contrary to studies in larger scales in the protein space, in family studies, extra-task pairs are always close to the tested pair because they belong to the same family. However, the performance reached when adding positive nearest-neighbour extra-task pairs remains above those reached when adding positive random extra-task pairs, as observed in the larger scale studies presented above. Overall, adding 10 to 50 extra-task positive pairs to the train set, and around 10 times more random negative extra-task pairs leads to the best performance.

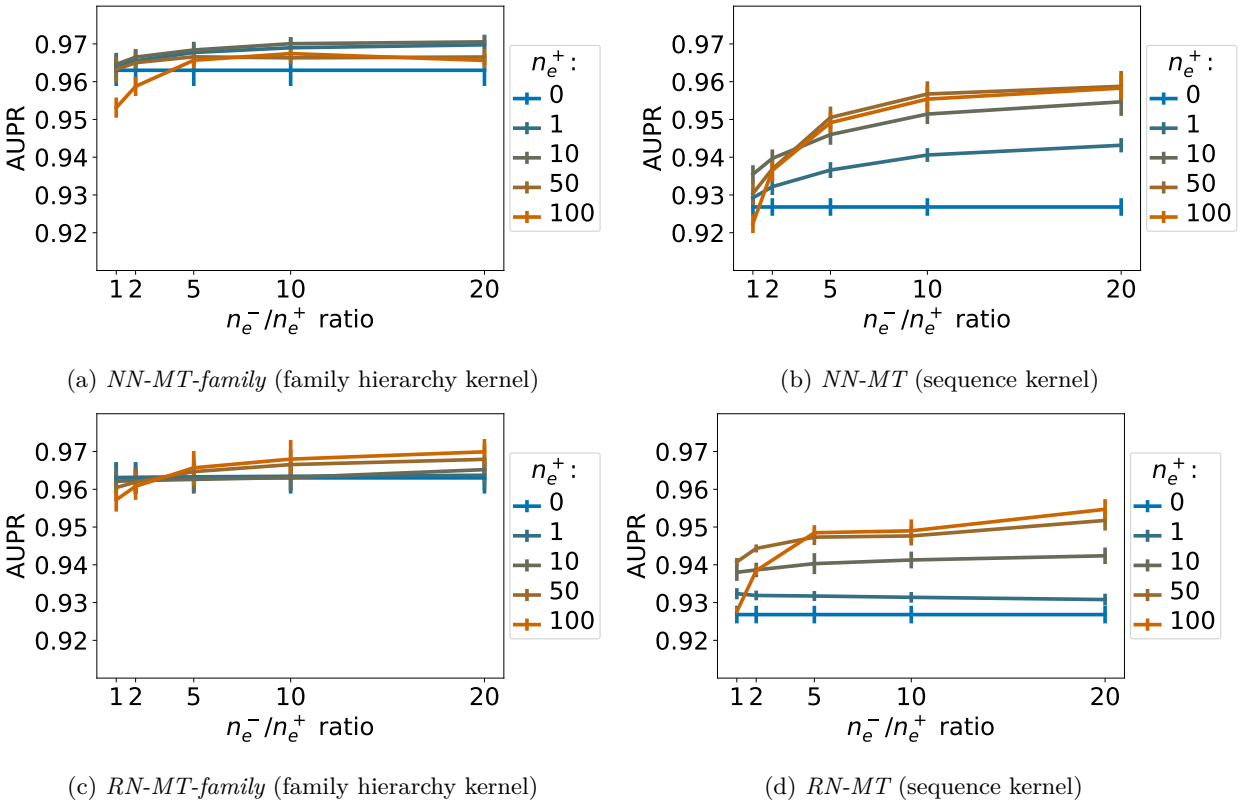
The best AUPR scores of the *NN-MT* and the *NN-MT-family* methods are close (0.96 and 0.97). Although the best scores of the *NN-MT-family* method are slightly above those of *NN-MT*, one should note that the family GPCR kernel is based on a GPCR hierarchy that was established using known GPCR ligands. Therefore, the results obtained by the *NN-MT-family* might be biased, which is not the case for those obtained by the *NN-MT*.

Ion Channel family

The conclusions obtained above in the GPCR family also hold in the IC family, as shown in Fig 4.11. Again, all methods perform very well, reaching AUPR scores above 0.97. As for the GPCR family, adding 10 to 50 extra-task positive pairs to the train set, and around 10 times more random negative extra-task pairs leads to the best performance.

Kinase family

In the kinase family, the results are somewhat different from those obtained on IC and GPCRs. The *NN-MT* and *RN-MT* methods both outperform the *MT-intra* method that is trained using only intra-task pairs, as shown in Figs 4.12(b) and (d). Again, 10 to 50 extra-task pairs, with a n_e^-/n_e^+ ratio in the range of 1 to 5 leads to the best results, with AUPR scores in the range of 0.93. Unexpectedly, the *NN-MT-family* and *RN-MT-family* methods, which both use the kinase family hierarchy kernel, tend to perform not as well when extra-task pairs are added to the training set, than when only the intra-task pairs are used, as shown in Figs 4.12(a) and (c). In addition, their best AUPR scores reaches 0.90, which is lower than those of the *NN-MT* and *RN-MT* methods which are in the range of 0.93. These observations may reflect the fact that the kinase family gathers proteins that are relatively more diverse than GPCRs and IC. For example, one can



(a) *NN-MT-family* (family hierarchy kernel)

(b) *NN-MT* (sequence kernel)

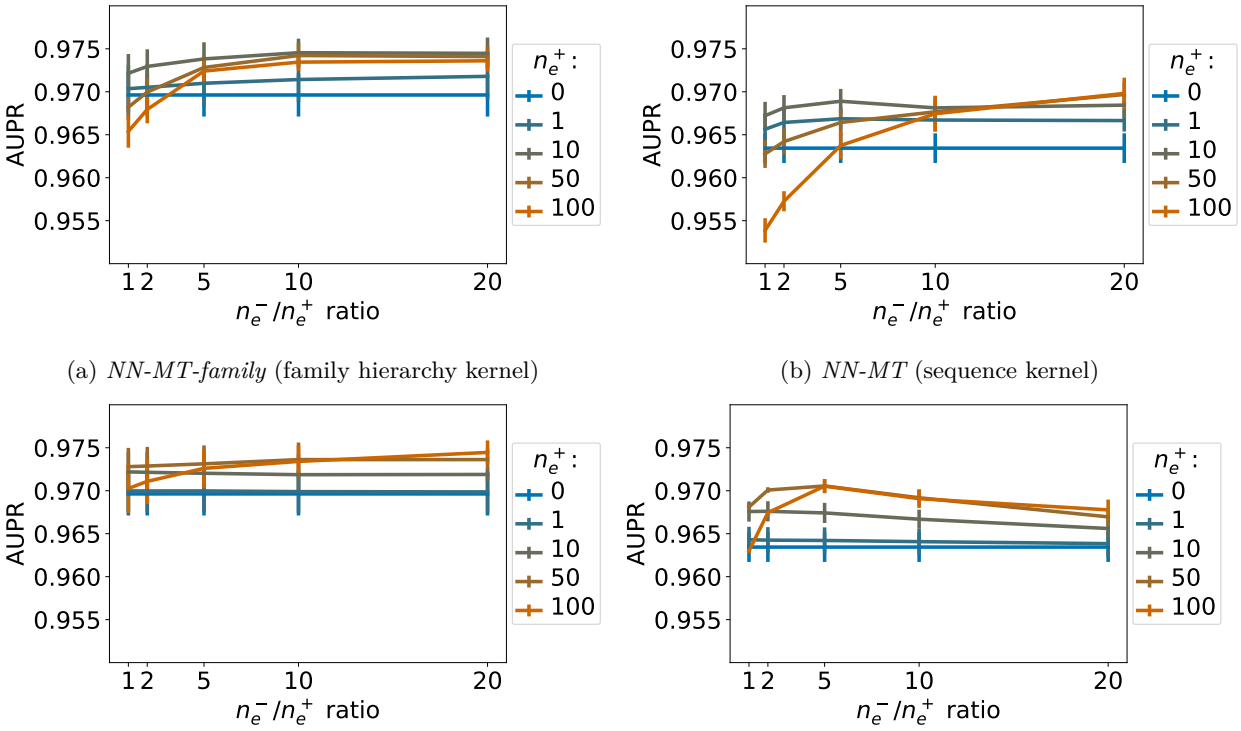
(c) *RN-MT-family* (family hierarchy kernel)

(d) *RN-MT* (sequence kernel)

Figure 4.10. AUPR score of the considered multitask methods on the GPCR family as a function of the n_e^-/n_e^+ ratio, for a varying number n_e^+ of extra-task positive pairs. (a): *NN-MT-family* (family hierarchy kernel). (b): *NN-MT* (sequence kernel). (c): *RN-MT-family* (family hierarchy kernel). (d): *RN-MT* (sequence kernel). The blue horizontal line corresponds to the *MT-intra* method trained only on intra-task pairs.

distinguish Tyrosine kinases and Serine/Threonine kinases, or globular protein kinases and receptor protein kinases. This diversity is illustrated by the organisation of the kinome in some 50 distinct sub-families [174]. In this context, the sequence kernel that was optimised in proteome-wide studies might better capture the degree of similarity between two kinases than the hierarchy kernel does.

Overall, the above results on the IC, GPCR and kinase families indicate that the proposed *NN-MT* method leads to the best results when the train set includes all positive intra-task pairs, 10 times more random negative intra-task pairs, a small number of nearest neighbours positive extra-task pairs (in the range of 10) and around 5 times more random negative extra-task pairs. These conditions are very similar to those leading to the best prediction scores when ligand specificity is studied on large scale in the protein space. Even if the performance of *NN-MT* on family datasets is better than those reached by other methods on similar datasets [132, 135], they remain in the same order of magnitude.



(a) *NN-MT-family* (family hierarchy kernel)

(b) *NN-MT* (sequence kernel)

(c) *RN-MT-family* (family hierarchy kernel)

(d) *RN-MT* (sequence kernel)

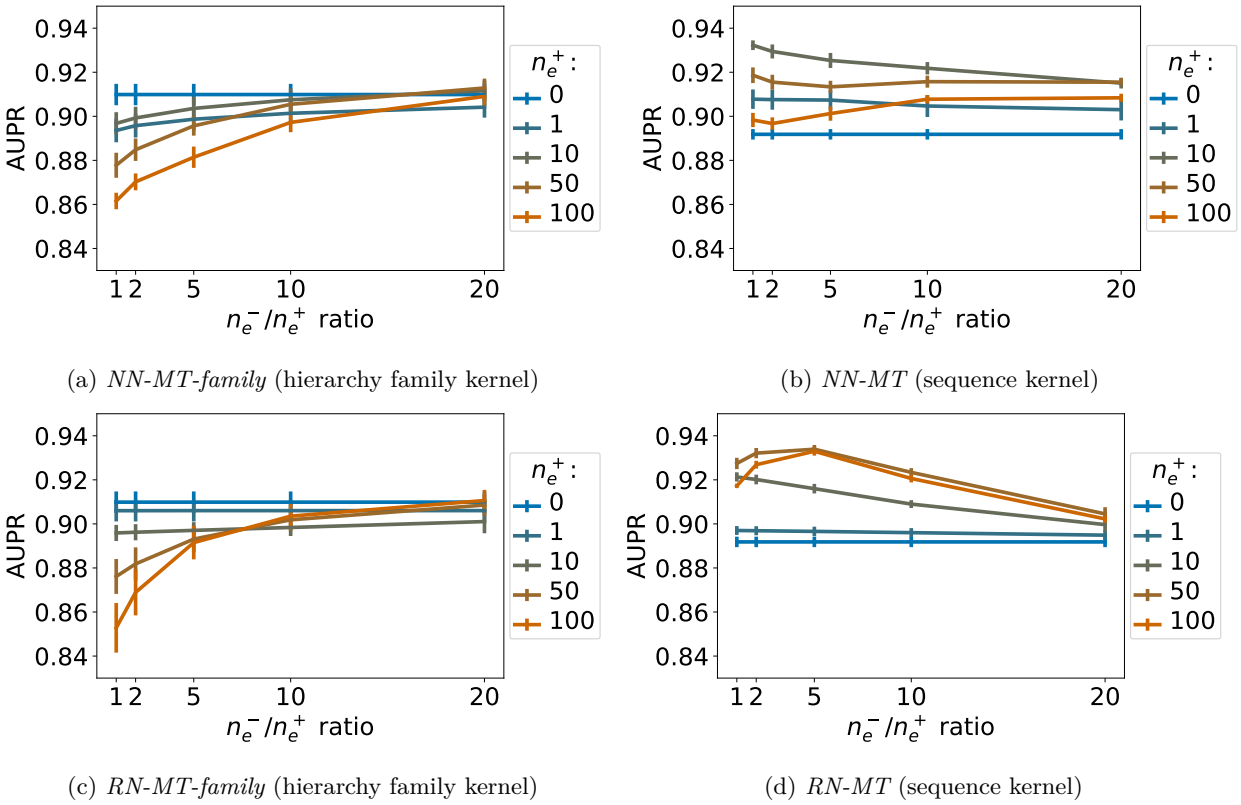
Figure 4.11. AUPR of the multitask methods on the IC family. (a): *NN-MT-family* (family hierarchy kernel). (b): *NN-MT* (sequence kernel). (c): *RN-MT-family* (family hierarchy kernel). (d): *RN-MT* (sequence kernel). The blue horizontal line corresponds to the *MT-intra* method trained only on intra-task pairs.

4.3 Illustration of the method on withdrawn drugs

One important application of the *NN-MT* method is to guide pharmaceutical companies in the choice of the best hit molecules early in the drug development process and to avoid deleterious side effects for patients.

In order to test the interest of the proposed method in this context, we considered a list of 174 drugs that were withdrawn in at least one country between 1960 and 2009 [182, 183].). Of these, 64 molecules had a DrugBank ID, among which 45 were present in our dataset S. We analysed the target predictions made by the *NN-MT* method on these 45 molecules, to determine whether they could be related to the side effects that lead to drugs withdrawal.

In three cases, the protein target responsible for the side effects were known and present in the dataset. In six cases, withdrawal was attributed to major hepatic or renal toxicity (i.e. metabolic toxicity) or severe but nonspecific side effects, which are not a priori attributable to unknown targets. We studied in more details the 36 remaining drugs (list in Table 4.3), and considered only their predicted target of highest score, according to the *NN-MT* method. In one case, the best prediction was confirmed, and in eight cases, we found a direct link between the cause of withdrawal and the best predicted target. These nine examples



(a) *NN-MT-family* (hierarchy family kernel)

(b) *NN-MT* (sequence kernel)

(c) *RN-MT-family* (hierarchy family kernel)

(d) *RN-MT* (sequence kernel)

Figure 4.12. AUPR score of the multitask methods within the kinase family. (a): *NN-MT-family* (family hierarchy kernel). (b): *NN-MT* (sequence kernel). (c): *RN-MT-family* (family hierarchy kernel). (d): *RN-MT* (sequence kernel). The blue horizontal line corresponds to the *MT-intra* method trained only on intra-task pairs.

are shortly discussed in the following. Among 174 withdrawn drugs between 1960 and 2009, we considered the following list of 36 drugs having a DrugBank ID, and for which the side-effect responsible of withdrawal could be due to unknown secondary targets.

name	year	country	DrugBank id
Alosetron	2000	US	DB00969
Adderall	2005	Canada	DB00182
Cerivastatin	2001	US	DB00439
Cloforex	1969	Germany	DB00631
Dexfenfluramine	1997	European Union, UK, US	DB01191
Propoxyphene	2010	Worldwide	DB00647
Zimelidine	1983	Worldwide	DB04832
Dofetilide	2004	Germany	DB00204
Encainide	1991	UK, US	DB01288
Etretinate	1989	France	DB00926

Fenfluramine	1997	European Union, UK, US, India, South Africa, others	DB00574
Fenoterol	1990	New Zealand	DB01288
Levomethadyl	2003	US	DB01227
Metipranolol	1990	UK, others	DB01214
Minaprine	1996	France	DB00805
Astemizole	1999	US, Malaysia, Multiple Nonspecified Markets	DB00637
Oxyphenbutazone	1984-1985	UK, US, Germany, France, Canada	DB03585
Bithionol	1967	US	DB04813
Pergolide	2007	US	DB01186
Phenacetin	1975	Canada	DB03783
Phenformin	1977	France, Germany US	DB00914
Phenolphthalein	1997	US	DB04824
Diethylstilbestrol	1970s		DB00255
Phenylpropanolamine	2000	Canada, US	DB00397
Prenylamine	1988	Canada, France, Germany, UK, US, others	DB04825
Remoxipride	1993	UK, others	DB00409
Rimonabant	2008	Worldwide	DB06155
Rofecoxib	2004	Worldwide	DB00533
Rosiglitazone	2010	Europe	DB00412
Sertindole	1998	European Union	DB06144
Sibutramine	2010	Australia, Canada, China, the European Union (EU), Hong Kong, India, Mexico, New Zealand, the Philippines, Thailand, the United Kingdom, and the United States	DB01105
Sparfloxacin	2001	US	DB01208
Tegaserod	2007	US	DB01079
Terfenadine	1997-1998	France, South Africa, Oman, others, US	DB00342
Thalidomide	1961	Germany	DB01041
Valdecoxib	2004	US	DB00580

Table 4.3: List of 36 considered withdrawn drugs

Rimonabant (DrugBank ID DB06155) was an anti-obesity drug acting as an antagonist of the cannabinoid receptor CNR1 (Uniprot ID P21554), withdrawn due to serious psychiatric effects. Its best predicted target is acetylcholinesterase (Uniprot ID P22303). The interaction between Rimonabant and acetylcholinesterase was absent from DrugBank, but this prediction is confirmed in the DrugCentral database (DrugCentral ID 4150), showing that Rimonabant inhibits acetylcholinesterase with an IC50 value of 4.6 μM. Although in this particular case, the known target CNR1 might be, at least in part, responsible for the psychiatric side effects [184], the fact that the best prediction was confirmed is an interesting result.

Two molecules were withdrawn because of teratogenicity risk: Diethylstilbestrol (DB00255) and Thalidomide (DB01041). This is consistent with the first predicted target DNA polymerase alpha catalytic domain POLA1 (Uniprot ID P09884) for the former, and DNA polymerase epsilon subunit 3 POLE3 (Uniprot ID Q9NRF9) for the latter.

Etretinate (DB00926) was withdrawn because of birth defect risks, while its best predicted target is steryl-sulfatase STS (Uniprot ID P08842), an enzyme that converts sulfated steroid precursors to estrogens during pregnancy [185].

Four molecules were withdrawn for cardiac toxicity : Dextropropoxyphene (DB00647), Rosiglitazone (DB00412), Prenylamine (DB04825), and Dexfenfluramine (DB01191). The first predicted target for the two former are involved in the metabolism of NO, a key molecule for the fine tuning of cardiac function. For Dextroproxyphene, the predicted SDF2 target (Uniprot ID Q99470) plays a critical role in the activation of NO production [186], and for Rosiglitazone, the predicted DDAH1 target (Uniprot ID O94760) has a role in NO production regulation [185]. Prenylamine was withdrawn for cardiac arrythmia while its best predicted target is the calcium-activated potassium channel KCNMB3 (Uniprot ID Q9NPA1), which plays a critical role in the the cardiac excitation-contraction coupling [187]. Dexfenfluramine was withdrawn for heart attack and stroke risks while its best predicted target is PI3K (Uniprot ID P48736), a kinase involved in many aspects of heart homeostasis [188].

Finally, Zimelidine (DB04832) is a serotonin reuptake inhibitor that was used as antidepressant, and withdrawn because of severe central and peripheral neuropathy. The best predicted target is sodium- and chloride-dependant GABA transporter 2 SLC6A14 (Uniprot ID Q9NSD5), involved in the transport of the key neurotransmitter GABA [189], which could be related to the observed side effects.

To summarise, for the 36 withdrawn molecules in S for which side effects could have been caused by unidentified secondary targets, we made the very stringent choice to consider only the top predicted target according to *NN-MT*.

The top predicted target could be confirmed for one molecule, and a direct link between the predicted target and the side effect responsible for drug withdrawal could be found for eight molecules. We do not claim that these observations constitute a validation test set of our the predictions (this was performed with the cross-validation schemes used on the various datasets studied in the above sections). However, they illustrate that in 9 cases out of 36, the function of the top predicted target alone could have helped to foresee potential side effects, and to conceive additional experiments to better evaluate the interest of these drug candidates before entering drug development phases.

For the remaining 27 molecules, no direct link was found between the best predicted target and the side effects that caused drug withdrawal. These side effects might have been indirectly related to the best predicted target, but a full analysis of the corresponding pathways for these 27 examples was out of the scope of this paper. They might also have been related to predicted targets of lower ranks, which we did not consider.

Such restrictions in the analysis of the results would of course not be employed in real applications, which would help to better foresee potential deleterious drug candidates.

4.4 Discussion: comparison to other methods

We compare now the prediction performances of the proposed *NN-MT* method to those of two state-of-the art methods: a recent Matrix Factorisation method called Neighbourhood Regularised Logistic Matrix Factorisation (*NRLMF*) [136], and the Kronecker (kernel) Regularised Least Square regression method *KronRLS* (a kernel-based method, as *NN-MT*) [114, 115].

The *KronRLS* and *NRLMF* methods were published based on their prediction performance on four protein family datasets, Nuclear Receptors (NR), GPCR, Ion channels (IC), and Enzymes (E) that contained respectively 90, 636, 1476 and 2926 interactions [129].

The *KronRLS* method uses a kernel $K_{molecule}$ for molecules that is defined by:

$$K_{molecule} = \frac{1}{2}(K_{\text{SIMCOMP}} + K_{\text{GIP},m}) \tag{4.2}$$

where K_{SIMCOMP} is a structure similarity kernel [86], and where $K_{\text{GIP},m}$ is a Gaussian kernel that compares the interaction profiles of molecules against the proteins of the dataset [114]. For proteins, the kernel $K_{protein}$ is defined by:

$$K_{protein} = \frac{1}{2}(K_{\text{sequence}} + K_{\text{GIP},p}) \tag{4.3}$$

where K_{sequence} is a protein sequence similarity kernel also based on the Smith-Waterman score [76], and $K_{\text{GIP},p}$ is a Gaussian kernel that compares the interaction profile of proteins against the molecules of the dataset.

A specific feature of the *NRLMF* method is that it integrates a neighbourhood regularised method which allows to take into account only the K nearest neighbours to predict a given (protein, ligand) interaction (in practice, the authors recommend K=5).

We performed benchmark experiments on these family datasets using the PyDTI package. This package initially contained the *KronRLS*, a variant of it called *KronRLS-WNN* [132] that generalises to orphan cases, and the *NRLMF* methods, and the kernel matrices $K_{protein}$ and $K_{molecule}$ calculated for the four family datasets. In all experiments, we compare the intrinsic performances of the algorithms: the similarity measures used are the same for all methods. More precisely, the three methods used the kernels available in PyDTI: the structure-based K_{SIMCOMP} for molecules, and a kernel K_{sequence} based on the Smith-Waterman score. In addition, *KronRLS* also used the $K_{\text{GIP},m}$ kernel, leading to the $K_{molecule}$ kernel for molecules defined in eq. 4.2, and the $K_{\text{GIP},p}$ kernel, and to the $K_{protein}$ kernel for proteins defined in eq. 4.3. The

two other methods do not use the K_{GIP} kernels because they do not take into account information about interaction profiles.

We also performed benchmark experiments on a dataset gathering more diverse protein and ligands. To this end, we used the DrugBank-based dataset S_0 described in Section 4.1.5) containing 5 908 interactions.

Because *KronRLS* and *NRLMF* could not make predictions on S_0 at a manageable computational cost in the *LOO-CV* scheme, we randomly sampled 2 000 of the 5 908 interactions of this data set to create a smaller test data set called $S_{0,2000}$. We still used all of S_0 (minus the test example) for training.

We calculated the Tanimoto and Profile kernels optimised in the present study (see Section 4.2.1), and these matrices were uploaded in PyDTI so that the three considered methods could used them. In addition, since *KronRLS* also use the $K_{\text{GIP},m}$ and the $K_{\text{GIP},p}$ kernels, we described all molecules and proteins in S_0 by their interaction profile. We calculated the $K_{\text{GIP},m}$ and $K_{\text{GIP},p}$ kernels on S_0 and uploaded these kernels in PyDTI. Only *KronRLS* used these additional kernels. All cross-validation experiments were performed building test sets from $S_{0,2000}$ and using all remaining data points in S_0 for training.

Table 4.4 presents the performance of the three considered methods on the protein family datasets.

Method / Dataset	NR	GPCR	IC	E
KronRLS	0.75 ± 0.14	0.9 ± 0.03	0.96 ± 0.01	0.96 ± 0.01
NN-MT	0.89 ± 0.09	0.95 ± 0.02	0.97 ± 0.01	0.97 ± 0.01
NRLMF	$\mathbf{0.96 \pm 0.04}$	$\mathbf{0.96 \pm 0.02}$	$\mathbf{0.98 \pm 0.01}$	$\mathbf{0.98 \pm 0.0}$

Table 4.4. AUPR scores and standard deviations in *10-fold-CV* , test sets balanced in positive and randomly chosen negative samples

Globally, the performance of all methods are high and close, with AUPR scores above 0.9 in most of the cases. On average, the *NRLMF* and *NN-MT* methods are on par and lead to the best results. These results are consistent with those reported in [136].

Method / Dataset	$S_{0,2000}$
KronRLS	0.91 ± 0.02
NN-MT	0.95 ± 0.01
NRLMF	$\mathbf{0.96 \pm 0.01}$

Table 4.5. AUPR scores and standard deviations in *10-fold-CV* , test sets balanced in positive and randomly chosen negative samples

Table 4.5 confirms the tendencies observed in Table 4.4. Although the performances are slightly lower on this more diverse $S_{0,2000}$ dataset than on the family datasets, they remain high, with *NRLMF* and *NN-MT* keeping the best AUPR scores.

As discussed in Sections 4.2.1 and 3, various prediction methods lead to such high performances because, in the protein family or $S_{0,2000}$ datasets, predictions are averaged over test pairs in which the protein and/or the molecule might be orphan, or not. These averaged results hide less favourable situations, typically double orphan samples. Because these cases are common and important when predicting specificity of a new drug candidate at the proteome scale, we would like to stress that comparing methods in orphan cases is a more stringent and relevant test. In such cases, the performance are expected to be more modest and the methods might not rank in the same order.

Therefore, we ran the three methods using a *LOO-CV* scheme on double orphan (protein, molecule) pairs on the same datasets. We considered the *LOO-CV* scheme as it is particularly convenient to test double orphan drug-target interactions. In these experiments, for each tested (p, m) pair, interactions involving the considered protein or the molecule are ignored in the train set. The *LOO-CV* schemes were balanced in positive and randomly chosen negative pairs.

In order to better explore the performance of the considered methods in this double-orphan setting, we used two versions of the two kernel-based methods, initially introduced for *KronRLS* . More precisely, in [115], the authors proposed an approach called WNN (weighted nearest neighbour) that, for each orphan molecule m (resp. protein), an interaction profile is computed by summing the weighted profiles of non orphan molecules in the dataset. The weighting depends on the similarity between the orphan molecule and all other non orphan molecules. This predicted profile is used in the training to predict labels to all (protein, m) pairs of the dataset. Thus, in the first version of *KronRLS* [114], all the labels of (protein, m) pairs involving the orphan molecule m were set to 0. Based on this WNN procedure some of these non interactions might be re-qualified as true interaction before training the predictor. In other words, the WNN algorithm can be viewed as a mean to de-orphanise molecules or proteins in order to help the predictions on such cases. In the following, we will call *KronRLS-WNN* the *KronRLS* method ran with the WNN algorithm. Using the PyDTI package, we also considered a version of *NN-MT* in which the WNN algorithm is the implemented, and call it *NN-MT-WNN* in the following.

Method / Dataset	NR	GPCR	IC	E
KronRLS-WNN	0.78 ± 0.03	0.83 ± 0.01	0.78 ± 0.01	0.84 ± 0.0
KronRLS	0.55 ± 0.01	0.62 ± 0.01	0.64 ± 0.0	0.56 ± 0.0
NRLMF	0.19 ± 0.03	0.15 ± 0.0	0.26 ± 0.01	0.23 ± 0.0
NN-MT	0.72 ± 0.04	0.76 ± 0.01	0.72 ± 0.01	0.66 ± 0.0
NN-MT-WNN	$\mathbf{0.77 \pm 0.05}$	$\mathbf{0.85 \pm 0.0}$	$\mathbf{0.79 \pm 0.01}$	$\mathbf{0.84 \pm 0.0}$

Table 4.6. AUPR scores and standard deviations on double orphan *LOO-CV* , balanced number of positive and randomly chosen negative test samples

Table 4.6 presents the results of the double-orphan benchmark on the family datasets. Surprisingly, in these double-orphan experiments, the *NRLMF* method has very modest results and does not perform as well as the other methods. We think it may be due to an undetected implementation issue.

The results of *NN-MT* remain well above the random performance of 0.5, but not the *KronRLS* method. The WNN algorithm dramatically improves the performance of *KronRLS* , and to a lesser extent those of *NN-MT* , and overall, the *NN-MT-WNN* algorithm leads to the best performance in most cases.

Method / Dataset	$S_{0,2000}$
NRLMF	None
KronRLS-WNN	75.6 ± 0.43
KronRLS	0.4979 ± 0.0071
NN-MT	0.60 ± 0.01
NN-MT-WNN	$\mathbf{0.85 \pm 0.01}$

Table 4.7. AUPR scores and standard deviations on double orphan *LOO-CV* , balanced number of positive and randomly chosen negative test samples

Table 4.7 presents the results of the double-orphan benchmark $S_{0,2000}$ dataset. We did not run the *NRLMF* method in this experiment, because it was computationally too intensive in this *LOO-CV* , and because it already gave very poor results on the easier family dataset. Moreover we shortened the train set of *KronRLS* and *KronRLS-WNN* methods by considering only the thousand molecules (resp. proteins) closest to the molecule (resp. protein) of the test sample. Thus, the computation time was reduced to some hours instead of months which made those methods computationally reasonable.

Overall, the scores are lower on this dataset than on the family datasets because $S_{0,2000}$ is a more diverse dataset on which predictions are more difficult, in general, than on the family datasets. However, the same tendency is observed: *NN-MT* performs better than *KronRLS* , and when the WNN algorithm is used, *NN-MT-WNN* performs better than *KronRLS-WNN* .

Overall, the results of these benchmarks show that the *NN-MT* method present state-of-the-art or better results on the protein family datasets and the more diverse DrugBank-based dataset. In the general case, it appears to be a good default method in terms of performance, number of parameters and computational efficiency, which are important issues for non expert users.

In the specific double-orphan case, only the two kernel-based methods *NN-MT* and *KronRLS* lead to performance well above those of a random predictor. The WNN algorithm, proposed in [115] improves the performance of *KronRLS* and of *NN-MT* , but resulting *NN-MT-WNN* method lead to the best performance.

Finally, it is interesting to compare the computational complexities of the methods as a function of the number of hyper-parameters that they contain. Indeed, these hyper-parameters need to be optimised by cross-validation, leading to heavy computational issues in the case of the large-scale datasets used in proteome-wide chemogenomics. As can be seen in the PyDTI package, *NRLMF* has 5 regularisation parameters, *KronRLS* has 2 hyper-parameters (decay parameter T and the weight parameter used to combine kernels; the regularisation parameter and the bandwidth of the GIP kernel are fixed), and *NN-MT* has 1 hyper-parameter (regularisation parameter C for SVM). In practice, the optimisation of *NRLMF* in the

LOO-CV scheme was out of reach, requiring several days of calculation while the other methods required hours, which represents a limitation of this method. This could explain in part the very low performances displayed by *NRLMF* in the double-orphan experiment. In addition, the double-orphan case was not considered in [136], and therefore, we do not have any performance reference for this case. In this setting, latent vectors describing proteins and ligands are both estimated from non-orphan neighbours, and interaction prediction may fail in this situation, and contribute to the poor performance observed. However, we did cross-validate *NRLMF* parameters in the double-orphan setting in the case of the family NR dataset (the smallest dataset used in this section). This allowed a modest increase in AUPR score from 0.14 to 0.19 (reported in Table 4.6). Therefore, even if the *NRLMF* method had been optimised on the other datasets, we do not expect that this would have changed the overall conclusion that this method is not suitable for handling orphan cases.

4.5 Conclusion

The present chapter tackles prediction of ligand specificity on large scale in the space of proteins. More precisely, our goal was to propose a method to explore the specificity of molecules with state-of-the-art or better performance over a wide range of prediction situations: at the proteome or protein family scales, on average or in specific situations such as tested pairs far from the train set, or such as orphan proteins and ligands. In other words, the aim was to propose a robust default method, applicable to many types of studies, thus avoiding development of *ad hoc* complex and specific methods to non expert users. We chose to formulated it as a problem of predicting (protein, ligand) interactions within a multitask framework based on SVM and Kronecker products of kernels on proteins and molecules. Within the kernel-based SVM methods tested in the Results section, we showed that the *NN-MT* method fulfils these requirements. In particular, *NN-MT* outperforms both the multitask *MT* method and the corresponding single-task kernel-based methods, while it also keeps a computational cost close to that of single-task approaches. The *NN-MT* algorithm fulfils these requirements, leading to the best prediction performance for the three tested settings which cover most of the prediction situations that would be encountered in real-case studies.

To summarise the main characteristics of the proposed *NN-MT* method (detailed in Sections 4.2.1, 4.2.4 and 4.2.5), we suggest to predict each (protein, ligand) interaction using the Profile kernel for proteins (with subsequences length k of 5 and threshold equalled to 7.5) and the Tanimoto kernel for molecules (with length of path 8), with a train set including:

- all positive intra-task pairs (i.e. all known interactions involving the protein or the ligand of the test pair), and around ten times more randomly chosen intra-task negative pairs.
- a small number of the closest positive extra-task pairs (i.e. a number similar to that of intra-task positive pairs), and a similar number of randomly chosen negatives extra-tasks pairs.

This should provide good default parameters to use the *NN-MT* method in a straightforward manner for users that are not familiar with machine learning approaches.

We used the DrugBank database to build several datasets that illustrate various prediction contexts and that we made available online to the community for future benchmarking studies. We also updated the PyDTI package [136] with an implementation of *NN-MT* together with several cross-validation schemes and our DrugBank-based dataset. This allowed us to compare the *NN-MT* method to recent approaches developed on drug-target interaction prediction [114, 115, 132, 152, 153, 133, 136]. In the context of wide-scale prediction of molecule specificity, the DrugBank-based dataset is more relevant that the family datasets that have been widely used. Indeed, it contains a set of proteins that can be viewed as a relevant druggable proteome to train and test computational models for drug specificity prediction.

The benchmark study comparing *NN-MT* to the matrix factorisation *NRLMF* and the kernel-based *KronRLS* algorithms on family and DrugBank-based datasets showed that, all methods displayed high performances, *NRLMF* and *NN-MT* leading to the best results. However, on the more demanding double-orphan tests performed on the same datasets, *NRLMF* performed much poorer than the kernel-based *NN-MT* and *KronRLS* algorithms. In this orphan case, the WNN algorithm makes it possible not only to significantly improve the performance of *KronRLS* , but also that of *NN-MT* , the *NN-MT-WNN* algorithm leading to the best results.

To conclude, this chapter suggests that including only similar samples to the tested sample forces the algorithm to adjust the hyperplane to segregate positive and negative sample distributions in a better way. In the future, it might be relevant to consider a weighted-sample SVM in order to integrate such behaviour in a more continuous way, by decreasing the error importance on train samples depending on their distance to the tested sample.

From another perspective, recent developments within the relatively old Artificial Neural Networks field resulted in significant improvement in some popular yet challenging fields such as computer vision [51], speech recognition & translation [190] and bioinformatics [191, 192].

In the case of drug-target interaction prediction, the first noticeable breakthrough was when a deep learning algorithm won the Kaggle Merk Virtual Screening Challenge. This drew considerable attention to employing deep learning techniques for virtual screening. Therefore, we explore this avenue in the next Part of this manuscript.

In between, we discuss in the next chapter another aspect of drug virtual screening as crucial as developing state-of-the-art approaches for chemogenomics. Indeed, we illustrate, in a couple of studies, how drug-target interaction prediction can be interfaced with therapeutic research as a guide for deciphering underlying biological mechanisms of candidate drugs.

Chapter 5

Two applications of drug virtual screening to assist pharmaceutical research

Abstract: *As introduced in Section 1.2, drug virtual screening is a valuable tool for guiding drug research throughout the drug design process. In particular, drug-target interaction prediction is a useful tool for deciphering the underlying biological mechanisms of a drug, which can help to identify deleterious side-effects as well as other characteristics of a drug. We were involved in three drug repositioning studies dedicated to triple negative breast cancer or cystic fibrosis diseases. In this chapter, we introduce two of these projects and summarise our contribution. For each of the two projects, we highlight, with strong evidences, side therapeutic paths unrelated to those used to treat the patients currently.*

Résumé: *Comme présenté dans la section 1.2, le criblage virtuel de médicaments est un outil utile pour orienter la recherche thérapeutique tout au long du processus de conception de médicaments. En particulier, la prédiction d'interaction médicament-cible est un outil utile pour déchiffrer les mécanismes biologiques sous-jacents d'un médicament, ce qui peut aider à identifier les effets secondaires néfastes ainsi que d'autres indications du médicament. Au cours des trois dernières années, nous avons participé à trois études de repositionnement de médicaments visant à faire face au cancer du sein triple négatif ou à la mucovisidose. Dans ce chapitre, nous présentons deux de ces projets et résumons notre contribution. En particulier, pour chacun des deux projets, nous mettons en évidence des voies thérapeutiques non liées à celles utilisées pour traiter les patients actuellement.*

In this chapter, we present two "real-life" applications of chemogenomics in which we have been involved: cystic fibrosis and triple negative breast cancer, two life threatening diseases. In both cases, better understanding of disease mechanisms at the molecular levels are needed, and there is an urgent need to improve the clinical management of patients. This includes better stratification to allow precision medicine, identification of the mechanism of action of drugs currently used to understand there limitations and identify new therapeutic targets and drugs. In both cases, these studies were undertaken in collaboration with biologists and medical doctors involved in clinical research, so that potential results could benefit from translational clinical research, in order to reach the patients. Indeed, for the cystic fibrosis project, we collaborate with clinicians and biologists from University Paris V and Hopital Necker-Enfants Malades. For the triple negative breast cancer project, we collaborate with clinicians and biologists from Institut Curie.

5.1 Drug virtual screening for cystic fibrosis research

5.1.1 Introduction

Cystic Fibrosis (CF) is the most common genetic disease among Caucasians, affecting approximately one in 3500 birth[193], and one person in 25 is an asymptomatic heterozygote carrier. It is an autosomal recessive genetic disease caused by mutations in the 250-kb CFTR gene, located in chromosome 7, structured in 27 exons, and coding for the Cystic Fibrosis Transmembrane Regulator protein (CFTR) [194].

CFTR belongs to the ATP Binding Cassette (ABC) superfamily of proteins, and shares their overall structural organisation consisting of two trans-membrane (TM) domains each consisting of six helices, two Nucleotide Binding Domains (NBDs), one Regulatory domain R situated between the first NBD and the second TM, with several consensus sequences for phosphorylation by the AMPc dependent protein kinase A (PKA) and protein kinase C (PKC) [195].

CFTR is a chloride ion transporter situated at the apical membrane of several polarised epithelia [196]. However, CFTR is also a transporter for larger other organic anions such as glutathione [197] or thiocyanate [198], and is involved in many other cellular functions (probably not all discovered) via its interactions with various protein partners.

Default in ion transport reduces airway surface liquid hydration, which prevents efficient mucociliary clearance, one of the basic immune defence mechanisms of the respiratory tract [199]. CF lung disease is characterised by chronic infections of Haemophilus influenza, Staphylococcus aureus, and Peudomonas aeroginosa. These infections cause progressive destruction of pulmonary tissue, ultimately leading to respiratory insufficiency, reduced quality and expectancy of life [200]. These infections are in turn responsible for chronic inflammation of airways, which participates to tissue damage.

Although respiratory failure is responsible for 80% of CF deaths, the disease is also associated with other symptoms [201]. In particular, exocrine pancreatic insufficiency occurs in the majority of CF patients. Indeed, thickened secretions from the pancreas, an organ responsible for secretion of digestive juices, lead to difficulties in digestion and absorption of nutrients. The endocrine function of the pancreas may also be affected, and some patients develop diabetes [202]. In addition, some patients also develop liver disease due

to biliary fibrosis that may progress to cirrhosis and portal hypertension [203].

However, the link between deficient chloride channel and the overall physiopathology of CF is not fully understood. In particular, it has been established that CF is characterised by chronic inflammation even in absence of infection [204], which cannot be related to CFTR chloride channel function in a direct manner. Similarly, other abnormal cellular phenotypes observed in CF such as increased oxydative stress, cell proliferation [205, 206]), proteostasis, and autophagy [207] do not appear to be related to a defect in chloride conductance. These remarks are consistent with the fact that, pathways enriched in genes differentially expressed in airways cells of CF versus control patients include inflammatory response, cell growth, proliferation and death, and immune cell trafficking pathways [208].

Life expectancy of CF patients has dramatically improved thanks to early diagnosis, and better health care based on nutrition support, and on limiting and treating lung damage: the median age of survival increased from 6 months in 1960 to around 40 years in 2010 [209]. However, management of CF mainly remained symptomatic until very recently, and deciphering the links between the gene defect and all the resulting molecular perturbations is critical in order to understand the overall physiopathology of the disease, and identify new potential therapeutic targets.

5.1.2 Impact of mutations in the CFTR gene on the function of CFTR protein

The CFTR gene is 230.000 base pairs long and encodes the 1480 amino acid long CFTR protein.

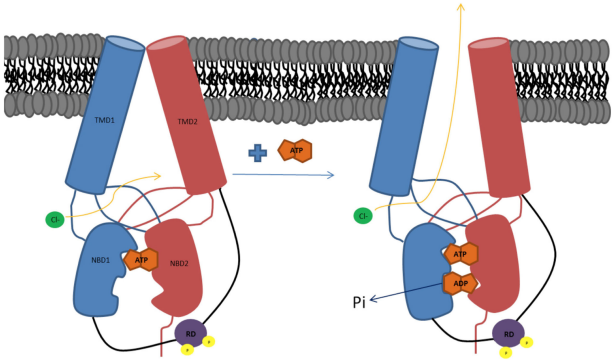


Figure 5.1. Schematic view of CFTR functioning (illustration from [210])

Schematically, the R domain of CFTR is proposed to partly block dimerisation of the two NBDs, which keeps the channel closed. Phosphorylation of R is accomplished by activation of other surface receptors that are coupled to adenyle cylase, leading to a local elevation in cAMP concentration and activation of PKA. Then, PKA phosphorylates the R domain, causing structural changes that remove R from its steric-interfering position and allows dimerisation of the NBDs [211] (see Fig. 5.5, discussed below). Finally, ATP binding at the NBDs initiates channel opening, while ATP hydrolysis and subsequent ADP release closes the channel [212].

More than 1700 mutations have been observed in CF, and the Cystic Fibrosis Genetic Analysis Consortium maintains a mutation database available for the research community. However, a small number of mutations are more frequent than others and have been studied in more details. In particular, the three more frequent mutations occur in the NBD1 domain: ΔF508 (deletion of F508), G542X and G551D mutations are found in approximately 80%, 3% and 2% mutated alleles, all other mutations representing the other 25% mutations. The phenotypic or functional impact of mutations have been grouped into five classes [213]:

1. class 1: protein production mutations include nonsense and splice mutations interfering with the production of CFTR protein.

2. class 2: protein processing mutations that cause a perturbation of the 3D structure of CFTR that cannot complete its full maturation and insertion in the membrane.

3. class 3: gating mutations block the gate in the closed position even in presence of ATP, so that chloride ions cannot go through.

4. class 4: conduction mutations change the shape inside the tunnel so that chloride ions cannot move through easily.

5. class 5: insufficient protein mutations result in a reduced amount of CFTR protein at the membrane surface. This can be due to insufficient protein production, an insufficient number of functional proteins at the surface, or proteins that degrade too quickly.

In the case of the three most frequent mutations, the ΔF508 mutant belongs to class 2, G542X to class 1, and G551D belongs to class 3.

5.1.3 Therapies for the rescue of CFTR processing or activation

Besides the progress in patients' symptoms management, three molecules have been progressively approved by the FDA since 2012. These molecules tend to target the basic defect in CF, i.e. mutated CFTR, and were identified by phenotypic HTS screens based on improving chloride channel transport in human CF bronchial cells. In particular, this screen did neither allow to identify the mechanism by which chloride current were increased, nor prove that this mechanism involved CFTR. Collectively, these three drugs are called "CFTR modulators", although their effects fall in two categories.

Ivacaftor (Kalydeco, VX-770, DrugBank ID DB08820, PubChem ID 16220172) is a "potentiator" of CFTR, which means that it activates its chloride channel function by increasing the probability of channel opening (or gating) in patients with impaired gating mechanism. This is typically the case for the G551D mutation, and more generally, for patients with class 3 mutations.

Lumacaftor (VX-809, DrugBank ID BD09280, PubChem ID 46199646) and tezacaftor (VX-661, DrugBank ID DB11712) are "CFTR correctors" that partially correct the folding, processing, and maturation defects of CFTR towards the cell surface, increasing the amount (mutated) CFTR at the membrane. These drugs are indicated for class 2 mutations such as ΔF508 which, when present at the membrane, do present some residual activity.

In the case most frequent case of patients bearing the ΔF508 mutation, a combination of ivacaftor and lumacaftor, called Orkambi, can be administrated, in order to both increase the quantity of protein present at the surface of the cell, and activate its function. Orkambi leads to an improvement of 10% in the FEV1 (forced expiration vital capacity) in about 30% of the patients, but more modest responders in terms of FEV1 tend to benefit from reduced inflammation [214], infections, antibiotic use, hospitalisation rate, and better quality of life [215].

Various studies have tackled the question of how these drugs act at the molecular level. In human CF model of cell lines, lumacaftor has been shown to increase the amount of functional ΔF508 CFTR present at the cell surface by 34% [216]. In cultured human bronchial epithelial cells, lumacaftor improved trafficking of ΔF508 CFTR from 14% [217]. The precise mechanism for better maturation of this CFTR mutant is not known, and in particular, it has not been demonstrated that lumacaftor directly binds to ΔF508 CFTR to promote its maturation. Ivacaftor is believed to directly bind to CFTR, causing opening of the wild type and G551D CFTR via a non-conventional phosphorylation-dependent but ATP-independent mechanism (remember that G551D CFTR cannot hydrolyse ATP), possibly by direct binding to phosphorylated CFTR [218]. However, there is not direct evidence for such mechanism. Even more, ivacaftor may activate the function of different classes of mutants through different mechanisms. In particular, class 2 mutants like ΔF508 CFTR, and class 3 mutants like G551D CFTR might be activated by different routes.

Overall, the molecular mechanisms by which these three newly approved drugs (available for french patients over 12 years old since about 3 years) modulate CFTR are not fully understood, and could result from indirect interactions with CFTR.

In addition, their role as CFTR modulators does not account for some of the benefits observed in patients with otherwise limited FEV1 increase and improvement in CFTR function, such as reduction in inflammation and infection episodes [219], better nutritional status and quality of life. For example, lumacaftor was found to stimulate phagocytosis and killing of Peudomonas aeroginosa by macrophage [220], which cannot be related in a direct manner solely to improvement in CFTR trafficking.

These observations suggest that ivacaftor, lumacaftor and tezacaftor might also act through off-target proteins, and that these interactions might account for some of the patients' clinical benefits.

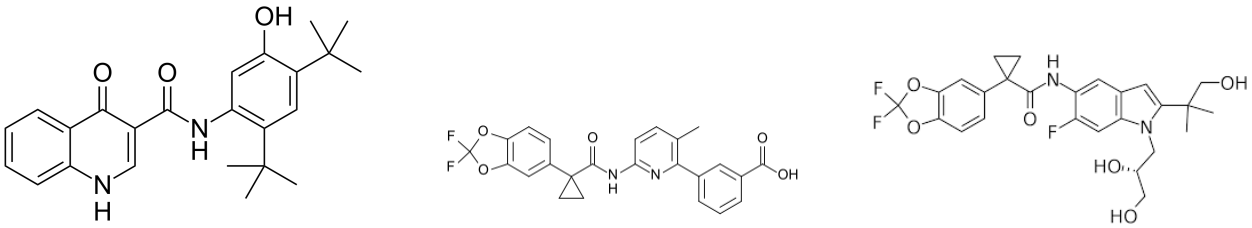
In line with this idea, a recent article showed that ivacaftor binds to several GPCR receptors that are expressed in the central nervous system: 5-HT_{2C} serotonin receptor, the β3-adrenergic receptor, and δ-opioid receptor, and it was postulated that this could contribute to the benefit on patients' mood, observed under ivacaftor therapy [221].

However, other targets might still have to be discovered.

To summarise the above discussion: (i) the mechanisms of action of CFTR modulators are not known, neither with respect to how they rescue CFTR processing and function, nor with respect to some of their other clinical benefits, and (ii) there are strong presumptions that the overall effects of CFTR modulators rely on their interactions with unidentified target proteins.

In addition, besides these presumptions, drugs are in general expected to interact with more than a single target [25], although designing experiments aiming at testing interactions between these drugs and other human proteins is not feasible on a large scale.

In this context, we undertook in silico approaches to predict interactions between ivacaftor, lumacaftor and tezacaftor and human proteins, in order to guide future biological experiments involving the most probable target proteins. We display the Lewis formula of the three compounds in Fig. 5.2.



(a) Ivacaftor (b) Lumacaftor (c) Tezacaftor

Figure 5.2.

Finally, we would like to mention that, besides the question of mechanism of action, other questions about CFTR modulators remain open, although they are of critical clinical interest. In particular, cystic fibrosis patients take many different drugs to manage various pathological aspects of their disease, and these drugs may lead to deleterious interactions with CFTR modulators if that target similar (or antagonist) molecular mechanisms, as discussed below. Identification of CFTR modulator targets could help to explain and anticipate these effects, and to optimise patients treatments in order to avoid these effects, which is one aspect of precision medicine.

In addition, it is anticipated that CFTR modulators should be taken life-long by patients who are clinically improved by these drugs. There is today no idea about what long-term side-effects those patients might develop, which is somewhat worrying considering the young age at which patients will start these therapies. Identification of CFTR modulators targets may also provide very important clues to answer this question.

5.1.4 Prediction of protein targets using chemogenomics for CFTR new modulators

Various chemogenomics methods have been proposed in the last decade, as discussed in detail previously. We used a multi-task learning algorithm [169] based on SVM with kernels. For proteins, we used the sequenced-based Local Alignment kernel (LA kernel). For small molecules (the ligands), we used the random walks kernel on the graph defined by the structure of molecules. As discussed earlier, this chemogenomics approach was shown to display state-of-the-art prediction performance [50].

More precisely, we trained our model with all known protein-molecule interactions available at the DrugBank database, involving a human protein and a small organic molecule, many of them being marketed drugs or molecules that were studied in the context of drug development. We made this choice because: (i) we were looking for human protein targets (not bacterial or virus targets) belonging to available the human "druggable" proteome, (ii) ivacaftor, lumacaftor and tezacaftor being drug molecules, a model trained with protein target information available for drug-like molecules was probably well suited to make predictions for other drug molecules like CFTR modulators, and (iii) one important long-term goal for this project is to identify known marketed drugs that could be repositioned in cystic fibrosis, to allow shorter development time.

Overall, the prediction model was trained based on 2570 proteins, 4834 molecules, corresponding to around 13,000 interactions available at DrugBank v5.1. Note that we considered an more recent version of the DrugBank database than the one we used in the previous study in chapter 4. Cross-validation studies showed that most of the known interactions had prediction scores above 0.9, whereas the distribution of unknown interactions had little overlap and corresponded to interactions of much lower scores (see fig. 5.3).

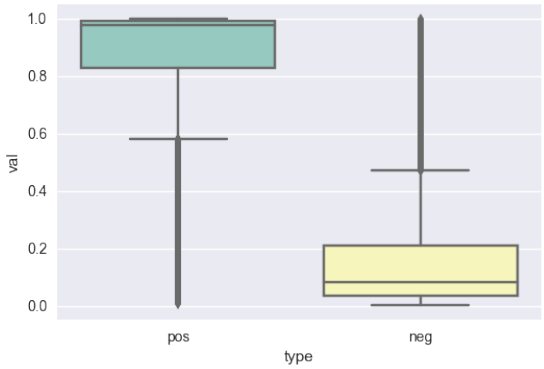


Figure 5.3. Distribution of known and unknown interaction prediction probabilities observed by cross-validation.

Then, we used this model to predict the probabilities of interaction of ivacaftor, lumacaftor and tezacaftor for the 2570 considered human proteins. We did not take into account any known targets for the three drugs, which means that we considered them to be orphan drugs. Indeed, CFTR is the only target reported for these drugs at DrugBank, although it has not been proven that any of them does interact directly with CFTR. Note that the only proven direct interactions (besides the proteins serum carriers albumine and Alpha-1-acid glycoprotein 1), at least to our knowledge, are those of ivacaftor with the GPCRs mentioned above, but these interactions are not updated yet in DrugBank. This lead, for the three molecules, to a ranked list of proteins with decreasing binding probability for binding. For each molecule, we analysed the top-ranked proteins based on the available literature on CFTR and cystic fibrosis, to select a short list of proteins that were expected to be realistic off-target candidates. These analyses aimed at guiding future experimental work devoted to confirm or infirm these predictions.

We limited the analyses to predicted interactions with very high scores, above 0.9, except for tezacaftor for which we considered scores down to 0.8 because overall, the prediction scores for this molecule were lower than for the two others. These thresholds were consistent with the distribution scores obtained for known interactions by cross validation (as shown in Fig. 5.3). With these criteria, 10 to 25 protein targets were shortlisted for each molecule. This number is relevant since, although a given molecule is expected to have more than one protein target in the cell, it is however expected to have only a small number of targets with respect to the 2570 considered human proteins. We report the predicted targets for each of the three drugs in Table 5.1.

	highly predicted targets (targeting probability higher than 0.9)
for ivacaftor, lumacaftor and tezacaftor	NR1I2 -PXR-, PPARG, F2, CDK2, MAPK14
for ivacaftor and lumacaftor only	CHECK1, PTGS2, SRC, ESR1, PTGS1
for lumacaftor and tezacaftor only	BACE1, PTPN1, NOS2
for ivacaftor and tezacaftor only	TNF, ACHE, ALOX5, AR
for ivacaftor only	AHR, HSP90AA1, PIM1, GSK3B, CALM1, MAPK10, ADRA2A, ADRA2B, ADRA2C, ADRA1A, ADRA1B
for lumacaftor only	KDR, PPARA, RXRA, F10
for tezacaftor only	ADRB2, KCNH2

Table 5.1. Predicted targets for ivacaftor, lumacaftor and tezacaftor.

We will not discuss all predictions, although most of them appeared extremely interesting in the context of biology of CF (this is out of the scope of this manuscript). We will restrain our analysis to predictions for which we could find strong evidence in the literature, and which appeared the most promising in terms of potential drug repositioning could be foreseen. In particular, these predictions are discussed in the light of current knowledge in CF.

Somewhat unexpectedly, the three molecules were found to share many of their best-ranked targets. This suggests that, besides their roles as CFTR modulators, they may share some other targets and mechanisms of action that contribute to the clinical benefit observed in patients, or share common metabolism pathways. Some of the best-predicted targets for which we found direct or indirect evidence in the literature, are discussed below.

Among the best ranked proteins, NR1I2 (also known as Pregnane X Receptor, PXR) and AHR (Aryl Hydrocarbon Receptor) are two nuclear receptors involved in detoxification of endogenous or exogenous toxic compounds. They are both characterised by a ligand-binding domain, and a DNA-binding domain, that, after translocation to the nucleus, controls the expression of the targeted genes. In particular, they bind to the regulatory regions of the CYP3A4 gene, encoding for a member from the cytochrome P450 family of major drugs phase I metabolising enzymes. We found several papers reporting that ivacaftor (and its metabolites) and lumacaftor, but not tezacaftor, are strong inducer of CYP3A4 expression [222]. The

three compounds induce the expression of CYP1A2, another gene from the cytochrome P450 family that is targeted by NR1I2 [223]. Taken together, these results are consistent with our predictions that the three CFTR modulators could be ligands for NRI12 and/or AHR. In the present case, the predicted interactions are related to the metabolism pathways that process the drugs, and not to their mechanism of action. However, there are strong evidence in the literature that the corresponding high scored predicted interactions with NR1I2 and/or AHR could be true, and these interactions would provide the precise molecular mechanism phase I metabolising for these drugs.

These results have strong clinical implications, since they explain some drug-drug interactions that have been reported. Among others, administration of ibuprofen (an anti-inflammatory drug commonly used in CF) together with lumacaftor/tezacaftor lead to sub-therapeutic concentration of ibuprofen, and one hypothesis it could be due to activation of CYP enzymes [224]. Similarly, strong drug-drug interactions were reported between ivacaftor and rifampin, an antibacterial drug commonly used to fight lung infections in CF. The authors also speculated that the two drugs might have reduce each other's concentration in blood, leading to sub-therapeutic levels [225]. Even more, one paper reported that lumacaftor might reduce the plasma concentration of ivacaftor, when the two molecules are co-administrated, which could limit the clinical effect of this drug combination [222].

Taken together, our predictions and the above observations indicate that one should avoid concomitant administration of ivacaftor/lumacaftor and other drugs that would either be inducers (via binding to NI1I2 or AHR) or substrates of CYP enzymes.

TNFα is another highly ranked target for the three modulators, particularly for ivacaftor and tezacaftor. TNFα is a cell signalling protein (a cytokine) involved in systemic inflammation, and TNFα gene is known to modulate CF disease severity [226]. We were not able to find any published work that explicitly confirmed direct interaction between CFTR modulators and TNFα. However, the Bioassay database, available at the pubchem database, indicates that ivacaftor was found active in Biossay AID 624479, an HTS screen for identification of small molecule antagonists for the TNFα/NFkB signalling pathway. NFkB is one of the downstream transcription factors activated by TNFα, leading to expression of a series of pro-inflammatory cytokines such as IL2, IL6, or IL8, and to anti-apoptotic factors and cell cycles regulators, leading to cell survival and proliferation. These two phenotypes (inflammation and proliferation) are well documented in CF, as mentioned above.

In addition, a paper tested the interest of the anti-inflammatory drug Sulindac in CF cells [227]. They studied the interest of this drug by following its effect on inhibition of the TNFα/NFkB signalling pathway, using a reporting gene whose expression is correlated to the activity of the promoters targeted by NFkB (in fact, they used a biological system similar to that of Biossay AID 624479). They also followed the expression level of IL8. In fact, they happen to also test lumacaftor, which indeed lead a decrease in the reporter gene signal, and to a decrease in IL8 expression. This means that lumacaftor does inhibit the TNFα/NFkB signalling pathway. However, this was not their interest in their study, and although this result was evident in their figures and tables, they did not make any comment or analysis. This does not prove that lumacaftor in a direct inhibitor of TNFα, but we can conclude that it, at least, inhibits one of the proteins in the

TNFα/NFkB cascade.

Finally, various papers report that pro-inflammatory cytokines from this pathway are up regulated in CF [219], and that IL8 and IL6 levels are reduced upon treatment with ivacaftor [228]. Overall, these results indicate that some of the known molecular effects of ivacaftor (and potentially of lumacaftor and tezacaftor) that are not explained by CFTR modulation, could arise from a mechanism involving inhibition of TNFα pathway (TNFα/NFκB can be found in figure 5.4), potentially by direct interaction with TNFα.

If confirmed, this could have strong implications in CF. First, it could reveal that dysregulation of the TNFα/NFκB pathway could be the causing mechanism for the intrinsic inflammation observed in the airways of CF patients, even in absence of infection. The link between CFTR mutation and dysregulation of this pathway is not be explained yet, although some studies have shown that activation of the TNFα/NFκB and inhibition of CFTR function are correlated [229]. They suggest that an increase in TNFα levels could be a cellular response in an attempt to favour processing of ΔF508-CFTR. Second, hyper-activation of this pathway is also observed in other inflammatory diseases such as rheumatoid polyarthritis, and therefore, TNFα antagonists are available on the market, and might be stronger and more specific inhibitors than the three CFTR modulator molecules. They would undoubtedly be interesting candidates for repositioning in CF. Indeed, while it is not clear whether inhibition of TNFα alone would be a good strategy in CF, since it would both reduce inflammation and ΔF508-CFTR processing, a bi-therapy with CFTR modulators and TNFα inhibitors would allow increase of expression and function of ΔF508-CFTR, while decrease chronic inflammation, one of the most deleterious symptoms in CF.

These results urged our collaborator to undertake experiments to evaluate our prediction of TNFα as a target for CFTR modulators. Direct interaction between CFTR modulators will be tested on purified TNFα by mass spectroscopy, in the laboratory of Professor Olivier Laprevote. In addition, this interaction will also be assayed by native gel electrophoresis in the laboratory of Professeur Isabelle Sermet-Gaudelus. In both cases, the principle will be to search for conversion of the trimer active form of TNFα into a dimer form, as observed in presence of other small molecules inhibitors of TNFα [230]. In addition, they will undertake transcriptomic experiments on primary culture of CF patients airways epithelial cells, in order to further study the effect of TNFα activation or inhibition on ΔF508-CFTR processing (with, and without CFTR correctors), and on downstream cytokines.

Other interesting predictions are the high scores predicted for interaction of ivacaftor with adrenergic receptors, in particular with α1-, α2-, and β2-adrenergic receptors (respectively ADRA1, ADRA2, and ADRB2). Adrenergic receptors (ADRs) form a family of G-coupled receptors involved in signal transduction of various cellular functions such as smooth muscle contraction and relaxation.

As mentioned above, it has been shown that ivacatfor binds to the β3-adrenergic receptor (ADRB3), an information that was not used in our chemogenomics predictions, since we considered all three CFTR modulators as orphan proteins. ADRB2 and ADRB3 display high sequence similarities (41 % sequence identity, and above 70 % sequence similarity, being in fact two isoforms expressed in different tissues), and

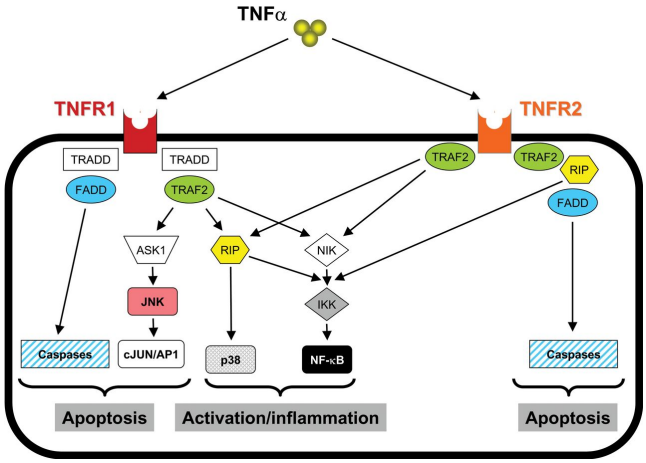


Figure 5.4. TNFa/NFkB pathway.

are known to share some ligands. For example, according the the Unipro entries of ADRB2 and ADRB3 (respectively P07550 and P13945) Amitriptyline, Amphetamine, Arbutamine, Bopindolol, Mephentermine, Propanolol, and many other molecules bind both to ADRB2 and ADRB3. This allows to conclude that our prediction of ADRB2 as a target for ivacaftor is highly realistic.

Consistent with this conclusion, many papers underline the functional links between ADRs and CFTR. In particular, it was shown that ADRB2, the major adrenergic receptor isoform expressed in airway epithelia, colocalises with CFTR at the apical membrane, and forms a macromolecular complex involving CFTR-EBP50-ADRB2 through PDZ-based interactions.

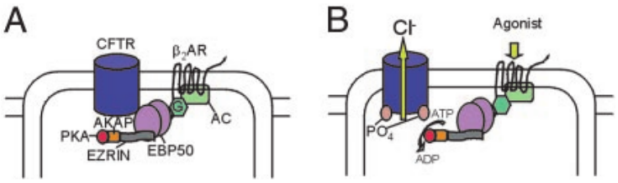


Figure 5.5. Scheme of the functional interaction of ADRB2 and CFTR (picture from [211]).

Thus, CFTR physically and functionally interacts with ADBR2, and stimulation of the later by agonists ligands increases CFTR chloride channel activity in airway epithelial cells (a scheme of the interaction is reported in Fig. 5.5). Assembly of the complex is regulated by PKA-dependent phosphorylation, and deletion of the regulatory R domain of CFTR abolishes PKA regulation of complex assembly [211]. Conversely, it was proven that antagonists of ADRB2 inhibit CFTR chloride secretion [231]. Interestingly, a study reports that ivacaftor activates CFTR function, provided that it is not already maximally activated by ADRB agonists [232].

Taken together, these results (and many others not discussed here) strongly suggest that, besides being a ligand for ADRB3, ivacaftor might also be an agonist ligand for ADRB2, and potentially other adrenergic receptors. This could highlight its mechanism of action, at least in the case of the most common ΔF508 mutant (and other mutants displaying some remaining activity), and would have considerable clinical implications. Indeed, many CF patients receive ADRB broncho-dilatators agonists such as albuterol, and chronic or excess of such treatment was found to reduce by an amount of 60% the activation of wild-type or ΔF508 mutant[233]. This could explain, at least in part, why half of CF patients receiving ivacaftor do not benefit from FEV1 improvement. It would be interesting to analyse whether there exists some anti-correlation between broncho-dilatator treatment and absence of response to ivacaftor. It would also mean that the mechanism of action of ivacaftor could be different for mutants that retain some activity (like the most frequent mutant ΔF508, R117H or A455E) and for mutants that are processed to the membrane, but inactive because unable to hydrolyse ATP, like the G551D mutant. Indeed, the the case of the G551D mutant, although not directly proven, a series of experiments tended to prove that activation of G551D-CFTR by ivacaftor could be the result of direct interaction between these two partners [234].

Taken together, our results and the above observations strongly suggest the mechanism by which ivacaftor activates the function of ΔF508-CFTR could be binding and activation of ADRB2, leading to activation of CFTR through the protein complex shown in Fig. 5.5.

If this is confirmed, it would be of considerable relevance in CF. First, since many CF patients already receive ADRB2 agonist broncho-dilatators as part of their daily medication, one could wonder whether these patients do benefit from ivacaftor therapy, or do not because the function of their mutated CFTR is already maximally activated through this route. This could explain part of the 70% patients who do not improve CFTR function and FEV1 under this therapy. This is an assumption that our clinician collaborators in Hopital Necker will evaluate by analysing the record of their CF patients receiving ivacaftor. For all ΔF508 patients, one interesting question would be to wonder which of the two combinations, i.e. lumacaftor/ivacaftor or lumacaftor/ADRB2 agonists(broncho-dilatator), would be the best choice in terms of ΔF508-CFTR activation and safety profile (side-effects, drug-drug interaction). In addition, since ADRB2 agonist molecules are also available as oral drugs. Therefore, it would also be interesting to investigate whether they would be more effective to activate ΔF508-CFTR that the nebulized broncho-dilatators, and whether they would be better choices for activation of ΔF508-CFTR in organs other than the lung, but also affected by CF (for example, the digestive track).

Finally, among other predictions of interest for CF, we will briefly comment those involving protein kinases. As can be seen in Table 5.1, several kinases were predicted with very high scores as ligands for one or several of the three CFTR modulators: MAPK14 and MAPK10, CDK2, CHECK1, SRC, and GSK3B. Remember that we used a very high threshold of 0.9 in score, which means that at lower (although still high) scores, these and other kinases are predicted ligands for one or several or the three considered molecules. There are so many publications that underline cross-talks between CFTR and kinases (see [235]), that it is out of the scope of this manuscript to make a comprehensive review. However, we can mention that

these links are related to many phenotypes and symptoms of CF, at the cellular or clinical levels, including: autophagy, inflammation, CFTR maturation and proteolysis, cell proliferation and so on. However, we would like to shortly mention some of the papers (among many others) providing the most convincing clues about potential interactions between our best predicted kinases and CFTR modulators. For SRC, refer to [236]. For MAPK kinases, refer to [237]. For kinase inhibitors in general, see [238].

It would therefore be of great interest to perform experiments dedicated to study the target profile of CFTR modulators at the human kinome level (around 500 kinases), or at least against some of the most promising kinase subfamilies such as MAP, the SRC, and the GSK3B kinases. Some companies such as Discoverex offer these services for a few thousand of euros to a few tens of thousand of euros, depending on the number of tested molecules and kinases, based on the so-called Kinomescan test. We think that gathering some money to perform such tests is an important issue in CF research. Indeed, kinases are extensively studied proteins, because they are involved in many diseases such as cancer. Therefore, many drugs are already available against various kinases, and some of them may also be candidate for repositioning in CF.

5.1.5 Conclusion and perspectives

Overall, chemogenomics predictions allow to propose highly realistic protein targets for CFTR modulators. Our predictions also highlight that the global effects (beneficial or limiting) of these drugs might result from several mechanisms of action involving diverse targets, in addition to CFTR (although their direct interaction with this protein is not clearly demonstrated).

They will guide future experiments to better decipher the complex mechanisms of action of CFTR modulators, which illustrates one of the contributions expected from chemogenomics approaches, as stated in introduction of this manuscript. Indeed, our predictions allowed to understand otherwise known biological facts that were not interpretable without guiding the intuition with the predicted targets. More generally, these results allow to analyse available literature in a more comprehensive manner.

On longer terms, identification of new therapeutic targets in CF (such as adrenergic receptors, kinases or TNFα) can also help to build a systems biology view of the disease, and better understand the link between the mutated CFTR gene and the defect in the chloride channel, to the overall complex physiopathology of CF.

We would like to add that, our predictions appeared convincing enough to CF experts to be the starting point of a wide collaboration involving in silico approaches based on systems biology and machine-learning approaches, and experimental approaches based on ad hoc experiments (devoted to predictions evaluation) and omics approaches of CF. This collaboration includes computational biologists from our group, from Institut Curie and from Institut Imagine, together with experimental biologists from Universite Paris 5 and Institut Imagine. Our results are also the starting point of a wider project involving clinicians from Hopital Necker in Paris, from the Oslo University Hospital, and from the Val d'Hebron hospital in Barcelona. This project is currently under evaluation of the european ERACoSysMed call for translational medicine.

5.2 Virtual screening for triple-negative breast cancer

5.2.1 Introduction

Institut Curie occupies a leading position in breast cancer research and treatment. The incidence of breast cancer is 54.000 in France each year, and one out of eight women will develop this cancer. Various classifications are used in breast cancer, but an important classification is based on the receptors that are expressed at the surface of cancer cells, because they condition therapeutic strategy.

In practice, the presence of three receptors is searched: progesterone receptor (PR), estrogen receptor (ER), and the human epidermal growth factor receptor 2 (HER2). The growth of PR and ER-positive tumours are fuelled respectively by progesterone and estrogen hormones, and therefore, hormonal therapy is used to decrease the level of these hormones. Two cancers out of three are positive for at least one of these two hormone receptors. In HER2-positive breast cancer, HER2 promotes the growth of cancer cells, and specific targeted therapy using antibodies are very effective. One cancer out of five is HER2+.

Unfortunately, 10 to 20% of breast cancers express none of these receptors, and are consequently called triple negative breast cancers (TNBC). These cancers tend to be aggressive, and have a poorer prognosis than the others. No specific drugs are available for them, precisely because they do not express a known protein target, and TNBC is probably a heterogeneous disease because cancers are classified as TNBC by default.

Therefore, there is an urgent need to find new drugs for these cancers, and to identify their biological characteristics. In a context where searching and developing a new drug is a very long and expensive process, one alternative, that we explored in this project, was again to favour new therapies against TNBC among marketed drugs.

The group of Prof. Fabien Reyal is very active in TNBC research, and performs experimental screens on TNBC cell lines to search for molecules able to kill these cancer cells. In particular, they screened 12 TNBC and two control cell lines using two complementary phenotypic assays: a fluorescent cytotoxicity assay [239], and luminescent cell viability assay [240]. More precisely, they screened 800 FDA-approved molecules, in addition to 80 kinase inhibitors, 33 phosphatase inhibitors, and 53 protease inhibitors, because these three classes of proteins are known to be potential cancer drug targets. A second goal was to identify markers for drug sensitivity or resistance, in order to define which patients, among TNBC patients, would benefit from potentially identified drugs. This would also allow to stratify TNBC patients into coherent sub-groups, based on these new biological markers.

Hit molecules were defined as molecules leading to less than 10% of viable cells in the viability test, or 90% of killed cells in the cytotoxicity test. With these criteria, the biologists identified 114 active molecules in the viability test, and 119 active molecules in the cytotoxicity test. A total of 94 molecules were common to the two sets, and were chosen as primary hits.

Although 966 molecules could be studied in these two preliminary assays, more extensive tests had to be undertaken to better characterise hits. In particular, it was necessary to perform dose-response experiments, which was possible for a maximum of 80 molecules, given the time, budget and human resources for the

project.

Our role in the project was first to help select 80 molecules out of the 94 hits.

Reducing the number of hits from 94 to 80 did not appear to be a strong challenge. The most simple idea was to measure pair-wise similarities between the 94 hits, and search for highly similar pairs of molecules that could be considered as "doublons". Although we are aware that subtle structural differences might sometimes strongly modulate the activity of a molecule, this appeared to be a reasonable strategy. Indeed, at this stage, the idea was to select hits that span the chemical space of active molecules as much as possible, in order to cover the larger possible panoply of underlying mechanisms, and identify the wider possible spectrum of TNBC subtypes.

Therefore, we computed the pairwise distance between the 94 identified molecules, based on their chemical graph, and manually checked whether or not the molecules forming the closest pairs shared enough chemical similarity. In particular, we used the Marginalised kernel to assess molecular similarities, computed via the Chemcpp package (cf section 2.3.3), to assess the structural distance between molecules.

We identified 14 pairs of very high similarity and considered that, in a context where all molecules could not be further tested, one of the two molecules in each of these pairs could be considered as redundant and removed from the hits (two examples of such very similar pairs are displayed in figure 5.6).

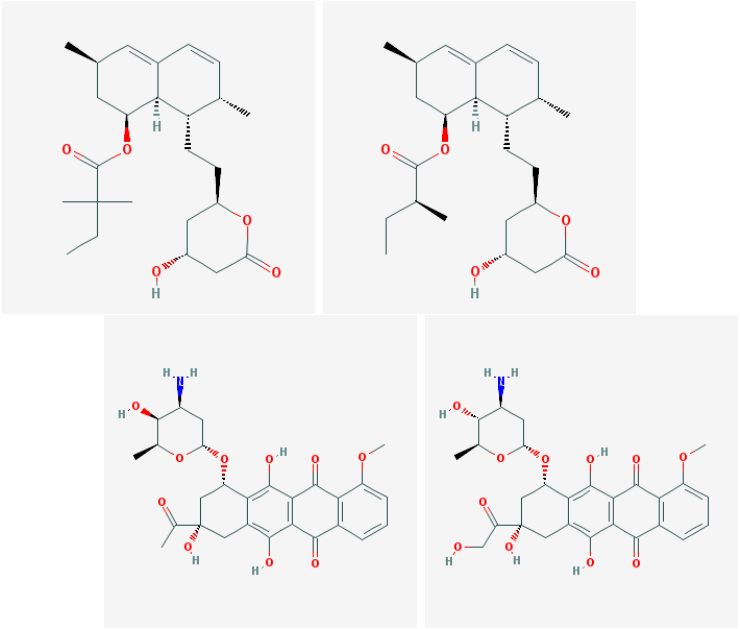


Figure 5.6. Examples of pairs of very high similarity

This allowed to reduce the number of hits to further analyse to the desired number of 80.

5.2.2 Characterisation of the hits

Our goal was to further characterise the hits, based on their chemical structure, targets (known or predicted), and mechanism of action, in order to help select the most promising compounds to be sent for further costly analysis.

Also, it was essential to identify the mechanism of action of the 80 molecules in order to point at potential biological markers to be used for stratifying patients and predict the response to treatment. For each hit molecule, some targets could be known or not. And, among the known targets (if several), the one that was responsible of the activity in TNBC could be different from that responsible of the activity against the disease for which the drug was initially marketed. It was also interesting to identify whether the mechanisms of action of hits were similar, or at least, related to a limited number of pathways. This would help better characterise TNBC, and understand how they differ from the three other breast cancer types, besides the default of expression in PR, ER, and HER2.

In essence, the aim was to understand why the 80 hits were active against the twelve TNBC cell lines, and identify potential protein targets that would be of interest for TNBC biomarkers.

Characterisation of hits molecular structure

To establish whether the hits shared some common structural characteristics, we clustered the 966 initially considered molecules with the ward-linkage agglomerative clustering algorithm (cf. appendix A.6), based on the molecular distance defined by the Marginalised kernel. The same clustering was repeated on the hit molecules only.

We determined the number of clusters based on the silhouette score, which compares the average intra-cluster distance with the inter-cluster distance. The silhouette score can vary from -1 (inter-cluster distance is 0) to 1 (average intra-cluster distance is 0). By comparing the molecules within the different clusters for all possible number of clusters, we determined that the optimal number of clusters was 5. Indeed, adding more clusters was only creating clusters containing a few molecules whose silhouette scores were below 0. The optimal number of 5 clusters was found both for the 966 molecules and the 80 hits.

Analysing the position of hits within the clusters of the 966 molecules, we found that the hits were spread over the 5 clusters. In addition we found that the 5 clusters obtained with the hits clustering had a one by one correspondence with those obtained with the 966 tested molecules. Therefore, the hits appeared representative of the chemical diversity of all tested molecules, and were not restricted to a specific family of molecular structures. This did not allow to identify particular molecular scaffolds of interest or mechanisms of action. However, it indicated that various options were potentially at hand for TNBC, and that TNBC was probably a diverse group of tumours [241], reflected by the fact that could be sensitive to different types of molecules.

Since it was not possible to derive comprehensive information from the structures of hit molecules, we then tackled the question of identifying potential common mechanisms of action shared by molecules that are active against TNBC.

Characterisation of hits mechanism of action based on their protein target profiles

We studied whether the 80 hits were enriched in particular mechanisms of action, compared to those of the 966 tested molecules. More precisely, we described the hit molecules by their protein target profiles in order to identify clusters, similarly to the clustering performed on the chemical structures.

To this end, we searched for the known targets of the 966 molecules based in several available databases [117, 118], but also from pharmaceutical companies information, and literature search. We collected 785 unique targets, and observed that 20% of the molecules were orphan, 30% were associated with a single target, and the mean number of known targets per molecules was about 3. Among these 785 targets, 170 proteins were targets known for the 80 hit molecules.

Thus we described each molecule by a binary vector of length 785, corresponding to the total number of targets. Each element was set to 1 if the corresponding protein was a target for the considered molecule, and other-wise to 0. This lead to a large number of orthogonal molecules, because most molecules did not share any target (most of them had only 1 or 2 known targets), in addition to the fact that 20% of molecules were orphans. In such situation, it is not possible to use the Euclidean or the Jaccard distance to cluster molecules.

To address this issue, one idea is to use independent biological data that would allow to increase hits targets promiscuity, so that it would be possible to measure distances between sets of proteins. We used the protein-protein interaction (PPI) network available at the STRING database that records interactions between proteins, including direct (physical) and indirect (functional) associations recovered from computational prediction, knowledge transfer between organisms, and interactions aggregated from other (primary) databases [242]. This allowed to assess the similarity between proteins based on the diffusion kernel [243] computed on the PPI. This kernel assesses the similarity between two proteins based on the length of the shortest path bridging their two nodes.

From this similarity between proteins, we defined a symmetric similarity measure between molecules as:

$$sim(m_1, m_2) = \max_{k\in\{1,2\}} [\frac{1}{|target\_set(m_k)|} \sum_{i\in target\_set(m_k)} \max_{j\in target\_set(m_{l\in\{1,2\},\ l\neq k})} (K_{diff}(i,j))].$$

In the above equation, $target\_set(m_k)$ refers to the set of protein targets of molecules m_k and K_{diff} refers to the above mentionned diffusion kernel on the PPI network of the STRING database. The underlying idea was to measure molecule similarity using their target profiles more smoothly than with the direct one-to-one correspondence computed by the Euclidean distance between the target binary profile.

Clustering of molecules with the ward-linkage agglomerative clustering algorithm (cf. appendix A.6) based on this similarity measure lead to completely different clusters from those obtained with the structural similarity measure. More importantly, most of the molecules felt in a single and non-informative cluster, in which all molecules were as far from each other, than from the other clusters. This cluster did not bear comprehensive information about any associated mechanism of action, probably because of too many orphan molecules, or/and too many molecules whose targets are isolated in the PPI network. Thus, the similarity measure defined from the PPI network did not address the issue of lack of information.

At this stage, the conclusion was that the mechanisms of action were diverse (at least, based on the known targets), and it was not possible to cluster molecules based only on targets information.

Characterisation of hits mechanism of action via pathway enrichment analysis

Since the hit targets did not allow to identify key mechanisms of action that trigger TNBC sensitivity to drugs, we searched for potential enriched biological pathways. In other words, we searched whether the hits targets preferentially belonged to some biological pathways that would be critical for TNBC survival. As mentioned above, the number of targets per hit was very low (on average, between 0 and 2), and therefore, we chose to add high score predicted targets to the target profiles of hit molecules before performing pathway enrichment analysis.

Predictions were performed for the 966 tested molecules at the druggable human proteome scale, using the method we have proposed in [50]. More precisely, we built ten classifiers based on ten different training sets that comprised all the known human targets in the DrugBank database v.4, and an equally sized set of unknown interactions as negative samples. The final probability of a drug-target interaction was equal to the mean probability of interaction over the ten classifiers.

We defined a minimum probability threshold above which a predicted interaction was considered as "real". Similarly as in the cystic fibrosis project, we chose a threshold of 0.9, since this was the mean probability observed when predicting known targets by cross-validation. This lead to add around ten targets per hit molecule, corresponding a "reasonable" number of targets per drug, i.e. in the range that is typically expected for drug-like organic compounds [24]. The resulting in 1137 known and predicted protein targets for the 966 tested molecules. From these, 298 corresponded to the 80 hit molecules (we recall that they had 170 known targets).

This finally defined two lists of proteins that will be used for pathway enrichment analysis: a "reference" list comprising the 1137 known and predicted targets for the 966 tested molecules, and a "highlighted" list comprising the 298 known and predicted protein for the 80 hit molecules.

Pathway enrichment analysis aims at identifying whether a highlighted list of proteins (or genes) contains subgroups of proteins belonging to the same biological pathways (such as "cell cycle", "apoptosis", or "DNA repair"), in a quantity above than the expected one when randomly picking proteins in a reference set. Biological pathways are defined as a set of genes related to a specific biological function.

Results are usually provided as a real number for each pathway, corresponding to a p-value of a statistical test, based on the probability of observing the subset of highlighted genes from this pathway "just by chance". The smaller the p-value, the more unlikely it would have been to find the same number of proteins from the considered pathway, when proteins are picked randomly in the reference list. When the p-value of a pathway is below a threshold, we can assert that this pathway is enriched, or equivalently, that the list of highlighted proteins from this pathway is "enriched". The likeliness is assessed for each pathway by the p-value of a statistical (hypergeometric distribution) test, corrected by the Benjamini-Hochberg procedure for multiple

testing. A typical threshold value is around 0.05, meaning that when the p-value of a pathway is below this threshold, there is less than 5% chance that the highlighted list of proteins would be enriched in proteins from this pathway by chance.

In general, pathway enrichment analysis helps to interpret the data in terms of the biological processes and pathways information that they contain.

We used the pathways recorded in the Reactome database [119]. Reactome defines a deep hierarchy of pathways, allowing an analysis of biological functions at different levels of granularity. Therefore, we searched for enriched pathways, i.e. for pathways whose proteins would be over-represented in the highlighted list, at each hierarchy level of the Reactome database, with respect to the reference list of 785 proteins targeted by the 966 tested molecules. Besides, we used binomial instead of hypergeometric statistical tests. Indeed, since several hit molecules can target the same proteins, we considered the binomial distribution that describes, similarly as hypergeometric distribution, the number of times an event occurs in a given number of trials, but with replacement. Then, we considered enriched pathways at a p-value threshold of 0.05.

Among the 30 Reactome pathways at the top level of the hierarchy, the enriched pathways were, by decreasing order of p-values: signal transduction, metabolism, neuronal system, and cell cycle. We report, in tables 5.2, 5.3, 5.4, and 5.5, the over-represented pathways at all hierarchical levels for each of the 4 top-level enriched pathways.

Hierarchical level	Reactome pathway	p-value
level 1	Signaling by GPCR	$1.78 * 10^{-20}$
	Signaling by Hedgehog	$2.469 * 10^{-2}$
	Signaling by Leptin	$2.508 * 10^{-2}$
	Signaling by ERBB4	$2.970 * 10^{-2}$
	MAPK family signaling cascades	$3.474 * 10^{-2}$
	Signaling by SCF-KIT	$4.095 * 10^{-2}$
	Signaling by ERBB2	$4.659 * 10^{-2}$
	Signaling by FGFR	$4.665 * 10^{-2}$
	Signaling by EGFR	$4.665 * 10^{-2}$
level 2	GPCR ligand binding	$3.422 * 10^{-25}$
	GPCR downstream signaling	$1.125 * 10^{-18}$
	Gastrin-CREB signalling pathway via PKC and MAPK	$3.663 * 10^{-9}$
	Regulation of KIT signaling	$1.201 * 10^{-2}$
	Hedgehog 'off' state	$2.467 * 10^{-2}$
	SHC1 events in ERBB2 signaling	$2.760 * 10^{-2}$
	SHC1 events in EGFR signaling	$2.760 * 10^{-2}$
	GRB2 events in ERBB2 signaling	$2.760 * 10^{-2}$
	GRB2 events in EGFRsignaling	$2.760 * 10^{-2}$
	RAF/MAP kinase cascade	$2.760 * 10^{-2}$

	SHC1 events in ERBB4 signaling	$2.760 * 10^{-2}$
	MAPK1/MAPK3 signaling	$2.861 * 10^{-2}$
	Downstream signal transduction	$4.259 * 10^{-2}$
	Signaling by FGFR4	$4.659 * 10^{-2}$
	Signaling by FGFR4	$4.665 * 10^{-2}$
	Signaling by FGFR1	$4.665 * 10^{-2}$
	Signaling by FGFR3	$4.665 * 10^{-2}$
	Signaling by FGFR2	$4.665 * 10^{-2}$
level 3	Class A/1 (Rhodopsin-like receptors)	$3.313 * 10^{-27}$
	G alpha(z) signalling events	$1.531 * 10^{-10}$
	alpha(z) signalling events	$5.08 * 10^{-7}$
	RHO GTPases activate IQGAPs	$9.115 * 10^{-4}$
	VEGFR2 mediated cell proliferation	$3.595 * 10^{-2}$
	RAF/MAP kinase cascade	$2.760 * 10^{-2}$
	Signalling to ERKs	$3.474 * 10^{-2}$
	Inactivation, recovery and regulation of the phototransduction cascade	$4.659 * 10^{-2}$
	Downstream signaling of activated FGFR3	$4.665 * 10^{-2}$
	Downstream signaling of activated FGFR2	$4.665 * 10^{-2}$
	Downstream signaling of activated FGFR1	$4.665 * 10^{-2}$
	Downstream signaling of activated FGFR4	$4.665 * 10^{-2}$
level 4	Amine ligand-binding receptors	$9.660 * 10^{-54}$
	PI3K Cascade	$6.517 * 10^{-4}$
	Peptide ligand-binding receptors	$2.143 * 10^{-2}$
	FRS-mediated FGFR1 signaling	$2.760 * 10^{-2}$
	FRS-mediated FGFR1 signaling	$2.760 * 10^{-2}$
	FRS-mediated FGFR2 signaling	$2.760 * 10^{-2}$
	FRS-mediated FGFR2 signaling	$2.760 * 10^{-2}$
	RAF/MAP kinase cascade	$2.760 * 10^{-2}$
	RAF/MAP kinase cascade	$2.861 * 10^{-2}$
	RAF/MAP kinase cascade	$2.861 * 10^{-2}$
	Signalling to RAS	$3.474 * 10^{-2}$
level 5	Dopamine receptors	$8.560 * 10^{-21}$
	Serotonin receptors	$4.919 * 10^{-12}$
	Adrenoceptors	$4.303 * 10^{-10}$
	Muscarinic acetylcholine receptors	$1.489 * 10^{-8}$
	PI3K Cascade	$6.517 * 10^{-4}$
	RAF/MAP kinase cascade	$2.760 * 10^{-2}$
	PKB-mediated events	$2.819 * 10^{-2}$
	ARMS-mediated activation	$2.861 * 10^{-2}$

	Frs2-mediated activation	$2.861 * 10^{-2}$
level 6	RAF/MAP kinase cascade	$2.760 * 10^{-2}$
	PKB-mediated events	$2.819 * 10^{-2}$

Table 5.2: Over-represented Reactome pathways in the 80 hits, associated with the signal transduction pathway.

Hierarchical level	Reactome pathway	p-value
level 1	Metabolism of lipids and lipoproteins	$3.550 * 10^{-4}$
	Biological oxidations	$1.764 * 10^{-3}$
	Inositol phosphate metabolism	$9.794 * 10^{-4}$
level 2	Synthesis of IP3 and IP4 in the cytosol	$9.794 * 10^{-3}$
	Phase 1-Functionalization of compounds	$2.861 * 10^{-2}$
	Purine metabolism	$4.259 * 10^{-2}$
level 3	Acetylcholine regulates insulin secretion	$9.794 * 10^{-3}$
	Cytochrome P450-arranged by substrate type	$4.659 * 10^{-2}$

Table 5.3. Over-represented Reactome pathways in the 80 hits, associated with the metabolism pathway.

Hierarchical level	Reactome pathway	p-value
level 1	Potassium Channels	$3.945 * 10^{-4}$
level 3	Acetylcholine Binding And Downstream Events	$4.25 * 10^{-2}$
	GABA receptor activation	$4.659 * 10^{-2}$
level 4	Activation of Nicotinic Acetylcholine Receptors	$4.259 * 10^{-2}$
level 5	CREB phosphorylation through the activation of CaMKK	$3.021 * 10^{-2}$
	Postsynaptic nicotinic acetylcholine receptors	$4.259 * 10^{-2}$
level 6	Activation of CaMK_IV	$3.745 * 10^{-3}$

Table 5.4. Over-represented Reactome pathways in the 80 hits, associated with the neuronal system pathway.

Hierarchical level	Reactome pathway	p-value
level 1	Cell Cycle, Mitotic	$7.493 * 10^{-3}$
level 2	Mitotic G2-G2/M phases	$1.561 * 10^{-2}$
	M Phase	$4.659 * 10^{-2}$
level 3	G0 and Early G1	$1.323 * 10^{-2}$
	G2/M Transition	$1.561 * 10^{-2}$
	Mitotic Metaphase and Anaphase	$2.861 * 10^{-2}$
level 4	Centrosome maturation	$2.861 * 10^{-2}$
	Mitotic Anaphase	$2.861 * 10^{-2}$
level 5	Separation of Sister Chromatids	$2.861 * 10^{-2}$
	Recruitment of mitotic centrosome proteins and complexes	$2.861 * 10^{-2}$

Table 5.5. Over-represented Reactome pathways in the 80 hits, associated with the cell cycle pathway.

We observed a small number of enriched pathways at lower levels of the hierarchy. This facilitated analysis, but most of all, it means that only a limited number of pathways is intended to govern TNBC cell lines sensitivity to drugs, based on our data.

Three of the best enriched pathways are notoriously associated to cancer in general: "signal transduction", "metabolism" and "cell cycle". Indeed, TNBC tumours are expected to share some biological characteristics which are shared by most tumours. The "signal transduction" key word covers a large variety of sub-pathways whose function is to induce a cellular response (usually in terms of expression of specific genes) to an extracellular signal such as a growth factor or a cytokine. For example, FGFR signalling pathways are well known targets for various cancer types (for a review, see [244]), and in our results, we found several FGFRs signalling pathways to be enriched at levels 1, 2, 3, and 4. Similarly, the RAF/MAPK pathways and signalling cascades are known targets for cancer [245], and were found enriched at levels 1, 2, 3, 5 and 6. Other pathways are somewhat intriguing, such as the dopamine receptors, serotonine receptors, adrenoreceptors or muscarinic acetylcholine receptors pathways, but we will come back to these in the following.

The "metabolism" pathways are also clearly linked to cancer in many ways, as recently reviewed in [246], as well as "cell-cycle" [247].

We will not further discuss the results from these three top-level pathways, because they are not specific to TNBC. Their presence simply means that TNBC do share some common biological features with other cancers (except for the intriguing pathways mentioned above, to which we will return), which is expected, and that TNBC can be sensitive to some of the classical cytotoxic drugs used in many types of cancers. However, this does not correspond to our goal in the present project, which was to identify specific targets for TNBC, opening ways for specific (not cytotoxic-based) therapies.

Among the four most enriched pathways, the "neuronal system" pathways was unexpected in the context of TNBC, and attracted our attention. In particular, level 1 of "neuronal system" is enriched with "potassium channels", and the active molecules whose known targets contributed to enrichment of this pathway

are: Chlorepromazine, Pimozide, and Thioridazine. Indeed, these three molecules share a common target: the potassium voltage-gated channel subfamily H member 2, KCNH2, also called Kv11.1, ERG1, HERG, or EAG1 (among many other names). In addition, six other active molecules contributed to this pathway, based on high probability prediction of targeting the KCNH2 channel: Fluphenazine, Trifluoperazine, Fluvastatin, Paroxetine, Nebivolol, and Perphenazine. These molecules are used as antipsychotic drugs (Perphenazine, Pimozide, Thioridazine, Fluphenazine, Trifluoperazine), antilipemic (Fluvastatin), anti-depressant (Paroxetine) or anti-hypertension drugs (Nebivolol). To evaluate the interest of KCNH2 as a drug target for TNBC, as suggested by the above results, we undertook extensive bibliographic search. In the following, we present how potassium channels play a role in cancer, and how they can relate the above drugs to their effect on TNBC cell lines.

In living cells, transmembrane ionic gradients determine membrane excitability, which regulates important cellular events including generation and transmission of neuronal electrical signals [248]. However, ion channels are also associated with cellular migration and proliferation [248]. In particular, potassium (K+) channels can control the switch from epithelial to mesenchymal phenotype (called epithelial–mesenchymal transition; EMT), leading to enhanced migratory and invasive capabilities in cancer. Among them, KCNH2 is abundantly expressed in various human cancers, and uncontrolled gain or loss of activity of its encoded protein Kv11.1 is often linked with tumour initiation and progression.

Regarding TNBC, Breuer et al. [249] showed that activation of Kv11.1 by NS1643 (a molecule known to activate this channel) inhibits tumour growth in mice, after injection of MDA-MDB-231 TNBC cells, and that NS1643 suppresses the EMT transition in this TNBC cell line. They conclude that Kv11.1 activators could be relevant therapeutic targets for TNBC. Dookeran et al. [250] showed that 2-pores potassium channels (encoded by the KCNK genes) had distinct expression patterns in TNBC with respect to hormonal-dependant breast cancers, and pointed that these channels could be novel biomarkers for clinical diagnosis and therapeutic targets for TNBC. Panaccione et al. [251] showed that basal-like breast carcinoma, a subtype of TNBC, is driven by SOX10+/CD133+ cells, that express neural stem cell-specific markers and share molecular similarities with cancer stem cells of neural crest origin. These specific markers included potassium calcium-activated channel subfamily N member 4 (KCNN4). As a last example, Zhang et al. [252] showed that intermediate-conductance calcium-activated potassium (SK4) channels are involved in the proliferation, apoptosis, migration and EMT processes of TNBC cells.

It is not possible to review here all relevant literature, but overall, there is a significant amount of recent evidence that potassium channels play a key role in breast cancer progression and dissemination (tumours in which these channels are under-expressed or under-activated tend to be more aggressive), and that they could be used as biomarkers for prognosis. In particular, our results, together with other studies, indicate that small molecule activators of the Kv11.1 channel represent a novel therapeutic strategy to inhibit tumour growth and metastasis in TNBC. In this context, the fact that FDA-approved drugs used in clinic as antipsychotic also activate this channel would considerably accelerate their development against TNBC.

At this point, we would like to go back to the above mentioned point, that a few unexpected transduction pathways were found enriched in proteins that are targeted by the 80 hit molecules, namely dopamine receptors, serotonine receptors, adrenoreceptors or muscarinic acetylcholine receptors pathways. Analysis of the molecules targets which led to the enrichment of these pathways showed that, in fact, the active antipsychotic drugs discussed above, all known or predicted agonists of potassium channels, do also target these receptors. This explains the presence of the corresponding pathways enriched in the "transduction pathways". Interestingly, the DrugBank database indicates that the mechanisms of action of Thioridazine and Chloropromazine, as antipsychotic drugs, depend on their interaction with the above receptors (as for Pimozide, Fluphenazine and Trifluoroperazine), and that its ability to bind to the product of KCNH2 is stated as of "unknown pharmacological action". Therefore, their mechanism of action in psychotic diseases would be different from that in TNBC. This means that drugs repositioning would, in this case, involve proteins considered as "off-targets" with respect to their primary indication.

To conclude this discussion, we would like to underline that, even if we found many references that already raised the hypothesis of potassium channels as targets for TNBC, we did not use this information in the present study. In addition, although the biological assays identified a few hits molecules known to target potassium channels, our chemogenomics predictions allowed to raise the "signal" related to KCNH2 by predicting it as a target for other active molecules. In other words, chemogenomics allowed to analyse results of the biological tests in a comprehensive manner. It also allowed to attract our attention on papers that formulated the same hypothesis, based on totally independent studies. This reinforces the interest of pursuing the idea of potassium channel pathways as targets for TNBC, which to our knowledge, has not been exploited yet in clinical research.

Many more drugs are reported at DrugBank to bind to KCNH2 gene product than those tested in the present project. Interestingly, most of these drugs are labelled as "unknown pharmacological action" with regards to this target, which opens many opportunities for future research focused on these drug candidates in TNBC.

Characterisation of hits mechanism of action via drug response assays

In the previous section, the goal was to identify critical pathways associated with TNBC, based on the pathways that are enriched in proteins that are known or predicted to be targeted by the 80 hit molecules. Another way to identify the key pathways is to analysed which pathways of TNBC cells are experimentally affected (up- or down- regulated) upon exposure to drugs.

Our biologist collaborators have planed to perform such studies in the future. The idea was to identify differentially expressed pathways upon exposure the hit molecules. However, it was totally out of reach to perform such experiments on the 80 hits, and at this stage of the project, there were no argument to tell which were the "best" promising hits, on which we could choose to perform transcriptomic experiments.

Therefore, we chose to work on publicly available large databases that gather the results of transcriptomic experiments performed on various cancer cell lines upon exposure to many drugs. The goal was to identify pathways whose expression correlate with response upon exposure to a given drug (sensitive or resistant).

Even if these databases did not consider any of our 80 hits, key pathways involved in response of TNBC to other drugs could provide independent experimental data to evaluate the candidate pathways identified by our in silico analysis. In particular, if TNBC were found sensitive to other drugs whose response can be predicted based on some of the pathways we proposed above, this would increase their interest in terms of biomarkers and therapeutic targets.

5.2.3 Identification of cancer biomarkers and drug mechanism of actions via drug response assays

Several public databases for drug-cell line sensitivity are available. Some of the most popular are the Cancer Cell Line Encyclopedia [253] (CCLE) and the Genomics of Drug Sensitivity in Cancer [254] (CGP).

Unfortunately, these databases are not consistent. Indeed, Haibe-Kains et al. [255] reported poor correlations between their biological data, meaning that the same cell lines can have strongly different drug-response curves for the same drugs in different databases. Indeed, it is particularly difficult to standardise the experimental protocols and analysis methodologies, and it is also hard to compare databases due to importance of the "batch effects" that sometimes appear as the dominant "signal". These "batch effects" are, for example, the number of cells seeded per well, the drug concentration range that was examined, analytical tools to calculate sensitivity values and many other experimental details.

Others have reported substantial agreement between the CCLE and the CGP dataset [256]. In particular, they observed a Pearson correlation larger than 0.5 for 67% of comparable compounds. From our point of view, these observations do not fully meet the concerns about data reliability.

Materials

Despite this significant issue, we chose to focus our study on the CGP dataset published by the Sanger institute because it is a very rich and well maintained database, which provides clear and rigorous information about the files that are provided. This also avoided the reproducibility problem of using several databases.

This database comprises 957 cell lines from 55 cancer types, including 11 over the 12 TNBC cell lines considered in our project. The CGP database comprises 250 molecules for which we recovered their known targets from the Drugbank and Kegg databases, in addition to those recorded in the database itself. We also predicted proteome-wide targets with the approach presented in chapter 4 [50].

Among the available data, we kept only those for which the area under the drug-response curve was available, and for which we could retrieve the molecular structure and external ID identifier. Indeed, the area under the drug-response curve is the better quantitative measure of drug sensitivity than IC50, in particular because it is more consistent across databases [257]. This lead to a dataset comprising 95 compounds and 608 cell lines.

Methods: data types and encoding

To identify biomarkers of drug-cell line sensitivity, we performed a systematic analysis of a wide range of data types (gene expression, gene mutation status, gene copy number variation) and featurization (how the different data types are encoded).

Data encoding is a core component of any machine-learning, since the prediction performances highly depend on the features that describe the data. Moreover, in the present study, we need to focus on feature selection, because our primary goal is to identify biomarkers or therapeutic targets, i.e. identify features that are the most related to drug-cell line sensitivity, in order to both achieve high prediction scores and provide interpretable results. Therefore, we also tested to describe the data with information about gene sets or pathways instead of individual genes, since it results in much smaller feature vectors.

In the following, we enumerate the different types of molecular descriptors that we considered and give their corresponding number of features in parentheses.

We considered (i) binary structural fingerprint (2901), (ii) the probability of targeting each protein in the human proteome (16724), (iii) the binary profile of targeted proteins. A protein was considered as "targeted" if its probability is higher than a threshold. We varied this threshold between 0.7 and 1 (corresponding respectively to 127 and 278 features, i.e. 127 and 278 proteins targeted by at least one molecule). We also featurized molecules with (iv) the probability of targeting Reactome pathways (674), where the probability of targeting a pathway is equal to the maximum probability of targeting one protein in this pathway.

The last type of features allow describing molecules based on more global mechanisms of action than protein targets, and is easier to relate to biomarkers than structural chemical features. Note also that such description would have been very sparse without performing DTI prediction. Indeed, most molecules of this dataset had only one or two known targets, as in the case of the 966 molecules (see previous section), although such drug-like molecules are expected to interact with more proteins (see section 1.2).

For cell line descriptors, we considered: (i) gene expression at the mRNA level (17419), (ii) Reactome pathways expression (674), where a pathway expression is defined by the mean of expression of the genes belonging to this pathway. Also, we considered (iii) gene mutation binary status (19100) and (iv) gene pathways mutation binary status (674), where a gene pathway is defined as mutated if at least one of its genes set is mutated. Finally, we considered (v) gene copy number variation -CN- (45149), or alternatively (vi) "discretised CN" (45149), which takes values in -1,0,1,2. In particular, "discretised CN" is set to -1 if CN is less than 2, 0 if CN is 2, 1 is CN is between 3 and 6, 2 if CN is greater than 6. Therefore, "discretised CN" is coarsening the information in copy number variation which may make the model easier to train.

Besides, describing cell lines with features at the pathway expression level rather that an the gene expression level, or gene pathway mutation level instead of the gene mutation level, may make the model easier to train and to interpret.

All descriptors and outputs were standardised (cf appendix A.1).

Methods: feature pre-selection

For each considered type of features (both for molecules and cell lines), we tested various feature pre-selection methods, with the hope to build sparser and more interpretable models.

We pre-selected features based on (1) internal variability, that is to say, the features whose variance was higher than a threshold. This threshold is defined by the inflexion point in the curve displaying variance of the considered features when these features are sorted by decreasing order of variance.

Alternatively, we pre-selected (2) cancer-related features, that is to say, features corresponding only to the 609 cancer-related genes referenced in the Sanger GeneCensus file. This means that only proteins encoded by these genes were kept in the target profiles encoding molecules, and only expression, mutation or CN of these genes were kept to encode cell lines. This corresponds to a knowledge-based pre-selection method, with the underlying idea that we expect stronger and more relevant signal to arise from genes known to be involved in cancer. Note however that this might filter out genes that would be specifically involved in TNBC, and not more generally in cancer, or genes not yet known to be involved in cancer.

Also, we pre-selected (3) features which were statistically correlated with the output, via an F-test with p-values below 5% and corrected for multi-testing by Bonferonni method.

Alternatively, we pre-selected (4) features whose correlation with the output is higher than a threshold. As above, this threshold is defined by the inflexion point in the curve displaying correlation of the considered features when these features are sorted by decreasing order of correlation.

Finally, we considered to pre-select (5) the most "dependent" features, that is to say, the features whose mutual information [258] with the output is higher than a threshold. Again, this threshold is defined by by the inflexion point in the curve displaying mutual information of the considered features when these features are sorted by decreasing order of mutual information. Mutual information and correlation are similar notions, but using mutual information enables to consider non-linear relationships between features and outputs.

To summarise, the idea was to examine feature pre-selection methods from different perspectives: internal variability, cancer relatedness, correlation and mutual information with the output (i.e. drug-cell line sensitivity).

Methods: prediction of cell-line sensitivity to drugs

We tested two predictors, ElasticNet and RandomLasso, which themselves also perform feature selection while fitting the data. These methods were ran with or without pre-selection of features.

ElasticNet [259, 260] refers to the linear regression with combined L1 and L2 norms as regularisers (cf. appendix A.3). ElasticNet has two hyperparameters to control the amount of L1 and L2 regularisation. The L2 regularisation is intended to overcome the main limitation of the lasso, which is to select randomly one feature over a set of highly correlated features, by assigning equal weights to correlated features [176].

RandomLasso [261] is a procedure based on the lasso estimator. It has been shown to make the lasso internal feature selection more stable. Indeed, RandomLasso refers to the repeat of the following procedure: (1) bootstrap data (drawn with replacement n samples among the N samples in the training set). (2) scale the features with random scaling weights drawn from independent trials of a fair Bernoulli random variable of

parameter s. (3) fit a lasso model on the dataset resulting from the bootstrap and feature scaling steps. After repeating r times this procedure, each feature is assigned to a value of importance, which is the frequency at which it has been selected. A threshold t of importance is then used to select the top-ranked features that are finally used as input of a Ridge regression to perform the final prediction. RandomLasso has five hyperparameters: n, s, r, t and λ the regularisation parameter of the lasso. Importantly, steps 1 and 2 aim at subsampling training data sets with different statistics.

We determined the optimal hyperparameters by nested cross-validation on the considered dataset.

Methods: evaluation procedure

We evaluated the performance of the predictors via 5-fold cross-validation with the mean Pearson correlation between the predictions and the known drug response. Considering this metric instead of classic MSE is relevant here, since we want to recover the drug-cell line sensitivity profiles more than the exact values of this sensitivity.

Most importantly, we searched the triplet (features types, pre-selection method -including no pre-selection-, prediction model) leading to the best performance in drug sensitivity prediction, while selecting as few features as possible, for sake of biological interpretability. Indeed, in the context of a future clinical application, it would be difficult to interpret a large number of features in terms of biomarkers or drug targets. This explained why we considered sparse predictors (ElasticNet and RandomLasso), and features pre-selection methods.

Results and discussion: identification of the best cell-sensitivity prediction model

First, we tested the prediction approaches in the singletask framework. We either predicted the sensitivities of all cell lines for each drug (i.e. we predicted the cell lines sensitivity profiles of drugs, called prediction of drug profile in the following), or the sensitivity of each cell line to all the drugs (i.e. we predict the drug sensitivity profiles of each cell line, called prediction of cell line profiles in the following).

In a nutshell, for these two singletask problems, and for all types of features, RandomLasso was found more potent than ElasticNet, both in terms of predicting performance and number of selected features. For example, in the case of predicting drug profiles with mRNA expression features for cell lines, ElasticNet reached a Pearson correlation of 0.45 with 500 features, whereas RandomLasso reaches the same performance but with ten times fewer features. The same behaviour was observed for predicting cell lines profiles with structural descriptors for drugs. Notably, in this case, we observed that RandomLasso reaches the best performance for a small number of features in the order of a dozen, which makes RandomLasso suitable for biomarker identification.

For the problem of cell-line prediction profile (which is our problem in the present TNBC project), the most efficient encoding for molecules was the structural descriptors (calculated by the RDkit python package), resulting in an average Pearson correlation of 0.5 for around 30 selected features with the RandomLasso procedure. The second best molecular encoding is the probability to target the Reactome pathways, which

reaches a Pearson correlation of 0.36 for around 16 selected features. Recall that this performance was reached by using drug-target predictions. In addition, these features allow biological interpretation of the selected features in terms of mechanism of action, and therefore in terms of identification of new bio-markers or therapeutic targets, which is not the case of structural descriptors.

For the problem of predicting drug profiles, the best prediction results were obtained with mRNA expression features for cell lines, reaching a Pearson Correlation of 0.45 for around 50 features. This number of features is too high to identify a biomarker, even if the prediction performance reached are comparatively good. However, they could be used as signatures for drugs. In other words, for each drug, cell-line sensitivity could be predicted based on the expression of around 50 genes (the list of 50 genes depends on the cell line). Note that a similar number of genes have been proposed as gene signature for the relapse risk in breast cancer (in general, not only for TNBC) [262]. Overall, our prediction performance in the problem of predicting drug profile reached the state-of-the-art obtained on a similar dataset by other authors in terms of Pearson correlation, although they used an ElasticNet [263]. Encoding of cell lines with copy number variation and binary mutation status descriptors lead to models with lower prediction performance, since they reached a Pearson Correlation scores of 0.13 with about 30 features. This lower performance is somewhat expected in the present problem, since copy numbers are features that are "further" from the mechanism of drug-target interaction, than mRNA data.

In addition to the choice of features, feature pre-selection had a significant impact on the performance. In particular, when predicting cell line profiles with structural features for drugs, or with features based on the probability of targeting Reactome pathways, feature pre-selection based on feature-output correlation allowed the RandomLasso to both reach significantly higher performance and select fewer features. This is of particular interest in the TNBC project, since our main purpose is to build prediction models that are both predictive and biologically interpretable.

Second, we also considered a multitask approach to the problem of predicting cell lines sensitivity to drugs. This consists in predicting the drug and cell line profiles simultaneously, i.e. to model drug response for all drugs and all cell lines at the same time. In this approach, we expect that sharing information between similar drugs and similar cell lines may improve prediction performance, as observed earlier in this manuscript, in the context of chemogenomics, where information is shared between molecules and proteins. In addition, an efficient multitask approach would allow the prediction of sensitivity for new drugs (i.e. in the present case orphan drugs in terms of cell-line sensitivity information), which is of great practical interest.

We first tested "cluster-MT", which consists of clustering tasks (tasks are drugs in the problem of predicting drug profiles, or cell lines in the problem of predicting cell line profile) based on their features before running MT-RandomLasso (or MT-ElasticNet) on each of the clusters. The choice of running MT-RandomLasso per clusters was guided by the idea to share information only between close tasks. We observed that the cluster-MT approach lead to identical performances as the singletask approaches, but selected significantly more features than the best singletask approaches, which leads to less interpretable models.

We also considered to use task-specific information in two ways. First, we considered to describe a (drug, cell line) pair by "$x_{mol_i,cell_j} = x_{mol_i} + x_{prot_j} + kron(x_{mol_i}, x_{prot_j})$", where x_{mol_i} refers to the descriptor

vector of molecule i, x_{prot_j} refers to the descriptor vector of cell line j and "kron" refers to the Kronecker product between two vectors. However, such a multitask approach was impracticable in our case because of the dimension of the feature vectors, even after feature pre-selection. Second, we considered to describe a (drug, cell line) pair by the Kronecker product of the molecule and cell line kernels (RBF kernel based on the above mentioned molecule and cell line descriptions). However, such an approach does not allow to perform feature selection.

Overall, we conclude that, in the context of bio-marker or target identification and for the problem of predicting TNBC drug profiles based on drug sensitivity assays with linear models, we recommend to use singletask approaches (one task is one cell line) with RandomLasso and feature pre-selection (although the pre-selection method leading to the best results depends on the features used to encode molecules).

As perspectives regarding approaches for drug-cell line sensitivity prediction, we steer the interested reader to the following work [263] which proposed a relevant two stages linear approach to identify biomarkers based on drug response data from multiple data sources, and to [264, 265], which defined a non-linear approach for biomarkers identification.

Indeed, we can reasonably expect that drug response relies on multiple pathways or multiple mechanisms that may involve the interaction of multiple targets such that linear methods do not have the capacity to reveal clearly these interactions. Note that some have recently proposed efficient sparse linear models taking into account two-way interactions of features in high dimensional space [266].

Results and discussion: identification of TNBC drug-sensitivity biomarkers

We now go back to our initial question, i.e. the identification of bio-markers or therapeutic targets for TNBC. First, recall that the CGP database comprises 11 of the 12 TNBC cell lines that we initially considered in our project, and therefore, spans enough diversity in TNBC cells to be relevant in our project.

Based on the results presented in the previous section, we predicted the cell line profile via a RandomLasso with drugs encoded by the probability to target Reactome pathways, with and without feature pre-selection based on statistical correlation with the output. Indeed, as discussed above, for cell line profile prediction, these drug features led to the second best prediction score, and allowed biological interpretation of the selected features. We recall that this encoding of molecules was enabled thanks to proteome-wide DTI interaction prediction.

The pathways that are mostly selected by the RandomLasso on average for the TNBC cell lines, with and without feature pre-selection, are given respectively in table 5.7 and table 5.6.

More precisely, table 5.7 and table 5.6 provide, by decreasing order, the pathways that the RandomLasso most frequently selected, on average over the 11 TNBC cell lines contained in the CGP dataset. In other words, the top classified pathways are those who mainly contribute to the prediction of TNBC cell lines drug sensitivity. We will not analyse all the information contained in Table 5.7 and Table 5.6, but we will focus

of the results that can be related to those obtained with the 80 hits from the Curie project.

Reactome Pathways	score
Iron uptake and transport	0.45359
Cell cycle	0.43277
Sphingolipid metabolism	0.4230
Mitotic prometaphase	0.39695
Cell cycle mitotic	0.36613
Glycosphingolipid metabolism	0.33881
Latent infection of homo sapiens with mycobacterium tuberculosis	0.33745
Tryptophan catabolism	0.33523
Chylomicron mediated lipid transport	0.33372
RIG_I MDA5 mediated induction of IFN alpha beta pathways	0.29468
Termination of O_glycan biosynthesis	0.29040
Chemokine receptors bind chemokines	0.28036
Cytochrome P450 arranged by substrate type	0.27795
O_linked glycosylaton of Mucins	0.27790
Endogeneous sterols	0.276727
DNA repair	0.27440
Tandem pore domain potassium channels	0.270181
Transport of organic anions	0.26777
Peptide ligand binding receptors	0.266136
Initial triggering of complement	0.263045
Activted TAK1 mediates P38 MAPK activation	0.25754
Class I MHC mediated antigen processing presentation	0.2575
MTORC1 mediated signalling	0.25668

Table 5.6. Reactome pathways most frequently selected by the RandomLasso, averaged over the 11 TNBC cell lines, without feature pre-selection.

Reactome Pathways	score
DNA repair	0.8020
Chylomicron mediated lipid transport	0.78386
P38MAPK events	0.78363
Glycoprotein hormones	0.67627
Defensins	0.64172
Digestion of dietary carbonhydrate	0.64090
Purine ribonucleoside monophosphate biosynthesis	0.61881
Norepinephrine neurotransmitteur release cycle	0.60877
Mitochondrial protein import	0.59063
Inwardly rectifying K_channels	0.56104

Presynaptic nicotinic acetylcholine receptors	0.5595
Collagen formation	0.55831
O_linked glycosylation of Mucins	0.55695
IL_3_5 AND GM_CSF signalling	0.54568
Complement cascade	0.53986
Hemostasis	0.51545
Signalling by NGF	0.49672
Termination of O_glycan biosynthesis	0.49281
Formation of ATP by chemiosmotic coupling	0.49204
NCAM1 interactions	0.47813
ADP signalling through P2RY12	0.46590
Citric acid cycle, TCA cycle	0.4440
Amine derived hormones	0.43186
Transport to the Golgi and subsequent modification	0.42586
Cell-cell communication	0.42568
Signalling by ERBB4	0.41709
Glucuronidation	0.4130
Respiratory electron transport	0.39754
Nucleotide like Purinergic receptors	0.39731
Transmission across chemical synapses	0.39440
Neurotransmitteur receptor binding and downstream transmission in the postsynaptic cell	0.39263
Ionotropic activity of Kainate receptors	0.39036
Androgen biosynthesis	0.38836
Endosomal sorting complex required for transport ESCRT	0.36954
Signalling by SCF_KIT	0.33795
Transport of mature transcript to cytoplasm	0.3260
TRAF6 mediated IRF7 activation	0.31836
Interactions of VPR with host cellular proteins	0.31759
Prostacyclin signalling through prostacyclin receptor	0.31368
ER phagosome pathway	0.31336
Purine catabolism	0.31295
Cytokine signalling in immune system	0.30504
Activation of IRF3_IRF7 mediated by TBK1 IKK EPSILON	0.30381
Negative regulators of RIG_I_MDA5 signalling	0.29954
Respiratory electron transport, ATP synthesis by chemiosmotic coupling and heat production by uncoupling proteins	0.29927
IRAK2 mediated activation of TAK1 complex upon TLR7 8 OR 9 stimulation	0.29722
Transport of mature mRNA derived from an intronless transcript	0.29654
Class I MHC mediated antigen processing presentation	0.29540

TRAF3 dependent IRF activation pathway	0.29204
Deadenylation of mRNA	0.29004
IL_2 signalling	0.28295
Antigen processing cross presentation	0.28068
Base excision repair	0.27181
Regulation of RHEB GTPASE activity by AMPK	0.27168
Resolution of AP sites via the multiple nucleotide patch replacement pathway	0.26677
DNA strand elongation	0.26668
Base free sugar phosphatase removal via the single nucleotide replacement pathway	0.26104
Processive synthesis on the lagging strand	0.25954
Resolution OF AP sites via the single nucleotide replacement pathway	0.25618
Factors involved in megakaryocyte development and platelet production	0.25386
TRAF6 mediated induction of TAK1 complex	0.25304
Signaling by ILS	0.25231
Energy dependent rgulation of MTOR by LKB1 AMPK	0.25118

Table 5.7: Reactome pathways most frequently selected by the RandomLasso, averaged over the 11 TNBC cell lines, with feature pre-selection.

Before analysing the results shown in the two tables, let us make a few remarks about the molecules of the CGP dataset and of the Institut Curie dataset.

First, none of the molecules in the CGP dataset was common to the 966 molecules tested in Institut Curie by our collaborators. In addition, the CGP dataset contains mainly drugs that were studied in the context of cancer, while our collaborators chose to test many families of drugs, not necessarily used in cancer. Indeed, it is precisely on purpose because the idea was to find drugs with other primary indications (i.e. not toxic as many antitumoral drugs are), to be repositioned specifically in TNBC. Therefore, not only were the molecules different in the two datasets, but in addition, they are expected to target different families of proteins. Therefore, there is considerable bias between the two sets of molecules, both in terms of structure and known mechanisms of actions. Consequently, one can expect that cancer-related pathways (in general, not specifically in TNBC) might be selected prevalently by the RandomLasso, because the drugs in the CGP dataset target mainly known cancer-related proteins, as shown in table 5.8.

In most cases, the molecules in the CGP dataset target well-known cancer-related kinases and their associated pathways, such as the PI3K, JAK2, mTOR, RTK, MAPK, and other well known cancer pathways such as p53, mitosis, cytoskeleton, apoptosis, or DNA repair. Importantly, none of the CGP molecules had a potassium channel or a receptor involved in neuronal signal transduction as known target.

That being said, let us first analyse Table 5.6, which presents the top ranked pathways when no pre-selection is performed before the RandomLasso (i.e. 674 Reactome pathways are classified). Interestingly, the pathway "tandem pore domain potassium channels" is found at position 17 over 674, in a context where

none of the molecules had any potassium channel as known target. This means that this pathway appears because it contains proteins that were predicted as targets for some of the molecules. Therefore, we searched for which proteins of this pathway had the best prediction scores for the CGP molecules. For all molecules, the best predicted target in this pathway was one of the KCNK proteins. This is very interesting, since, as mentioned in the analysis of the 80 hits from Intitut Curie, breast cancer subtypes (including TNBC) are known to display specific expression patterns of the KCNK genes, which was suggested to be used as biological markers [250].

Feature pre-selection kept only 94 over 674 pathways. Among them, table 5.7 shows the pathways that are the most frequently selected by the RandomLasso. Notably, the "inwardly rectifying potassium channels" appear in position 10, which again, underlines the interest of potassium channels in TNBC. "Inwardly rectifying potassium channels" are encoded by the KNCJ family of genes, also named GIRK genes, and many references point at their role in some types of cancers. In particular, according to the Human Genome Atlas databases, KCNJ3 (or GIRK1) was found particularly abundant in breast cancer, and to a lesser extent, in renal cancer, but not in other cancer localisation. However, other KCNJ genes are also expressed in breast cancer, including TNBC [267], and over-expression of KCNJ3 was suggested to serve as a biomarker of poorer outcome in breast cancer, independent of the hormonal receptor status [268].

Taken together, and despite the fact that the data from CGP and from Institut Curie are only very loosely related (in addition to being totally independent datasets), we still could find some clues in the CGP data indicating the interest of potassium channels as protein targets for TNBC. We are aware that the results obtained with RandomLasso on the CGP data deserve to be analysed at length and that RandomLasso's selected pathways can not be taken at face value. But we want to highlight here that a large bundle of evidence from independent data and analysis converges towards the identification of the same therapeutic direction.

Conclusion

Overall, the analysis performed on the TNBC project led us to point at potassium channels as potential drug targets in TNBC, in agreement with other authors cited above. We would like to recall that such conclusions could not have been drawn without prediction of protein targets using our chemogenomic approach. The fact that consistent observations were obtained based on the data of Institut Curie, the CGP dataset, and other authors based on totally different studies and approaches, are strong arguments to be confident in this suggestion for future research directions in TNBC. More importantly, since marketed molecules targeting potassium channels are available, this offers opportunities for drug repositioning in TNBC, potentially in combination with classical chemotherapy.

Drug name	known mechanism of action
CP466722	Genome integrity
TG101348	other

JW-7-24-1	other
PIK-93	PI3K signaling
TPCA-1	other
ZSTK474	PI3K signaling
QL-X-138	other
XMD13-2	other
GSK2126458	PI3K signaling
OSI-027	TOR signaling
PHA-793887	cell cycle
NPK76-II-72-1	mitosis
KIN001-244	PI3K signaling
PI-103	PI3K signaling
UNC0638	chromatin histone methylation
KIN001-102	PI3K signaling
AT-7519	cell cycle
CAY10603	chromain histone acetylation
BMS345541	other
BIX02189	other
QL-XI-92	RTK signaling
BX-912	PI3K signaling
Foretinib	RTK signaling
Tubastatin	A chromain histone acetylation
I-BET-762	chromatin, other
NG-25	other
CUDC-101	chromain histone acetylation
QL-XII-47	other
YM201636	other
AR-42	chromain histone acetylation
Ispinesib Mesylate	mitosis
5-Fluorouracil	DNA replication
WZ3105	other
MPS-1-IN-1	mitosis
GSK1070916	mitosis
CX-5461	other
KIN001-236	other
TAK-715	JNK and p38 signaling
KIN001-260	other
BHG712	RTK signaling
CAL-101	PI3K signaling

XMD14-99	RTK signaling
KIN001-266	other
EKB-569	EGFR signaling
Masitinib	RTK signaling
LY317615	other
OSI-930	RTK signaling
Y-39983	cytoskeleton
XL-184	RTK signaling
AS605240	PI3K signaling
FR-180204	ERK MAPK signaling
GSK690693	PI3K signaling
STF-62247	other
YM155	apoptosis regulation
GSK429286A	cytoskeleton
Phenformin	metabolism
UNC0638	chromatin histone methylation
T0901317	other
NSC-207895	p53 pathway
VX-11e	ERK MAPK signaling
KIN001-055	other
AC220	RTK signaling
JQ1	chromatin, other
CH5424802	RTK signaling
PFI-1	chromatin, other
Linifanib	RTK signaling
Tivozanib	RTK signaling
BMS-708163	other
MP470	RTK signaling
BMS-536924	IGFR signaling
Ruxolitinib	other
GW-2580	RTK signaling
SN-38	DNA replication
CP724714	EGFR signaling
Zibotentan	other
EX-527	other
Tamoxifen	other
Piperlongumine	other
JNJ-26854165	p53 pathway
EHT 1864	cytoskeleton

AG-014699	Genome integrity
TW 37	apoptosis regulation
UNC1215	other
(5Z)-7-Oxozeaenol	other
CCT007093	other
SB 505124	other
PF-4708671	TOR signaling
CHIR-99021	WNT signaling
BMS-708163	other
selumetinib	ERK MAPK signaling
XAV939	WNT signaling
Bleomycin	DNA replication
Afatinib	EGFR signaling
IOX2	other
PLX4720	ERK MAPK signaling

Table 5.8: List of the considered molecules associated with their known mechanism of actions as recorded in the CGP database.

Part III

Drug virtual screening with end-to-end extracted representations for molecules and proteins

Chapter 6

End-to-end encoding of molecular graphs and protein sequences

Abstract: *Along with the rise of deep learning approaches, recent studies have focused on designing models to automatically learn, in an end-to-end fashion, numerical representations of molecules and proteins. This is intended to make the feature extraction more flexible and more data specific. In this chapter, we first discuss deep learning methods to extract data-driven descriptors on protein sequences and molecular SMILES strings. As molecules form naturally undirected graphs, we also examine data-driven feature encoders on such graphs. In particular, we offer a general framework for graph representation learning and survey the variety of dedicated available approaches.*

Résumé: *Parallèlement à la montée en puissance des approches d'apprentissage profond au cours de la dernière décennie, les dernières études sont principalement consacrées à la conception de modèles permettant d'apprendre automatiquement, de manière intégrale, les représentations numériques de molécules et de protéines. Cela a pour but de rendre la représentation des données plus flexible et plus spécifique aux données et à leurs labels. Dans ce chapitre, nous abordons d'abord les méthodes d'apprentissage profond pour extraire les descripteurs de données séquentiels tel que les séquences protéiques et les chaînes SMILES de molécules. Comme les molécules peuvent être vu naturellement comme des graphes non dirigés, nous examinons également les méthodes d'apprentissage de représentation sur de tels graphes. En particulier, nous proposons un cadre général pour l'apprentissage de la représentation sur graph et étudions la diversité des approches disponibles dédiées.*

All methods for drug-target interaction prediction previously discussed rely on "data-blinded" features or "data-blinded" similarity measures. The term "data-blinded" refers to the fact that the features or the similarity measures are either handcrafted or automatically extracted from the input data $\mathbb{P}(x)$ based on human expertise, but are not guided by the labelled data $\mathbb{P}(x, y)$ themselves.

For the majority of ML applications, machine learning prediction performance strongly relies on these extracted features, and on the ability of ML models to identify the relevant features and combine them for the prediction.

However, humans may fail at identifying these features. Indeed, such a process may not be sufficient in cases where humans fail themselves to execute the task or to identify the features of the problem. It is actually the case in many bioinformatics tasks. Therefore, recent advances in machine learning algorithms belong to the family of *representation learning* or *feature learning* methods.

The goal of representation learning [269] is to extract a representation of the labelled data, that is optimised to best solve a machine-learning task, rather than to rely on expert-based data-blinded features. In the case of supervised learning, the representation is learnt in an end-to-end manner, which refers to the joint optimisation of the feature vector and of the prediction model itself.

In particular, deep learning models, or equivalently "deep Artificial Neuron Networks" (ANN) [51, 270], are particularly suitable for end-to-end learning. They became increasingly popular in the past few years [190, 191, 192]. We discuss general aspects of representation learning via deep learning and deep neuron networks in appendix A.8.

In addition, within the representation learning framework, we name "data-driven features" the numerical descriptors that are extracted from the raw input data in an end-to-end fashion. In the specific case of chemogenomics, data-driven features refer to the descriptors that are extracted directly on the molecular graphs and protein sequences in an end-to-end manner (i.e. so that the learnable feature is optimised through the minimisation of the error of the supervised prediction task). However, models that consist of stacked fully connected neuron layers (see appendix A.8) perform representation learning on data-blinded features, so that we are to name these approaches as data-blinded.

In this chapter, we discuss how neuron network modules can extract numerical attributes on molecular graphs and protein sequences. The material reviewed in this chapter is thus essential to explore the use of data-driven features for drug virtual screening in the next chapters.

6.1 End-to-end encoding of sequences

In this section, we introduce data-driven feature extractors for chemoinformatics and proteomics. That is, how one may encode a learnable attribute vector from SMILES representation of molecular graphs (section 6.1.2) and protein sequences (section 6.1.1).

We begin this introduction by data-driven feature extraction on protein sequences.

6.1.1 End-to-end encoding of protein sequences

In this section, our goal is to numerically encode protein sequences in the context of proteome-wide drug-target interaction prediction. Indeed, the protein sequence is a description of the protein which is available over the entire human proteome.

Representation learning on protein sequence appears natural because of its sequential nature, which is particularly suited for dedicated neuron network architectures such as convolutional neuron networks and recurrent neuron networks (see appendix A.8.2).

When encoding a sequence of amino acids with neuron network-based encoders, we still require to define attributes for amino acids.

An obvious solution is to use the so-called one-hot encoding, where amino acids of the protein sequence are encoded by a binary vector of the length 20 (one bit for each natural amino acid), containing a single one and 19 zeros.

A second and richer encoding is to use a "sequence profile", encoding each amino acid by a mutation probability profile which is the position-dependent probability that the current amino acid get mutated in each other amino acids. Such mutation probability profiles are conventionally generated by running sequence alignment algorithm like PSI-BLAST [77] against a reference database. In particular, the use of mutation profiles was found to be helpful in the case of secondary structure prediction [271].

Note that we can also consider additional structural or physical amino acid properties to complete amino acid attribute vectors. However, this knowledge is not available for the overall proteome, and we should rely on available predictors to obtain this information for all proteins.

We now focus on the encoding of sequences of amino acids (i.e. protein sequences).

First, one can consider protein sequences as sentences of words [272] that can be encoded via natural language processing techniques such as word2vec [273]. Note that although Word2vec is an unsupervised method, it is trained using an auxiliary prediction task, which is either (i) the continuous bag-of-words: the task is to predict a word from the context, or (ii) the skip-gram : the task is to predict the context based on a word. More precisely, word2vec learns a small linear model for each word to solve the auxiliary prediction task, and the fitted weights of the word-specific models are the learnt representation of the words.

In the case of protein sequences, words are n-grams amino acids, each shifted by m amino acid (typically, n is three and m is one).

Another approach to encode proteins are specialised neuron network architectures like convolutional neuron networks (CNN) and recurrent neuron networks (RNN), in particular Long Short-Term Memory cells (LSTM) and bi-directional LSTM (bi-LSTM), have been successfully designed and trained on proteomic data to achieve state-of-the-art performance in various prediction problems [274, 275, 276, 277, 278, 279]. These neuron network architectures are introduced in appendix A.8.2.

In this perspective, the work of Jurtz at al. [279] is of significant interest as it demonstrates the interest of the CNN and LSTM neuron network architectures for representation learning on protein sequences.

We now discuss these neuron network architectures for learning a protein sequence representation.

First, one-dimensional convolutional layers can efficiently process protein sequences by detecting local patterns. Indeed, CNN filters provide a learnable bank of sequence motifs.

Second, bi-directional LSTM (or RNN in general) layers can integrate local information both forward and backwards in the sequence, to encode local and contextual information. The local information could be the amino acid type if the bi-LSTM is built on the original description of amino acids, or amino acid patterns if the bi-LSTM is built on a convolutional layer.

Moreover, when predicting protein-level properties, it is reasonable to assume that individual sub-parts of its input sequence may contain most of the predictive information. In this case, it can be beneficial to add an attention weighting mechanism, which is a learnable neuron module that is trained to pass the most relevant sub-parts of the input (see appendix A.8.2). Moreover, since an attention mechanism highlights elements with the most task-specific relevant information, it naturally helps to visualise and understand what part of the protein sequence leads to the decision of the model.

In practice, all these neuron modules (convolutions, bi-LSTM, attention mechanism) can be combined. For instance, in the case of the prediction of sub-cellular localisation, Jurtz at al. [279] showed that an attention mechanism on top of an encoder made of a convolutional layer followed by a bi-directional LSTM performs better than the same encoder without the attention mechanism, which in turn performs better than an encoder made of successive convolutional or LSTM layers.

Intuitively, the bi-LSTM can integrate the patterns detected by the convolutional layer locally, both forward and backwards. Then, the attention function focuses on identifying what parts of the sequence trigger the subcellular localisation, based on the contextual patterns learnt by the convolution-biLSTM layers.

To conclude this section, we introduce one of the earliest re-branding [280] of artificial neuron networks for protein property prediction based on protein sequence.

This study questioned the efficiency of multitask learning for several protein properties (secondary structures, physical properties and so on). They encoded protein sequences with a single standard RNN that is shared between tasks, and they used task-specific fully connected layers taking as input the same data-driven encoding of protein sequences, for prediction. They report that training the protein sequence encoder on multiple task was not efficient, except for the tasks with fewer data which benefited from multitask learning.

Interestingly, they created an "artificial task" to pre-train the protein sequence encoder, inspired by the field of natural language processing. This task consists in separating artificial proteins (generated by random juxtapositions of amino acids) and "real" proteins from the Swiss-Prot database. With this pre-training, the encoder learns to represent amino acid patterns in naturally occurring protein sequences, i.e. to encode amino acids into a semantically meaningful space. This strategy enhanced prediction performance.

6.1.2 End-to-end encoding of SMILES molecular representation

As proteins, molecular compounds can also be encoded into a one dimensional representation, like the SMILES or the InCHl representations introduced in Section 2.1.

In this case, each character of the SMILES (or InCHl) is typically one-hot encoded. That is to say, each character is represented by a vector full of zeros except a one at the position of the corresponding character corresponds to.

A first attempt to process SMILES is the work of Wan et al. [281, 282]. It is inspired from natural language processing (NLP) techniques. They proposed to adapt the word-embedding approach to DTI prediction. More precisely, they considered an unsupervised representation learning procedure named word2vec, introduced in 6.1.1, to encode molecular SMILES.

First, they derived compound substructures from the Morgan algorithm to define chemical "words", so that compounds are "sentences" of these chemical words [282]. "Words" feature vectors, learnt via the word2vec algorithm, can be simply summed up to obtain the associated "sentence" feature vector.

Thus, intuitively, this approach builds latent representations such that molecules with co-occurring functional groups are close in the latent space. Note that this has been visually checked [282]. Moreover, this method showed equally good or slightly better performance compared to the standard ECFPs and other representation learning-based approaches, on large-scale ChEMBL bio-activity data [281, 282]. This indicates the efficiency of the word-embedding approach, at least for the considered tasks.

From another perspective, SMILES2vec [283] and variants [284] considered a recurrent neuron network on the SMILES representation of molecules. Such naive approach shows the potential of deep learning in extracting relevant features, even without carefully constructed features and without hyper-parameter selection with Bayesian optimisation (see appendix A.8.4). Indeed, depending on the dataset, they outperform, or underperform, an FNN trained using data-blinded features [283, 284].

As already noticed in Section 2.1, one molecule can have various SMILES representations, since they depend both on the atom where the enumeration starts, and the path followed along the 2D graph of the molecular structure.

Although this is often mentioned as a drawback, it can be used as data augmentation. Indeed, multiples SMILES of a same molecule can be seen as several views of the same molecule. In particular, using various SMILES representation may reduce overfitting in the training step. During the testing step, SMILES augmentation can be used via an ensemble learning strategy by proposing a "majority voting scheme" (mean or max, for instance) based on the predictions associated with the different SMILES views of the same molecule. Kimber et al. [285] observed a synergistic effect of the multiplicity of SMILES views to improve data-driven feature-based models.

Finally, Dalke et al. [286] proposed an enhanced unequivocal SMILES syntax named DeepSMILES. DeepSMILES relies mainly on the modifications on the branch and ring encodings in the SMILES, with the hope to improve the performance of machine learning models that encode SMILES strings in an end-to-end

fashion.

Independently, Zang et al. [284] considered recurrent neuron networks directly on SMILES representations of molecules, but within a semi-supervised learning framework. To tackle the gap between the amount of available data and the amount of data usually required by deep learning networks, they proposed to train the RNN encoder on two tasks: one molecular biological property (supervised) and one auto-encoding (unsupervised) prediction task. This can also be viewed as a transfer learning approach. Indeed, the self-recovery task serves as an auxiliary task to augment the main supervised prediction task. In this model, the total loss allows a trade-off, controlled by λ, between the supervised and unsupervised tasks: $\mathcal{L}_{semi} = \mathcal{L}_{unsup} + \lambda\mathcal{L}_{sup}$.

Interestingly, they noticed that the performance is robust to changes in architectures and in range of λ. Their semi-supervised approach exhibited significantly higher performance on a solubility prediction task, than an approach based on a pioneering form of graph convolutional neuron networks by Duvenaud et al. [287] (we introduce this work in appendix C.3). However, the performance reached by the latter is surprisingly low, and we think this is due to the choice of the datasets whose size is unusually small.

Graph convolutional neuron networks are neuron network modules that directly extract descriptors on undirected graphs, such as molecular graphs.

Approaches based on data-driven features extracted on SMILES proved to be competitive with these graph-based representation learning models which directly process the molecular graph [288, 289]. However, molecular graph-based representation learning models, appear as a more valuable extension of method repertoire, because of their flexibility and because they offer, a priori, a larger room for improvements.

In the next section, we introduce such graph convolutional networks (GCN), and present the main recent developments dedicated to the improvement of their expressivity and flexibility.

6.2 End-to-end encoding of undirected graphs: applications to molecular graphs

As discussed in Section 2.1, molecules are usually represented via their molecular graphs.

A molecular graph is an undirected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (or nodes), and $\mathcal{E}$ the set of edges. Nodes correspond to atoms and edges correspond to covalent bonds. Thus, molecular graphs belong to the family of attributed undirected graph, since nodes are attributed with atom properties (like atom type category, but also any physicochemical and topological properties), and edges are attributed with bond properties (like bond type, but also topological properties, for instance).

In the following, we use the word "attribute" to describe the original feature vector of nodes and edges. We use the terms "embedding" or "representation" to refer to the feature vectors of nodes and edges that are extracted by GCNs. We note that each node i has an attribute vector $\mathbf{x}_i$ and each edge (i, j) has an attribute vector $\mathbf{x}_{ij}$. Regarding the learnt representation, we write $\mathbf{h}_i$ and $\mathbf{h}_{ij}$ the node and edge embeddings. $\mathcal{N}(i)$ refers to the neighbouring nodes of node i.

In this section, our goal is to defined learnable molecular graph encoders to be used for drug virtual screening in the next chapters. These encoders can extract data-driven features from the labelled data, so that we do not need to refer to the chemists' expertise to design handcrafted or calculated data-blinded features, as discussed in sections 2.1 and 2.2. Indeed, such learnable encoders of molecular graphs are trainable in an end-to-end manner, which means that the representation is extracted in order to optimise the prediction on a particular prediction task. Performance improvements can thus be expected from these approaches, compared to data-blinded approaches.

We provide a historical perspective on graph representation learning, in particular for chemoinformatics, in appendix C.

In this chapter in particular, we present a general framework for node and graph representation learning. To our knowledge, only few papers [290, 291] proposed a general overview for graph convolutional network (GCN), or simply graph neuron network (GNN). Among these, Hamilton et al. [292, 291] proposed a general algorithm for graph convolutional networks. The general framework of graph convolutional neuron network that we introduce in this section is greatly inspired from their work, but we also consider more recent developments from other studies to re-write this general algorithm.

6.2.1 Graph convolutional neuron networks

We present the general framework of graph convolutional neuron networks in Alg. 1.

Algorithm 1 Graph convolutional neuron network (GCN): at each iteration, each node aggregates information from its local neighbours. As this process iterates, nodes incrementally gain more information from further part of the graph while encoding local graph topology and distribution of attributes.

Require: (i) Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; (ii) input features $\mathbf{x}_v, \forall v \in \mathcal{V}$; (iii) depth K; (iv) learnable and differentiable aggregating functions $AGGREGATE^{(l)}_{graph}$ and $AGGREGATE^{(l)}_{node}$, $\forall l \in 1, \dots, L$; (v) neighbourhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$.

Goal: compute a graph-level representation: $\mathbf{m}$.

$\mathbf{h}_i^0 \leftarrow \mathbf{x}_i, \ \forall i \in \mathcal{V}$.

for $l = 1 \dots L$ **do**

  for $i \in \mathcal{V}$ **do**

    $\mathbf{h}_i^{(l)} \leftarrow AGGREGATE^{(l)}_{node}(\{\mathbf{h}_i^{(l-1)}\} \cup \{\mathbf{h}_j^{(l-1)}, \forall j \in \mathcal{N}(i)\})$

  end for

  $\mathbf{m}^{(l)} \leftarrow AGGREGATE^{(l)}_{graph}(\{\mathbf{h}_i^{(l)}, \ \forall v \in V\})$

end for

$\mathbf{m} \leftarrow COMBINE_{graph}(\{\mathbf{m}^{(l)}, \ \forall l\})$

Such an approach is spatial and convolutional because it iteratively updates a node representation with a function inputting its spatial neighbourhood, in a manner similar to the receptive field of a convolutional kernel in computer vision (see appendix A.8.2).

In other words, the node representation is iteratively updated based on the local nodes' representations. Therefore, the learnt representation simultaneously encodes the topological structure of each node's neighbourhood, as well as the distribution of node features in the neighbourhood. We expect the extracted structural properties to reveal both the node's local role in the graph, as well as its global role, thanks to the iteration of the update.

Note that it is also possible to consider, at each iteration, several neighbourhood sizes by simply duplicating and applying the $AGGREGATE^{(l)}_{node}$ function to several r-hop neighbourhood. The latter is defined as all nodes reachable by paths of length r [293].

Furthermore, this algorithm is applicable both in transductive (test data are observed during training) and inductive (test data are not observed during training) settings, since we can apply alg. 1 to new unseen nodes and graphs after it has been fitted on the training data.

Overall, the main drawback of Alg. 1 lies in its computational and memory cost caused by recursive expansion of neighbourhoods. To address this issue, graph convolutional networks have been interpreted as integral transforms of embedding functions under probability measures [294], that can be more cheaply computed via Monte-Carlo approaches. Alternatively, Hamilton et al. [291] proposed to sub-sample the nodes to be updated, to address the memory bottleneck issue.

Another drawback of this algorithm is that it is not conceptually new. Indeed, it is strongly related to the 1-WL algorithm which is well known in the graph mining community. More precisely, it is a differentiable and parametric generalisation of the 1-WL algorithm on graphs. We propose a fundamental analysis of graph neuron networks in appendix D.

Furthermore, recall that such models are mostly trained in an end-to-end manner. This refers to the fact the weights are adjusted based on the back-propagation of the prediction error of a particular prediction task.

More precisely, in the case of node classification, a softmax function taking as input the final node embeddings $\mathbf{h}^{(L)}$ returns node-specific probability vectors.

In the case of edge prediction, the probability $p(A_{ij})$ of an edge between nodes i and j can be computed via $p(A_{ij}) = \sigma((\mathbf{h}_i^{(L)})^T \mathbf{h}_j^{(L)})$.

In the case of graph-level prediction, a fully connected neuron network (FNN), or Multi-Layer Perceptron (MLP) or simply Deep neuron Network (DNN), is usually connected to the graph-level representation to perform the prediction.

The flexibility and representation power of the GCN general formulation in Alg. 1 relies on the aggregation functions. They update node-level representations based on the nodes in their neighbourhood and compute a graph-level representation based on every node representations.

We discuss in the following the modifications that have been proposed in the literature to enhance the performance of end-to-end graph representation learning. In particular, we discuss the most relevant dedicated aggregation functions. This is critical to explore the use of such GCNs for drug virtual screening in the following chapters.

6.2.2 Aggregation functions for graph convolutional networks

The $AGGREGATE^{(l)}_{graph}$ function computes a graph-level representation at step l based on the node representations $\{\mathbf{h}_v^{(l)}\}_{v\in\mathcal{V}}$. The $AGGREGATE^{(l)}_{node}$ function updates the node-level representation $\mathbf{h}_i^{(l)}$ based on the representations in the neighbourhood $\{\mathbf{h}_j^{(l)}\}_{j\in\mathcal{N}(i)}$.

Note that both $AGGREGATE$ functions must be order-invariant transformations because, in the general case, there is no injective node ordering for graphs. However, some algorithms approximate injective node ordering for a particular set of graphs. In the case of molecules, the well known "Morgan algorithm" performs such ordering, but not in an end-to-end fashion. Therefore, we focus on order-invariant $AGGREGATE$ functions.

Second, these functions must also handle different sized neighbourhoods and maintain the weight-sharing property of CNNs.

Thus, both can be considered as multi-set functions, i.e. functions operating on a set of vectors and which output a vector assembling the information of the vectors in the set.

In the following, we introduce in details the aggregation functions that we found the most relevant in the graph representation learning literature.

In this literature, authors use several gold standard benchmark datasets for graph representation learning [291, 295]. We can mention (i) a paper subject category prediction task, performed on a large citation dataset (302,424 nodes with an average degree of 9.15), (ii) a dataset made of Reddit posts for prediction of the community to which they belong (232,965 nodes with an average degree of 492), and (iii) the Protein-Protein Interaction (PPI) network on which protein class is predicted (2373 nodes, with an average degree of 28.8, 50 features per node , 121 labels for each node). Note that the PPI dataset is inductive since 20 PPI networks are used for training, two for testing and two for validation.

Transductive datasets are also considered like the (i) "Cora dataset" (2708 nodes, 5429 edges, 7 classes and 1433 features per node), (ii) the Citeseer dataset (3327 nodes, 4732 edges, 6 classes and 3703 features per node), (iii) the Pubmed dataset (19717 nodes, 44338 edges, three classes and 500 features per node).

All of these datasets are composed of graphs much larger than molecular graphs and they correspond to the prediction of node-level labels. Thus, most of the following approaches were evaluated on settings completely different from the one we are interested in.

Simple aggregation

An obvious multi-set function is the sum (or mean) function. Indeed, the classical way of building a graph-level representation is to sum over the node representations [287, 296, 297, 288, 289, 298]:

$$\mathbf{m}^{(l)} \leftarrow \sum_{i\in\mathcal{V}} \mathbf{h}_i^{(l)}$$

In this case, the sum function captures the distribution of each node features in a single value.

Similarly, [298] passes node features through a shared hidden layer before summation, to explicitly transfer the node-level representation into a graph-level representation. In the following, $\mathbf{W}$ refers to a learnable weight matrix and σ to any activation function (see appendix A.8).

$$\mathbf{m}^{(l)} \leftarrow \sum_{i \in \mathcal{V}} \sigma(\mathbf{W}^{(l)}_{graph} \cdot \mathbf{h}^{(l)}_i)$$

Equivalently, node neighbourhood can also be aggregated by summing their representations. However, in the case of neighbourhood aggregation, the representations necessarily pass through a shared hidden layer before the summation (otherwise, we do not learn any convolutional functions to process the graph).

$$\mathbf{h}^{(l+1)}_i = \sum_{j \in \mathcal{N}(i)} \sigma(\mathbf{W}^{(l)}_1 \cdot \mathbf{h}^{(l)}_j)$$

Alternatively, as in [299], we can normalise incoming messages from neighbours with parameters α_{ij} (fixed or learnable):

$$\mathbf{h}^{(l+1)}_i = \sigma(\mathbf{W}^{(l)}_0 \cdot \mathbf{h}^{(l)}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}^{(l)}_1 \cdot \mathbf{h}^{(l)}_j) \tag{6.1}$$

A second obvious multi-set function is max-pooling, first proposed by [291]. Again, each vector in the set usually passes through a shared layer before an element-wise max-pooling operation is applied to aggregate information across the set. With max-pooling, the model can directly learn the absence or presence of specific features within the set. It is thus efficient for prediction tasks relying on the presence of key elements.

$$AGGREGATE^{(l)}_{graph} = \max_{i \in \mathcal{V}}(\{\sigma(\mathbf{W}^{(l)}_{pool} \cdot \mathbf{h}^{(l)}_i)\}_{i \in \mathcal{V}})$$

Edge type-based aggregation

The $AGGREGATE^{(l)}_{node}$ function can also take the edge attributes as input if they exist, like in the case of molecules [300, 301, 302, 303, 304, 305].

For instance, some studies have considered to sum over edge type-specific standard multi-set functions [301, 304, 305] as below

$$\mathbf{h}^{(l+1)}_i = \mathbf{W}^{(l)}_0 \cdot \mathbf{h}^{(l)}_i + \sum_{j \in \mathcal{N}(i)} \sum_{k=0}^{N_e} \frac{\tilde{A}_{ijk}}{|\mathcal{N}(i)|} \mathbf{W}^{(l)}_k \cdot [\mathbf{h}^{(l)}_j, \mathbf{x}_{ij}]$$

where N_e is the number of edge types, N is the number of nodes, $\tilde{A}$ is the augmented adjacency matrix ($\tilde{A} \in \mathbb{R}^{N*N*N_e}$) where each vector $\tilde{A}_{ij.}$ is a zero vector if there is no edge between nodes i and j, or encode the edge type in a one-hot encoding fashion otherwise.

Note that Shang et al. [305] have reported significant better performance on gold standard chemoinformatics datasets (Tox21, or HIV) with this procedure, compared to the weave module-based GCN of Kearnes et al. [300] presented in appendix C.3, which also consider edge attributes.

Other works [302, 303, 301] also proposed to update edges attributes similarly to node attributes, hoping for more expressive GCN. For instance, Gaidya et al. [301] proposed the following edge convolutional layer:

$$\mathbf{h}_{ij}^{(l+1)} = \sigma(\mathbf{W}_{edge}^{(l)}.[\mathbf{h}_{ij}^{(l)}, \mathbf{h}_{i}^{(l)}, \mathbf{h}_{j}^{(l)}])$$

where $\mathbf{h}_{ij}^{(l)}$ is the latent representation of the edge between nodes i and j at step l.

LSTM-based aggregation

Hamilton et al.[291] also proposed an LSTM-based multi-set function. The LSTM has the advantage of being more expressive than summing or pooling. However, an LSTM learns an order dependent function whereas a multi-set function must not be order-dependent. Therefore, in this study, they applied several LSTMs to random permutations of the vectors in the set.

In the original study [291], they found that such LSTM-based aggregators performed better than the sum-based aggregator on different benchmark datasets for graph representation learning. Note however that the observed difference in performance is about one or two points, or even less, while the accuracy is in the range of 0.80 to 0.95 depending on the dataset. In the following, this method is called GraphSAGE, as in the original paper.

Interestingly, an LSTM-based order-invariant multi-set function has been defined and named attLSTM [306]. It is introduced in appendix A.8.2. It is an iterative way to build a learnable linear combination of the vectors in the set. It was first used for aggregating representation vectors in the work of Altae et al. [307].

Aggregation capturing the distribution of the learned features

Kearnes et al. [308] proposed to design a graph-level description by concatenating the distribution of each feature of the node representations. More precisely, they encoded the distribution of each individual features by a n-bits histogram. In other words, the distribution of each feature is approximated by "n" values, each value corresponding to a frequency to which the feature falls within a small range (a bin). This introduces the idea of directly capturing the distribution of each learned feature.

Attention mechanisms-based aggregation

Powerful aggregation functions have been designed based on attention mechanisms. In particular, Lee et al. [309] proposed a survey about attention models for graph representation learning.

Attention mechanisms are particularly relevant in the case of graphs, since some prediction tasks may rely only on some parts of the graph. In addition, large graphs can be complex and noisy. An attention mechanism can highlight elements with the most task-relevant information, and therefore, can help to improve the signal-to-noise ratio. Another benefit of attention mechanisms is that they allow dealing with unordered and variable sets of any size, focusing on the most relevant parts of the input to make decisions. It is also

computationally efficient, since it is a parallelizable operation across the vectors of the set. Importantly, it can also be used to interpret individual predictions. Indeed, the attention weights, specific to each sample, highlight the most relevant elements for the dedicated prediction. We discuss more lengthily attention mechanisms in appendix A.8.2.

In the context of graph representation learning, there are three prominent ways to define attention mechanism-based $AGGREGATE$ functions.

Attention mechanism-based $AGGREGATE^{(l)}_{node}$ function

Velickovic et al. [295] were the first to consider rigorously an attention mechanism to design a neighbourhood aggregation function for graph representation learning (see Fig. 6.1). More precisely, the multi-set function they defined is:

$$\mathbf{h}_i^{(l+1)} \leftarrow \sigma(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}^{(l)} \cdot \mathbf{h}_j^{(l)})$$

The attention weights α_{ij} are defined by:

$$\alpha_{ij}^{(l)} = \frac{exp(LeakyRelu(\mathbf{a}^{(l)^T}.CONCAT[\mathbf{W}.\mathbf{h}_i^{(l)}, \mathbf{W}.\mathbf{h}_j^{(l)}]))}{\sum_{k \in \mathcal{N}(i)} exp(LeakyRelu(\mathbf{a}^{(l)^T}.CONCAT[\mathbf{W}.\mathbf{h}_i^{(l)}, \mathbf{W}.\mathbf{h}_k^{(l)}]))}$$

The LeakyRelu function is defined by: $f(x) = x \; if \; x > 0$ or $0.01x$ otherwise. It is an activation function like the *Relu* function but with non-zeros outputs for negative inputs, which allows non-zero gradient when the unit is not active. Thus, the gradient can flow even through inactive neurons, which prevents from the so-called "dead" neurons.

Intuitively, a shared linear transformation parameterised by $\mathbf{W}$ is first applied to each node latent representation $\mathbf{h}_i^{(l)}$ and $\mathbf{h}_j^{(l)}$, $\forall j \in \mathcal{N}(i)$. Then, the term $\mathbf{a}^{(l)^T}.CONCAT[\mathbf{W}.\mathbf{h}_i^{(l)}, \mathbf{W}.\mathbf{h}_j^{(l)}])$ computes the attention score of node j based on the similarity between a learnable query vector $\mathbf{a}$, and the concatenation of the transformed embeddings $CONCAT[\mathbf{W}.\mathbf{h}_i^{(l)}, \mathbf{W}.\mathbf{h}_j^{(l)}]$. Finally, the soft-max function computes the attention weights based on the attention scores. This is meant to enforce the attention weights to be close to zero or one, such that it passes the information coming only from relevant neighbours. Importantly, all operations are differentiable.

Note that the authors actually execute K simultaneous attention-based aggregations, since it was found to stabilise the learning process of attention mechanism [310]. The aggregation then results from the concatenation of the K attention-based aggregation functions:

$$\mathbf{h}_i^{(l+1)} \leftarrow CONCAT_k[\sigma(\sum_{j \in \mathcal{N}(i)} \alpha_{ij} \mathbf{W}_k^{(l)}.\mathbf{h}_j^{(l)})]$$

The authors named this method GAT. They obtained, compared to the GraphSAGE approach, slightly better performances on the transductive datasets (from two to three points, depending on the metric), and much better on the inductive dataset (from 77 to 97% in F1score).

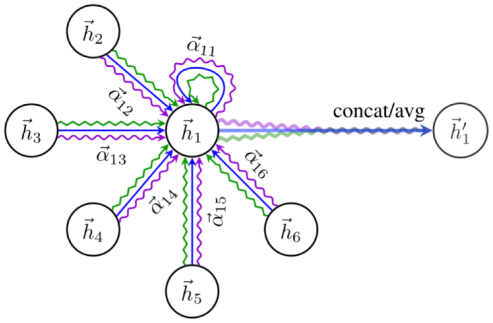


Figure 6.1. GAT: graph neuron networks with attention in [295]

Recently, Shanthamallu et al. [311] quantitatively assessed that the distribution of attention weights for any node obtained by GAT is almost always uniform, in the case of gold standard dataset for graph representation learning. This means that the attention mechanism is not suitable in these cases, since its selects all neighbours as relevant. Intuitively, this behaviour is relevant for nodes with few neighbours, since it is, in this case, more reasonable to use all the available information. However, when the neighbourhood is populated with an adversarial node (i.e. a node that cannot be characterised as part of any consistent community in the graph), we expect the attention mechanism to prioritise only relevant neighbours, by assigning a comparatively low weight to the adversarial nodes.

Thus, authors proposed two additional regularisation terms to the original GAT formulation (with their own hyper-parameter to control their contribution). One enforces a uniform attention weight distribution for each node. A second prevents a node with a high degree to arbitrarily participate in the update of all its neighbours, which is particularly important for noisy or adversarial nodes.

These modifications do not address all the concerns raised, as they do not help more to prioritise relevant neighbours. They avoid adversarial nodes with high degrees to have an inadequate strong influence. In addition, even though they have exhibited slightly higher performance than GAT on gold standard datasets (such as citation or social network datasets), such modifications do not affect molecular graph representation learning, since nodes in molecular graphs have degrees (atoms' valence) in the same range. However, it provides an interesting discussion of the GAT model.

Self-attention mechanism-based $AGGREGATE^{(l)}_{graph}$

Li et al. [312] considered a self-attention mechanism (see appendix A.8.2) to build a graph-level representation, which automatically selects the most relevant nodes for the graph-level task.

In the following equation, the $tanh_{\mathbf{W}_1,\mathbf{b}_1}$ layer transforms the concatenation of final node embeddings $\mathbf{h}_i^{(L)}$ and original features $\mathbf{x}_i$ into a feature vector to be added to the graph representation, and the $\sigma_{\mathbf{W}_0,\mathbf{b}_0}$ layer acts as a learnable gate filtering the information conveyed by the $tanh_{\mathbf{W}_1,\mathbf{b}_1}$ layer.

$$\mathbf{m} = \sum_{i \in \mathcal{V}} \sigma(\mathbf{W}_0.CONCAT(\mathbf{h}_i^{(L)}, \mathbf{x}_i) + \mathbf{b}_0) \odot tanh(\mathbf{W}_1.CONCAT(\mathbf{h}_v^{(L)}, \mathbf{x}_i) + \mathbf{b}_1)$$

Attention mechanism-based walks for graph-level encoding

From another perspective, Lee at al. [313] used L-length walks guided by an attention mechanism to build graph-level embeddings.

The walks process the graph, i.e. they integrate the node attributes $(\mathbf{x}_1, \ldots, \mathbf{x}_L)$ covered by the walk to build a latent representation of the walk. The walks capture the structure of the graph, while adaptively and selectively roaming the graph nodes thanks to an attention mechanism that selects the most task-specific relevant neighbouring node to roam next. In particular, they use an LSTM-based model with a built-in attention mechanism, trained with reinforcement learning.

To build a graph-level embedding, they deploy z LSTMs simultaneously from various parts of the graph, yielding to z walk-level representations $\{\mathbf{s}_1, \ldots, \mathbf{s}_z\}$. A second attention mechanism is then used to determine the relevance of the walk α_i before these are combined to form a graph-level embedding: $\mathbf{m} = \sum_{i=1}^{z} \alpha_i \mathbf{s}_i$.

Furthermore, such strategy can also be used to define an $AGGREGATE_{node}^{(l)}$ function, like in the DeepWalk paper [314] and node2vec paper [315]. These papers used biased random walks to learn to select surrounding nodes, and therefore, define a more selective notion of node's neighbourhood than the 1-hop neighbourhood.

Virtual super node for graph-level encoding

Several authors [316, 317, 318] proposed to learn a graph-level embedding by creating a virtual "supernode", that is connected to all other nodes via a particular type of edge. This virtual node is updated like real nodes, and the final representation of this virtual node is used as the graph representation. Note however that the virtual node is usually not used as a neighbour when updating real node representations. This method does not seem natural and did not exhibit striking performance improvements.

However, Ishiguro et al. [319] showed that a virtual super node, used to update real nodes' latent representations, improved the representation power of a wide variety of existing GNNs. Intuitively, such supernode allows a pair of distant nodes to communicate through the supernode in one hop. However, it can lead to "over-smoothing" of the information. Hence, they equipped the supernode with a trainable gated message passing mechanism to regulate the information that warps through the feature space of the supernode, and to deliver appropriate messages to real nodes in the original graph.

Hierarchical pooling for graph-level encoding

A much more promising approach to define a suitable $AGGREGATE_{graph}$ function is based on differential hierarchical pooling, which has been proposed simultaneously by several groups [320, 321, 301]. It allows for the graph convolutional architecture to iteratively operate on coarser representations of a graph. To be precise, such procedure actually replaces both the $AGGREGATE_{graph}$ and $COMBINE_{graph}$ functions, since it results directly in a final graph-level representation (see fig 6.2).

This idea is motivated by the design of an analogue of the pooling operation for images, which was shown to be of crucial importance for CNN success in the field of image processing. Indeed, one of the limitations of

current graph convolutional networks is that they are inherently flat, since they only propagate information across the edges of the graph, without reducing the size of the graph.

Because of the topological structure of graphs, the definition of a "patch" to be pooled is not straightforward. Depending on the graph and the hierarchy level, we may need to pool varying numbers of nodes and edges. In other words, the pooling strategy needs to generalise across graphs with different nodes, edges, and graph structures during inference. To this end, the patches to be pooled are obtained via a differentiable clustering-like approach.

We present now the work of Ying et al [320].

Formally, at each step l, we are given a graph adjacency matrix $A_l \in \mathbb{R}^{n_l * n_l}$ and a node representation matrix $H_l \in \mathbb{R}^{n_l * f_l}$, where n_l is the number of nodes in the graph and f_l the dimension of the nodes latent representation at step l. For all l, n_l and f_l are hyper-parameters of this approach.

For the next step $l+1$, we differentially cluster the n_l nodes of the graph at step l in order to build a new coarsened graph containing $n_{l+1} < n_l$ nodes, with weighted adjacency matrix $A_{l+1} \in \mathbb{R}^{n_{l+1} * n_{l+1}}$ and clustered node embeddings $X_{l+1} \in \mathbb{R}^{n_{l+1} * f_{l+1}}$. Typically, $f_{l+1} = f_l \ \forall l$. Thus, the pooling strategy requires to learn, at each depth l, a cluster assignment matrix $S_l \in \mathbb{R}^{n_l * n_{l+1}}$ via a standard graph convolutional layer. In S_l, each row corresponds to one of the n_l nodes (or clusters) at layer l, and each column corresponds to one of the n_{l+1} clustered nodes at the next layer $l+1$. Intuitively, S_l provides a soft assignment of each node at layer l to a cluster in the next layer $l+1$. In particular, S_l is defined as:

$$S_l = \text{row-wise softmax}(GraphConv_{pool}^{(l+1)}(A_l, X_l))$$

. The $GraphConv$ function refers to the update of encoding for all nodes in the graph, following for instance eq. 6.1.

Then, the coarse graph adjacency matrix A_{l+1} and clustered node embedding matrix X_{l+1} are defined as:

$$\begin{aligned} A_{l+1} &= S_l^T A_l S_l \in \mathbb{R}^{n_{l+1} * n_{l+1}} \\ X_{l+1} &= S_l^T A_l \in \mathbb{R}^{n_{l+1} * f_l} \end{aligned}$$

Finally, X_{l+1} is in turn updated into H_{l+1} via a graph convolutional layer:

$$H_{l+1} = GraphConv_{update}^{(l+1)}(A_{l+1}, X_{l+1})$$

.

This process is repeated L times, based on a series of L $GraphConv_{update}$ and $GraphConv_{pool}$ layers operating on coarser versions of the input graph (see Fig 6.2).

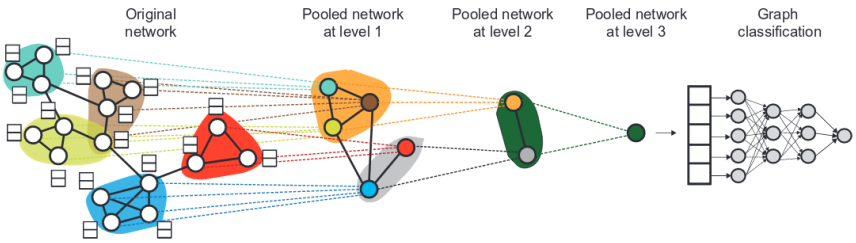


Figure 6.2. Graph neuron networks with hierarchical pooling in [320]

Note that A_{l+1} is a real matrix that represents a fully connected edge-weighted graph. Therefore, to enforce hard assignment clustering at each step, two additional losses are added to the prediction loss: $L_{LP} = ||A_l - S_l S_l^T||_F$ and $L_E = \frac{1}{n}\sum_{i=1}^{n} H(S_i)$ (H is the entropy function). L_{LP} encodes the intuition that nearby nodes should be pooled together. L_E enforces the cluster assignment matrix to trigger a hard clustering (i.e. enforce each node to be mostly assigned to a single cluster, and very little assigned to the other clusters).

Regarding performance, the original paper [320] reports that this hierarchical pooling GCN outperformed GraphSAGE [291] by seven points on graph prediction gold standard datasets (in a context where other studies usually reported improvement of a single point).

A slightly different cluster-wise node aggregation scheme has been proposed by Porrello et al. [321]. This approach first selects, for each cluster, a number of candidate nodes (set to five in this study) whose feature vectors are aggregated via a standard convolution. Ultimately, the concept remains the same.

6.2.3 Graph-level representation combination for graph convolutional networks

To conclude this section, we discuss the $COMBINE^{graph}$ function. Recall that the $COMBINE^{graph}$ function aggregates the molecular embedding $\mathbf{m}^{(l)}$ at all steps l, to build the final graph-level representation $\mathbf{m}$.

In most cases, the last graph-level embedding $\mathbf{m}^{(L)}$ is set directly as the graph-level representation $\mathbf{m}$.

Furthermore, graph convolutional networks rarely consider more than two layers so that combining graph-level embeddings at different levels is relatively not a concern. Kipf et al. [322] reported that considering more than two layers performs worse with their model. Although deeper models can in principle access to more information, some of the information may be "washed out" by too many aggregations.

Some studies alternatively considered either the sum [287, 298] or the concatenation [295] of the $\{\mathbf{m}^{(l)}\}$s to define a final graph-level representation $\mathbf{m}$. The concatenation stores explicitly sub-graph information at different granularity. Indeed, the aggregation process (at every nodes), after l iterations, learns the topology

as well as the distribution of node features in the l-hop neighbourhood. It also allows the network to skip the last layers if they appear to be harmful for the prediction.

A smoother $COMBINE^{graph}$ function has been proposed by Xu et al. [323]. The authors proposed to flexibly adjust (i.e. learn) the molecular embedding $\mathbf{m}$ depending on nodes and tasks, by carefully selecting intermediate representations $\mathbf{m}^{(l)}$, and potentially combining a few.

More precisely, they considered three possibilities of intermediate representations combination. First, a concatenation of linear transformation of the intermediate representations: $CONCAT_l[\{\sigma(\mathbf{W}.\mathbf{m}^{(l)})\}_l]$. This is not node adaptive as it uses the same transformation for all nodes. Second, an element-wise max-pooling of intermediate representations. It is adaptive and it does not introduce any additional parameters to learn. Third, a biLSTM-attention module processing the sequence $(\mathbf{m}^{(1)}, \ldots, \mathbf{m}^{(L)})$ which is node adaptive, but may overfit on small graphs due to its relatively high complexity.

Their result shows, on average, one point improvement over state-of-the-art graph convolutional networks (GraphSAGE [324] and GAT [295]) on graph prediction gold standard datasets. For the intermediate representation combination, max-pooling or biLSTM-att performed better on small and large graphs data respectively (by only a single point, however).

6.2.4 Conclusion for our studies

In this chapter, we got "fully" empowered, to explore representation learning on molecular graphs and protein sequences for drug virtual screening.

Among the works we surveyed, the GAT and hierarchical pooling approaches appeared the most promising. First, they resulted in performance improvements higher than those reported in other studies, which usually show very little performance increase (one or two points maximum) on graph prediction gold standard datasets. Second, the concepts on which they rely appear particularly relevant to our problem.

The GAT $AGGREGATE_{node}$ function seems particularly powerful since it aggregates neighbouring information after identifying the most suitable task-specific neighbours at each node. However, we are concerned that this could be a smoke screen in the case of molecular graphs, since they are small graphs for which all atoms are supposedly relevant.

The hierarchical pooling strategy seems particularly relevant since it is the graph analogue of the pooling operation for images, which was shown to be of crucial importance for CNN success in image processing. However, we are concerned that this could be not particularly suitable for molecular graph. Indeed, they are small enough so that iterative convolutional node updates, followed by a simple summation of graph nodes to build a graph-level embedding could equivalently encode and hierarchically agglomerate graph substructures of different sizes.

In the next chapter, we start by an organised critical review of the works on drug-target interaction prediction with deep learning-based feature encoders. Based on the analysis of this review, we raise issues and openings to enhance data-driven approaches for chemogenomics. Finally, in chapter 8, we investigate the identified promising directions in the chemoinformatics, proteomics and chemogenomics frameworks.

Chapter 7

Deep-learning based features encoders for drug virtual screening

Abstract: *Recent developments in deep learning models are expected to be highly competitive with existing drug virtual screening approaches. Indeed, they resulted in significant improvement in some popular yet challenging fields such as computer vision and speech recognition & translation. As introduced in the previous chapter, representation learning on molecules and proteins can be performed either on top of expert-based features or on the molecular graph and protein sequence. Therefore, in this chapter, we first survey multi-layer perceptron to derive abstract features from expert-based features. In a second phase, we review more promising approaches that consist of using dedicated neural architectures, that we introduced in the previous chapter, to directly extract descriptors automatically on molecular graphs and protein sequences. Finally, we conclude this chapter by drawing a critical summary of our survey.*

Résumé: *Les développements récents des méthodes d'apprentissage en profondeur sont attendus pour être très compétitifs par rapport aux approches existantes de criblage virtuel de médicament. En effet, ces méthodes sont à l'origine d'améliorations significatives dans certains domaines, à la fois populaires et difficiles, tels que la vision par ordinateur et la reconnaissance de la parole. Comme cela a été présenté dans le chapitre précédent, l'apprentissage de la représentation de molécules et de protéines peut s'effectuer soit sur des descripteurs extraits par les experts, soit directement sur la base des données brutes. Par conséquent, dans ce chapitre, nous examinons d'abord le perceptron multicouche afin d'obtenir des attributs abstraits à partir des attributs basés sur l'expertise. Dans une seconde phase, nous passons en revue des approches plus prometteuses consistant à utiliser des architectures neuronales dédiées, que nous avons présentées au chapitre précédent, pour extraire les descripteurs automatiquement sur les graphes moléculaires et les séquences protéiques. Enfin, nous concluons ce chapitre en faisant un résumé critique de notre enquête.*

Before introducing previous works on representation learning for drug virtual screening, we recall the two main concepts differentiating DTI prediction approaches.

On the one hand, there are ligand-based approaches [325, 326, 327], taking molecular features as input to predict interactions with one protein, or a relatively small number of proteins (i.e. multitask ligand-based virtual screening). On the other hand, chemogenomics approaches take protein-ligand pairs as input to predict DTI interactions. When trained on a wide range of proteins and ligands, such approaches can make proteome-wide predictions [328, 329, 281, 330].

In addition, methods also differ on the molecular and protein features used. Some considered data-blinded features, or equivalently "a priori" expert-based features [325, 326, 328, 329, 327]. In practice, they can be recovered via available toolkits (toolkits are presented Section 2.3.3). Others are data-driven, i.e. they use feature extractors to learn abstract representations of the molecular graphs and protein sequences in an end-to-end manner [281, 330].

In the first section of this chapter, we discuss methods based on feed forward neuron networks with data-blinded features as input (Section 7.1). Such approaches perform representation learning at a "high level" of initial description, since, in this case, feed forward neuron networks do not process the raw data (i.e. molecular graph and protein sequences) but pre-defined data-blinded features. In the second subsection, we thus discuss more appealing representation learning approaches for drug virtual screening that are directly built from the molecular graphs and the protein sequences.

In the second section of this chapter, we keep on investigating the approaches based on end-to-end data-driven learnt representations, but in the chemogenomics framework, since we are mostly interested into proteome-wide drug-target interaction prediction.

Finally, we summarise shortly the conclusions we can draw from the review of the literature.

7.1 Deep learning-based feature encoders for ligand-based drug virtual screening

7.1.1 Feed-forward neuron networks on expert-based features for ligand-based drug virtual screening

The striking arrival of deep neuron networks into the field of DTI prediction occurred during the virtual screening Merk challenge. The Merk company offered a high-quality QSAR dataset. The winners considered an ensemble approaches made of FNN, RF, SVM and others, since ensemble approaches are usually beneficial. However, their success relied mostly on the use of FNN.

After the indisputable success of FNN in the virtual screening Merk challenge, further studies carefully evaluated the performance of FNN while studying the effect of hyper-parameters and the multitask aspect of this approach. In particular, the winners of the Merk QSAR challenge proposed some studies [331, 332] investigating FNN for virtual screening.

In the first study [331], they employed multitask FNN to predict the activities of around 140,000 compounds against 19 target assays from the PubChem database. They use around 3000 structural and physicochemical descriptors generated by the DRAGON toolkit. Moreover, they simplified the QSAR affinity prediction problem to a classification problem (interaction or not interaction). They showed that multitask neuron networks, i.e. an FNN with 19 output units (one per bio-assay), outperformed random forest, gradient boosted decision tree ensemble and logistic regression methods, as well as singletask FNN on most tasks. We emphasise that multitask FNN only performed slightly better in most cases, even when compared with singletask FNN. They also observed no significant performance gain from feature selection.

A second study [332] analysed the effect of hyper-parameters of FNN: the number of layers (from one to four), the number of neurons per hidden layer (from 100 to 4500, the learning schedule parameters, the type of activation or the regularisation hyper-parameters. Note that we introduce and discuss deep neuron network hyper-parameters in appendix A.8.4. While still considering the same molecular descriptors, they consider an extended dataset of 30 bio-assays (tasks) with around 130,000 unique compounds. Although they tested Bayesian optimisation for task-specific hyper-parameter selection (see appendix A.8.4), they reported that the use of a single set of hyper-parameters for all tasks could perform as well as optimised parameters on each task. The authors [331, 332] also reported that the performance of the singletask FNNs increase with the amount of training data. These observations motivated further studies to understand the efficiency of multitask learning with FNNs.

First, Ramsundar et al. [333] considered a pyramidal multitask FNN (i.e, each layer has fewer neurons than the previous layer) to predict the interactions of around 1.6 million compounds, encoded by structural fingerprints, against 259 targets. They showed that a pyramidal architecture maintained performance while reducing the number of parameters. The authors also reported that pyramidal multitask deep neuron networks result in significant improvement over standard machine-learning algorithms, namely logistic regression, random forest and singletask FNN (they did not test SVM-based models). More importantly, they noticed that the amount of shared bio-active compounds between tasks is moderately correlated with multitasking performance improvement, and that the biological class of the targets is not. Moreover, they found that adding more training data still improved the multitask model, which suggests that the multitask performance could still be improved with more data.

Later, Xu et al.'s work [334] studied the effect of multitask learning in the same setting as Dahl et al. [332]. They reported that multitask learning is only beneficial for two tasks sharing a large pool of input compounds with correlated or anti-correlated bio-activities. In many cases, the improvement is not observed by an increase of prediction performance, but by an increased speed of convergence. In the case of uncorrelated activities, a negative transfer has been observed.

In other words, within ligand-based drug virtual screening, multitask learning has been reported as a panacea whereby merely increasing the number of tasks improves performance [333]. Other studies have reported evidence of negative transfer in drug discovery [334]. Thus, choosing the right support tasks to match a particular target task may be crucial for positive transfer learning.

Other recently published studies reported more contrasting results.

Korotcov et al. [335] compared FNN-based and SVM-based approaches taking as input ECFP6 fingerprints for the prediction of several chemical endpoints comprising biological activity, solubility and ADME properties. They reported that FNN performed better than the SVM approach, but only for some metrics.

Furthermore, Bajorath et al. [336] compared FNN to data-blinded approaches for the prediction of 100,000 compounds against 53 different targets. Their results did not show any superiority of the deep learning approach regarding performance. To explain this disappointment, the authors blamed the relatively small size of the datasets, which is unsuitable to demonstrate the full potential of FNNs, although the dataset was composed of 100,000 samples.

Other FNN applications in chemoinformatics

FNN have been widely used and reported to outperform state-of-the-art methods in numerous VS studies [331, 332, 334, 333, 328, 330, 337]. Furthermore, FNN have also been successfully considered for the prediction of other molecular properties.

Among them, the Tox21 challenge [338] proposed several prediction tasks related to molecular toxicity, either in terms of interaction with nuclear receptor signalling pathways, or regarding phenotypes such as cellular stress and so on. The multitask FNN (i.e. an output neuron per task) outperformed singletask FNN, because the task datasets shared a large pool of molecules. Moreover, it also overtook the performance obtained with an ensemble approach wrapping an SVM based on the Tanimoto kernel with structural fingerprints and an RF-based model. They also notice that some of the abstract features learnt by the FNN correspond to well-known toxicophores.

However, these results should be contrasted. Very recently, Karim et al. [339] reported efficient toxicity prediction using simple feature selection combined with a shallow neuron network (SNN). In this study, they considered several toxicity datasets with some thousands of recorded compounds. As input features, they calculated around 1,500 chemical descriptors such as "atom counts"," solubility", "path counts", "ring counts", "topological charge", or "topological distance matrix". These features were ranked on the basis of the Gini index (see appendix A.4), and they selected only the n top-ranked features (n in the order of ten). Finally, they used a simple shallow neuron network with only 1 hidden layer of 10 neurons to perform the prediction. In the singletask framework, they reported better performance with this SNN procedure compared to the direct use of an FNN, for 1 minute of CPU time computation instead of 10 minutes of GPU time for the FNN.

Other studies successfully used FNN for therapeutic classification [340], or modelling drug-like molecules [341].

At this stage of the discussion, representation learning on data-blinded features with FNN appears as a reliable baseline for chemoinformatic prediction tasks, as well as SVM with the Tanimoto similarity measure on molecular fingerprints.

We now investigate data-driven feature extraction with graph neuron networks.

7.1.2 End-to-end feature extraction with graph neuron networks for ligand-based drug virtual screening

Contrasted performance of graph neuron networks in chemoinformatic problems

First, the pioneering work on graph neuron networks of Duvenaud et al. [287] (see appendix C.3) showed slightly improved performance compared to models using standard descriptors, for molecular solubility and photovoltaic efficiency datasets. However, the prediction of biological activity did not benefit from their data-driven feature extraction approach.

Another major study in the field, by Kearnes et al. [308] (see appendix C.3), explicitly concluded that standard graph convolutional networks do not consistently outperform standard data-blinded descriptors-based models. However, they presented their work as a valuable extension of method repertoire and a proof of concept, i.e. a new flexible and efficient framework with room for improvements.

In the study of Rodriguez et al. [342], the authors compared RF, FNN and SVM taking as input ECFP4 descriptors to graph convolutional networks-based model [287] on a PubChem-based dataset containing three assays (one assay per protein target) and more than 100,000 compounds. They report that FNN and graph convolutional networks slightly under-perform SVM and RF, by few points of ROCAUC score in the worst case. However, no standard deviation are displayed.

Other studies have reported that graph neuron networks-based methods reached state-of-the-art performance for some chemoinformatics tasks.

Hop et al. [343] reported that graph convolutional neuron networks outperformed an RF-based model taking as input the concatenation of 101 RDKit descriptors and the 384-bit wide ECFP4 fingerprint. They even show the mean, standard deviation and p-value of the statistical test of the difference of performance. However, such RF-based model has not been reported to be state-of-the-art. Moreover, the prediction tasks considered are not related to protein interactions. The authors considered the prediction of acid-base dissociation constant, human intrinsic clearance (i.e. the rate at which the human body eliminates the drug) and thermodynamic solubility.

Another study [344] explicitly showed that an FNN-based model taking as input data-driven features seemed to perform better than the same model on data-blinded features, which in turn performs as well as the KronRLS. However, the authors did not display standard deviations. They considered several DTI data sets (Davis, Metz, KIBA and Toxcast) containing between around 30,000 interactions to 530,000 interactions.

Despite this controversial success, several studies investigated GCN for multitask ligand-based virtual screening. It is a typical promising direction to address the lack of bio-activity data for each individual targets (i.e. tasks).

Performance of graph neuron networks for multi-target ligand-based virtual screening

Among recently published studies, one has focused rigorously on the multitasking aspect of ligand-based DTI prediction with graph neuron network [345]. More specifically, they studied the choice of source tasks for positive transfer learning. Note that the choice of source tasks is conceptually similar to the bias-variance trade-off. Indeed, selecting closely related tasks may help to build a robust local data-driven representation, but is unlikely to generalise. In contrary, considering very dissimilar tasks enables to learn regularised representation and is unlikely to yield a strong local performance.

They considered 48 prediction tasks for molecules, taken from the MolNet collection of datasets. Some of the prediction tasks are interaction with specific targets, some are related to basic chemical properties such as molecular solubility. Regarding the ML models, they considered a data-driven approach in which the molecular graph encoder is the weave module by Kearnes et al. [308] (see appendix C.3), and the molecular level embedding is built by an attLSTM [306] (see appendix A.8.2). This encoder is used to learn a molecule-level representation for all input data, but task-specific fully connected layers are used to perform task-specific predictions. It corresponds to the "hard sharing multitask" framework (see appendix A.8.3 for an introduction to multitask with neuron networks).

To analyse multitask performance with respect to singletask performance, they quantitatively assess the similarity s_{ij} between any pairs of tasks i and j as the difference between MT and ST final performances: $S_{i,j} = \mathcal{L}_{i,j}^{MT} - \mathcal{L}_{i}^{ST}$, where $\mathcal{L}_{ij}^{MT}$ is the performance obtained while training the molecular graph encoder jointly on tasks i and j, and $\mathcal{L}_{i}^{ST}$ is the performance obtained while training the molecular graph encoder only on task i. They did not include higher order effects by considering more than two tasks simultaneously. Based on that similarity measure between tasks, they build two task similarity matrices. One when using data-driven features (graph convolutional networks), and the other when using data-blinded Morgan fingerprints. The former gave a richer (i.e. more diverse) similarity matrix than the latter. The authors argued that the greater representation flexibility afforded by graph convolutional neuron networks may allow capturing a richer task information landscape.

Furthermore, in both cases, the task similarity matrices can be understood based on some underlying physicochemical properties. Indeed, they observed that basic physical property prediction tasks benefited more from co-training with other basic property prediction tasks, than with biological tasks to which they are less obviously related. Reciprocally, the biological prediction tasks are more likely to benefit from features that express steric interactions which are less relevant for the prediction of basic physical properties.

Finally, they reported that actively selecting source tasks is better for transfer learning than random selection of source tasks. Nevertheless, the singletask approach provided the best performance for most of the hold out tasks (90% of the tasks). These results put into perspectives multitask learning for ligand-based drug virtual screening.

Another significant study [346] proposed a rigorous comparison of various methods for multitask ligand-based DTI prediction. In particular, they compared SVM with the Tanimoto similarity measure on different kinds of fingerprints, RF, FNN, GCN [287, 308], SMILES-LSTM [283] and kNN. The benchmark dataset was composed of 500,000 compounds and 1000 assays from the ChEMBL database. It is essential to consider

a dataset with many bio-assays, as deep learning can express its full power in this case, allowing hidden representations to be shared among prediction tasks. In particular, in this dataset, few ligands were known per target (i.e. task) and assays with few measurements are likely to benefit from the data of similar assays.

In addition, they assessed performances by cluster cross-validation to address the sampling bias issue (see section 1.3). Indeed, compounds are usually synthesised by chemists as series of molecules deriving from a common scaffold by adding various functional groups. However, we are mostly interested into evaluating bioactivity of compounds dissimilar to the recorded ones, so that cluster-cross validation is suited.

More precisely, they considered *nested* cluster cross-validation, to avoid the hyper-parameter selection bias of deep learning approaches. Indeed, deep learning networks require many hyper-parameters, and their optimisation requires "early stopping" in order to avoid overfitting (i.e., stopping the training if the performance evaluated on validation data, different from the final test data, decreases for some successive training epochs).

Based on this rigorous comparison settings, they reported that FNNs significantly outperform the other tested methods, although by only one point, for all prediction tasks and all types of fingerprints. In particular, they found that FNNs are significantly better than graph convolutional networks [287, 308] (see appendix C.3) or SmilesLSTM [283] (see section 6.1.2), for almost all feature types. Moreover, the second best method is SVM which also performs better than graph convolutional networks [287, 308] or SmilesLSTM [283]. Interestingly, even the SmilesLSTM [283] has a higher average ROCAUC score than the two graph-based representation learning approaches [287, 308].

However, we report, in figure 7.1, the performance plot of the original paper. We observe that the performances are in fact almost all the same. No p-values are reported.

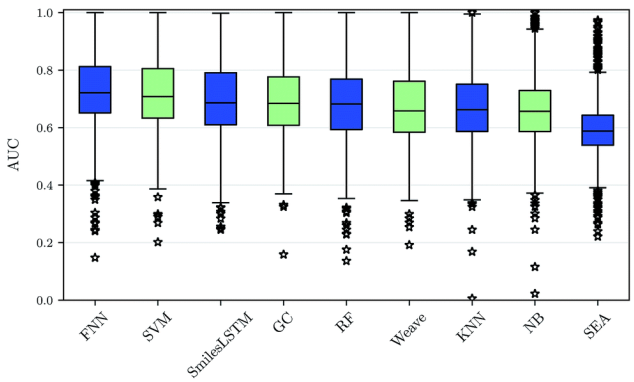


Figure 7.1. Results as reported in [346].

Recall that we are to draw some conclusions from the literature at the end of this chapter. At this stage, we can certainly doubt the performance of standard graph neural networks for ligand-based virtual screening. However, GCNs remain, a priori, appealing thanks to the remaining large room of improvement that we discussed in chapter 6.

7.2 Molecular graph and protein sequence encoders for chemogenomics

We now focus on recent studies which evaluated data-driven feature extractors of molecular graphs and protein sequences, in the context of proteome-wide drug-target interaction prediction.

In the chemogenomics framework, data-driven approaches rely on the automatic encoding of protein and molecule representations, respectively built from a protein sequence encoder (typically RNN or 1D CNN) and molecular graph or SMILES encoder (graph neuron network). The two data-driven representations are combined to build a pairwise latent representation that is then fed into a feed-forward neuron network to predict the affinity between the molecule and the protein.

In the following, we refer to this neuron network architecture as the "chemogenomic neuron network", or simply, the "chemogenomic network".

The following works mostly discuss neuron architectures to efficiently combine the learnt protein and molecule representations.

As a first natural attempt to combine the molecule and protein learnt representations, one can propose to concatenate them. Thus, any type of protein sequence and molecular graph encoders that we introduced in the previous chapter can be used.

For instance, Ozturk et el. [347] used the following architecture: (i) three convolutional layers on SMILES repreŝentations to encode compounds, (ii) three convolutional layers on protein sequences to encode proteins, and (iii) three fully connected layers on the concatenation of the molecule and protein encodings, in order to predict the affinity value between the tested molecule and protein.

Regarding the performance assessment, the authors considered the Davis and Kiba datasets, both containing selectivity assays in the kinase protein family. Hence, note that this study focuses on DTI prediction within a single protein family. The Davis dataset contains interactions between 442 proteins and 68 ligands and the Kiba dataset contains interactions between 467 targets and 52 498 drugs.

The authors compared their data-driven method to Simboost (a gradient boosting machine-based method, cf. section 3.2.2) and KronRLS (cf. section 3.2.2). Importantly, they did not compare to standard FNN, which led to state-of-the-art performances in many studies [346, 308, 342]. In addition, they did not compare to methods shown to outperform KronRLS, in particular, the *NRLMF* approach [136] (cf. section 3.2.2).

They obtained better or similar performance compared to Simboost and KronRLS in terms of MSE. When discretising the datasets (they set the threshold of pKd value, distinguishing actives from inactives, to 7), their method performed the best, in terms of AUPR, for both datasets, but significantly better only for the KIBA dataset.

From another point of view, Gao et al. [348] did not consider the direct concatenation of learned molecule and protein representations as the input of an FNN to predict the binary interaction of a molecule-protein pair.

Instead, they first projected atom and amino acid representations on the same space following: $A = tanh(P^T UD)$ where U is a learnable projection matrix, $P \in \mathbb{R}^{H_p * L_p}$ (H_p amino acids' representation of dimension L_p along the protein sequence) and $D \in \mathbb{R}^{H_d * L_d}$ (H_d atoms' representation of dimension L_d). Then, they defined attention weights for amino acids and atoms following: $[\alpha_p]_i = \max_{1 \leq j \leq L_d} A_{i,j}$ and $[\alpha_d]_j = \max_{1 \leq i \leq L_p} A_{i,j}$. The final protein-level and molecule-level representations are given by: $\mathbf{h}_{prot} = P \cdot softmax(\alpha_p)$ and $\mathbf{h}_{mol} = D \cdot softmax(\alpha_d)$. Thus, the learnable projection on a shared space enabled to learn to select information of task-specific relevant atoms (whose representation encodes its neighbouring substructure) and amino acids (whose representation encode its neighbouring sub-sequence).

Then, a "siamese network" is used to compute the interaction probability. A siamese network is a couple of identical MLPs, each taking the projected representations $\mathbf{h}_{mol}$ or $\mathbf{h}_{prot}$ as inputs and outputting their similarity value. The similarity value is interpreted in this context as the interaction probability.

Attention mechanism can be used to interpret individual predictions. In this study, they used docking to "confirm" the top-ranked predicted interactions. Indeed, they checked that the interacting parts of the protein and the molecule highlighted by the attention mechanism are indeed involved in the interaction via the docking pose.

To conclude, we introduce now the work of Tsubaki et al. [349] inspired from the previous work [348]. Among recently published studies considering the chemogenomic network, this work is of significant interest.

In this study, the molecular graph feature extractor is a standard graph convolutional neuron network (see Section 6.2), and the protein feature extractor is a convolutional network on sequences. The molecular graph-level representation $\mathbf{h}_{mol}$ is learnt by summing over the atom level representations. No protein-level representation $\mathbf{h}_{prot}$ is learnt directly from the abstracted amino acid representations $\mathbf{h}_{(aa)_i}$.

Indeed, the protein level representation $\mathbf{h}_{prot}$ is learnt via an attention mechanism whose weights are learnt via the similarity between a learnable linear transformation (parameterised by $\mathbf{W}_{inter}$ and $\mathbf{b}_{inter}$) of the graph-level and amino acid-level representations: $\mathbf{h}_{prot} = \sum_{i=1} \alpha_i \mathbf{z}_{(aa)_i}$ in which $\alpha_i = softmax(\mathbf{z}_{mol}^T \mathbf{z}_{(aa)_i})$, $\mathbf{z}_{mol} = \sigma(\mathbf{W}_{inter}\mathbf{h}_{mol} + \mathbf{b}_{inter})$ and $\mathbf{z}_{(aa)_i} = \sigma(\mathbf{W}_{inter}\mathbf{h}_{(aa)_i} + \mathbf{b}_{inter})$.

Again, this attention mechanism allows interpretation of individual predictions. Indeed, the weight α_i on each amino acid of index i indicates what part of the protein sequence was necessary for the prediction. As in [348], interpretability was tested by comparing the interacting parts of the protein found by the attention mechanism, and those observed when docking the ligand into the binding pocket of its target. However, regarding performance, the authors did not justify this choice by comparing the performance to those obtained without the attention mechanism.

Concerning performance assessment, the authors used two datasets built based on the DrugBank database. First, a human data set with around 3,500 positive interactions between about 1,000 unique compounds and 850 unique human proteins. Second, a C.elegans dataset containing around 4,000 positive interactions between 1,434 unique compounds and 2,504 unique C.elegans proteins. They compared the performance of their data-driven approach to an approach based on SVM with fingerprint-RBF kernel, on various test "positive:negative" ratios (1:1, 1:3 and 1:5). Indeed, much more negative than positive interactions are recorded, and considering more negative interactions in the test set correspond to a more realistic setting in the context

of proteome-wide DTI prediction.

For the human-specific dataset, the precision score is always higher for SVM (around 5 points better), but conversely, the ROCAUC is always higher for the data-driven approaches (from 1 to 5 points better). Overall, the reached ROCAUC varied in the range of 0.91 to 0.97. For the recall score, SVM suffers more from an imbalanced positive:negative ratio than the data-driven approach. In conclusion, one method or the other performs better depending on the metrics that is considered.

Unexpectedly, when comparing to the state-of-the-art techniques on the gold standard Yamanishi dataset, they do not provide the performance results. However, they do display a bar plot (reported in Fig. 7.2) albeit with no standard deviation. One cannot distinguish the performance of the KBMF2K [134] (introduced in section 3) and that of their method.

Moreover, the Yamanishi dataset is relatively small so that it is not suited to evaluate fairly deep learning models. In addition, we wonder why the authors compared to the KBMF2K and kronRLS methods while they have been shown to be outperformed by *NRLMF* on the Yamanishi dataset.

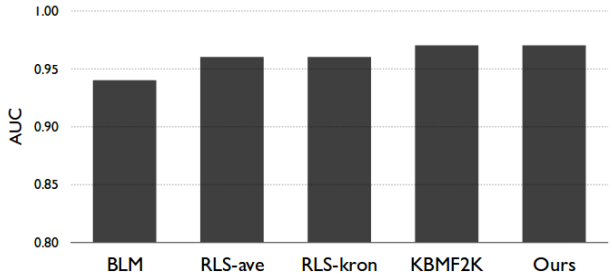


Figure 7.2. ROCAUC scores of various methods designed for DTI prediction on the gold standard Yamanishi dataset as reported in [349]. All baseline methods are introduced in section 3

To conclude this chapter, we now draw conclusions from the above review. This critical summary is meant to drive us to investigate the most promising directions for improving chemogenomic data-driven approaches.

7.3 Summary and perspectives

Regarding algorithmic consideration, some early works [322, 287] showed that graph convolutional networks extract structural features in a similar fashion as standard ECFP procedures [287], and both similarly as the 1-WL algorithm [322]. Thus, conceptually speaking, graph neuron networks are not a new way to process graphs. However, GCNs extract structural features in a differentiable and data-driven formulation.

Others [350, 351] emphasise that GCNs can, at best, be as discriminative as the 1-WL (in terms of the ability to distinguish non-isomorphic graphs with the extracted features). We discuss in more details the algorithmic analysis of graph convolutional networks in appendix D.

The flexibility of GCN alone is expected to enhance their representation power, and therefore, to enable them to overtake state-of-the-art performance.

From another perspective, when visualising the extracted features [345, 352], it seems that GCN enable greater representational flexibility than standard fingerprints. For instance, Fare et al. [345] observed that the 2D projection of data-driven fingerprints clearly discriminated active and inactive compounds, whereas standard ECFP failed to discriminate between the two groups in this 2D chemical space.

However, such linear projection is not sufficient to justify the superiority of data-driven features over standard expert-based data-blinded features. Indeed, data-driven features have been extracted so that they can be classified by a classical linear logistic regression, whereas expert-based data-blinded features are meant to be processed non linearly via SVM or MLP before performing predictions.

Furthermore, the performance improvements of data-driven approaches are very contrasted in chemoinformatics.

In the case of chemoinformatics prediction tasks, several authors [287, 308] observed that GCN do not consistently outperform standard descriptor-based models. Others [344, 343] also emphasised that FNN with data-blinded features performs the best, including with respect to graph neuron networks.

In the particular case of multitask ligand-based virtual screening, only Ramsundar et al. [333] reported that multitask FNN outperforms other approaches, and that adding more training data was still improving the multitask model. Others [331] showed that multitask FNN performed slightly better than singletask FNN only for some tasks. This was confirmed by Xu el al. [334] which showed that positive transfer learning is conditioned by sharing a significant amount of input data whose labels are correlated.

In general, others [335, 336, 339, 342] stated that FNN performs better than data-blinded methods only for some metrics and datasets.

All ligand-based virtual screening studies reported that standard GNNs did not significantly outperform data-blinded feature-based methods, and FNN in particular [342, 346]. Moreover, hard-sharing the parameters of GNN over multiple ligand-based screening tasks was reported inefficient in most cases [345].

In the chemogenomics framework, the combination of standard molecular graph and protein sequence encoders (CNN or RNN) was reported to be competitive with data-blinded approaches such as FNN and SVM-based approaches [348, 349]. Moreover, these authors did not discuss their choice of neuron architecture. Furthermore, only one study [347] reported that their method significantly overtook the others, on a dataset devoted to a single protein family.

When looking at the numbers, we find that whatever method is reported as the best, the differences in terms of performance are very small. We are far from the striking gap of performance obtained by deep learning-based models in the image processing and natural language processing fields. Moreover, the datasets considered in the studies that we discussed are very heterogeneous, and recent promising developments in GCN have not been evaluated in chemoinformatics, and in particular, in chemogenomics.

Overall, we cannot rely on the literature to conclude whether graph neuron networks and related very recent developments can lead to significant improvements for chemogenomics. At this stage, we can seriously doubt the potential of graph neuron networks to extract relevant features from molecular graphs and of CNN or RNN to extract more relevant features from protein sequences, compared to expert-based features.

In the next chapter, we stress out the most promising directions to improve data-driven extraction on molecular graphs and protein sequences, and we investigate them experimentally on various chemoinformatics, proteomics and chemogenomics datasets.

Chapter 8

Evaluation of end-to-end extracted representations for chemogenomics

Abstract: *Based on the critical appraisal of the literature in the previous chapter, we state that recent development in deep learning methods did not exhibit the same sticking improvements in the field of chemoinformatics as in the field of computer vision. In the meantime, a considerable number of publications staged such methods for chemoinformatics prediction tasks, or attempted to empower graph representation learning. In this chapter, we discuss the suitability of deep learning methods for chemogenomics, and more specifically, we challenge a variety of promising neuron modules to encode molecular graphs and protein sequences. To test and discuss very recent developments highlighted in previous chapters, we conduct several experiments on chemoinformatics, proteomics and chemogenomics datasets.*

Résumé: *D'après l'évaluation critique de la littérature réalisée au chapitre précédent, les progrès récents en matière de méthodes d'apprentissage en profondeur n'ont pas montré les mêmes progrès dans le domaine de la chimio-informatique que dans le domaine de la vision par ordinateur. Entre-temps, un nombre considérable de publications ont exhibé de telles méthodes pour des tâches de prédiction de chimioinformatique, ou tenté d'empuissanter l'apprentissage de la représentation sur graphe. Dans ce chapitre, nous discutons de l'adéquation des méthodes d'apprentissage en profondeur pour la chimiogénomique et, plus spécifiquement, nous mettons au défi diverses approches prometteuses fondées sur l'apprentissage de la représentation à partir du graphe moléculaire et de la séquence protéique. Pour tester et discuter des développements très récents mis en évidence dans les deux chapitres précédents, nous avons mené plusieurs expériences sur des bases de données de chimioinformatique, protéomique et chimiogénomique.*

8.1 Improving end-to-end extracted representation

In this section, we explore how to improve molecular graph and protein sequence representation learning. Recall that all evaluation metrics and procedures are introduced in appendix A.2 if necessary. Besides, deep learning is broadly introduced in appendix A.8.

8.1.1 Graph convolutional network architecture

Within the graph neuron network framework defined in section 6.2, many efforts were devoted in the last two years to design flexible and relevant aggregation functions. These aggregation functions update nodes representation from neighbouring nodes, and build a graph-level representation from node-level representations (see Alg. 1).

From our point of view, these efforts were necessary, since the "classical" summation on a shared hidden layer relied on the ability of the network to learn graph-level representations from a flat message passing process. Even if such a simple approach can theoretically efficiently encode topological information combined with attribute distributions, it is always relevant to design a neuron architecture which enforces the learning of some desired property.

From our point of view, the "minimal" GNN formulation corresponds to the GNN algorithm defined in section 6.2, in which the $AGGREGATE^{(l)}_{graph}$ and $COMBINE^{graph}$ are simply the sum function, and $AGGREGATE^{(l)}_{node}$ is the standard update function:

$$\mathbf{h}_i^{(l+1)} = \sigma(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}(i)} \mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)})$$

where $\mathbf{h}_j^{(l)}$ is the vector representation of node j at step l, $\mathcal{N}(i)$ is the 1-hop neighbourhood of node i, and $(\mathbf{W}_0^{(l)}, \mathbf{W}_1^{(l)})$ are learnable weights.

In practice, if the architecture of the GNN itself is essential, fine modifications proposed in section 6.2 of the minimal GNN should be effective. Alternatively, if the fine-tuning of the model parameters via backpropagation is a paramount, the performance for the minimal GNN should go a long way, since a simple neuron network architecture should already perform the best.

Preliminary work

We initially tested the prominence of the graph neuron network architecture on several gold standard chemoinformatics datasets gathered in the MolNet collection (cf. section 2.3). We first compared the minimal GNN and an SVM-based model (baseline) performances via a 5-fold nested-cross validation.

We used the RDKit Python library to extract Morgan fingerprints (analogue of ECFP4 structural fingerprints), and considered the Tanimoto similarity measure to build a positive-definite similarity matrix between molecules, based on these fingerprints.

Regarding the original description of atoms required by graph neuron networks, the work of Coley et al. [298] showed that considering diverse structural and physicochemical atom features results in improved performances. Therefore, we used RDKit to define the following atom descriptors (see Coley et al. [298] for a detailed description): "atom identity", "number of hydrogens", "number of heavy neighbours", "formal charge", "is in ring", "is aromatic", "polar surface area", "partial charge", "chirality tag".

The entire neuron architecture comprises an FNN on the graph-level representation learnt by the GNN. We implemented this neuron network with the TensorFlow library. Furthermore, we identified the best hyper-parameters by recording the performance on a set of train/validation/test data splits for each dataset, and manually exploring hyper-parameter spaces, since a grid-search or random-search would have required too much computation time. More precisely, we separately optimised a set of architecture hyper-parameters, a set of learning hyper-parameters, and a set of regularisation hyper-parameters (see appendix A.8.4 for a complete description of these hyper-parameters). In the following, we enumerate the optimised hyper-parameters and the range of values that we considered. The architecture parameters are the dimension of the latent representation, or equivalently the number of convolutional filters ($[10, 1000]$), the number of update steps or equivalently the number of convolutional layers ($[\![2, 5]\!]$), and the number of neurons is the last hidden fully connected layers which processes the graph-level embedding ($[10, 1000]$). The learning parameters are the batch size ($[1, 100]$), the initial learning rate ($[10^{-5}, 10^{-2}]$), and the learning rate decay factor ($[0.8, 0.99]$). The regularisation parameters are the probability of dropout ($[0., 0.9]$) and weight decay ($[10^{-4}, 10^{-1}]$). We set the number of epochs to 100, and we also considered early stopping such that the training stops if the performance (ROCAUC score for classification tasks or MSE for regression tasks) on the validation set does not increase in ten successive epochs. In practice, the training was always stopped early.

For the learning hyper-parameters, we observed that the convergence of the network heavily depended on the initial learning rate, which must be set between 10^{-4} and 10^{-3}. Ultimately, we set the batch size to 10, the learning rate decay factor to 0.9, the number of convolutional filters and number of neurons in the final hidden layer to 100, the number of convolutional layers to 3, the dropout probability to 0.3 and the weight decay to 0. Overall, we found that, for all hyper-parameters, a wide range of values around those mentioned lead to the same best performance.

Indeed, the training is not sensitive to small variations of hyper-parameters as long as the number of trainable weights (i.e. width and depth) of the network is large and monitored enough (i.e. limited and regularised).

Table 8.1 shows the mean performance (and its standard deviation), for the membrane permeability dataset and some of the Tox21 prediction tasks. In the case of the Tox21 datasets, we used the AUPR score to measure the performance, and we used the MSE for the membrane permeability dataset. The performances are displayed in the format "mean score±score standard deviation".

	Tox21: 0	Tox21: 1	Tox21: 2	Tox21: 3	Membrane Permeability
SVM	0.7413 ± 0.0941	0.8007 ± 0.0957	0.8498 ± 0.0311	0.7673 ± 0.0882	0.2298 ± 0.0745
minimal GNN	0.7357 ± 0.1004	0.7457 ± 0.1275	0.8343 ± 0.0351	0.7352 ± 0.0792	0.2761 ± 0.1318

Table 8.1. AUPR (MSE for the membrane permeability dataset) performance on some benchmark chemoinformatics prediction tasks

The results reported in Table 8.1 are typical of those we obtained on the other datasets of the MolNet collection. In particular, the minimal GNN is competitive and reaches slightly lower performance than ECFP-Tanimoto SVM. Note that similar observations have been reported with other standard formulations of GNN [308].

Again, from our point of view, the weak point in the minimal formulation of GNN seemed to be the sum operator used to build a graph-level representation from the learnt representations of each atom. Indeed, this summation captures the distribution of features over the atoms in a single value and without prioritising relevant atoms. Therefore we tested several strategies to enhance the graph-level representation.

First, we tested the concatenation of the graph-level representations $\mathbf{m}^{(l)}$ built after each update steps to build the final graph-level representation $\mathbf{m}$, in order to guide the network to encode molecules with information of substructures of various sizes. We refer to this concatenation as jumping connections, similarly to [323], since it is equivalent to connect the final graph-level representation to the output of each hidden layers of the graph encoder.

Second, we alternatively defined the graph-level embedding by the deciles of the distribution of each feature in the learnt representation of the atoms. Thus, if the dimension of the nodes' latent representation is d, the dimension of the graph-level representation is $10 \times d$. This strategy encodes the distribution of each feature of the node latent representation.

Third, we proposed a self-attention mechanism, similar to [312] introduced in section 6.2, to build the graph-level representation in order to help to select relevant contribution of each atom. In this formulation, the graph-level embedding $\mathbf{m}$ is defined based on the final atom-level representation $\mathbf{h}_i^{(L)}$ for each node i ($\odot$ stands for element-wise multiplication):

$$\mathbf{m} = \sum_{i \in \mathcal{V}} \sigma(\mathbf{W}_{att} * \mathbf{h}_i^{(L)} + \mathbf{b}_{att}) \odot Relu(\mathbf{W} * \mathbf{h}_i^{(L)} + \mathbf{b})$$

	Tox21: 0	Tox21: 3	Membrane Permeability
minimal GNN	$\mathbf{0.7476 \pm 0.0646}$	0.7168 ± 0.0685	$\mathbf{0.2535 \pm 0.0511}$
GNN with jumping connections	0.7385 ± 0.085	$\mathbf{0.7316 \pm 0.0855}$	0.3006 ± 0.2031
GNN with decile-based graph-level representation	0.727 ± 0.0706	0.7199 ± 0.0528	0.2826 ± 0.1284
GNN with self attention-based graph-level representation	0.7297 ± 0.0692	0.7176 ± 0.0549	0.3149 ± 0.1218

Table 8.2. AUPR (MSE for the membrane permeability dataset) performance on some chemoinformatics tasks for some GNN formulations

We can summarise the results in table 8.2 by observing that the proposed modifications did not consistently improve the performance of the minimal GNN for these chemoinformatic tasks. Although the differences are not statistically significant, we even notice that the minimal GNN performs better than its proposed modifications for some tasks. This means that the proposed modifications potentially make the training more difficult by making the optimisation landscape more rugged.

This preliminary work was undertaken before a colossal amount of studies on GNN were published in the last two years. They proposed a variety of modifications to GNN that we reviewed in section 6.2. Furthermore, some of the modifications that we proposed here can be found in the recent literature [308, 323]. However, our preliminary work could not conclude on the efficiency of more sophisticated graph neuron network architectures.

Therefore, in the last section 8.2, we examine the interest of the most relevant improvements on GCNs in the context of DTI prediction.

Before that, we explore , in another preliminary work, another promising direction to enhance end-to-end feature extraction.

8.1.2 Multitask learning for end-to-end feature extraction

Since deep neuron networks require a large amount of data to reach their full potential, a promising direction of research is to consider multitask learning (in particular transfer learning).

In addition to trigger implicit data augmentation, multitask learning is expected to prevent artificial neuron networks from overfitting. Indeed, learning a representation on multiple tasks may lead to a molecular representation which generalises better to other tasks and, in general, to new unseen data.

However, considering multitask neuron networks brings its own issues. For instance, we must question which tasks are intended to help each other, how to assess the closeness of tasks, and more generally, how to design a neuron architecture and a learning schedule to efficiently share knowledge between loosely-related

prediction tasks.

For example, even though molecular solubility might not be related to some molecular biological properties, such as protein interactions, we can expect that positive transfer learning can occur between the PubChem bio-assay data and the DrugBank data, because they may rely on common mechanisms since they both concern molecular bio-activities.

To get a bigger picture, we introduce multitask learning for neuron networks in appendix A.8.3, and for the general case in appendix A.7.

Preliminary work.

In order to test the prominence of multitask learning and get experience in data-driven feature extraction on protein sequences, we investigated multitask learning for a collection of tasks corresponding to the prediction of some protein property.

Among the considered datasets, we first discuss (i) the "CellLoc" dataset [279], labelling 5917 proteins with one of the twelve sub-cellular localisation. Also, we consider (ii) the "SCOPe" dataset [126] processed from the SCOPe database and classifying 13963 proteins over seven types of 3D fold (alpha proteins, beta proteins, alpha and beta proteins, multi-domain proteins, membrane and cell surface proteins and peptides, small proteins). Finally, we use (iii) the "Secondary Structure" dataset [279] which is composed of 4388 proteins in which all amino acids are assigned to one of the eight possible protein secondary structures. They are all imbalanced multi-class classification tasks, but the two former consider proteins as a whole, whereas the latter labels individual amino acids. Note that the SCOPe and secondary structure prediction tasks are explicitly related since they both characterise protein folding, but at different level (respectively, at the sequence-level and amino acid-level).

We first compared singletask neuron network-based encoders and SVM performances via 5-fold nested cross-validation. We used the local alignment kernel to build the protein kernels (centred and normalised) as mentioned in section 2.2. Furthermore, we needed to define original descriptors for amino acids, in order to process the sequence with convolutional or recurrent neuron networks. We described each amino acid by its type in a one-hot encoding fashion. We implemented convolutional neuron layers and LSTM with the TensorFlow library. Again, we separately optimised three sets of hyper-parameters (architecture, learning and regularisation hyper-parameters) by recording the performance via nested cross-validation on a few folds for each dataset, such that we manually explored the hyper-parameter space.

We tested two types of neuron architecture on protein sequences: stacked convolutional layers (CNN) and a convolutional layer stacked with a biLSTM layer (conv-biLSTM), as motivated in section 6.1. In the following, we detail the optimised hyper-parameters, and the range of values we considered are given in parentheses. The architecture parameters are the number of convolutional filters ($[10, 1000]$), the size of the convolutional filter receptive field ($[6, 12]$), the convolution stride ($[2, 6]$), the number of convolutional layers for the CNN architecture ($[\![2, 5]\!]$), and the number of neurons in the last hidden fully connected layer which processes the sequence-level embedding ($[10, 1000]$). The learning and regularisation hyper-parameter spaces

were explored identically to the previous section.

After parameter tuning, we did not observe any statistically significant difference between the performance of CNN and conv-biLSTM on the three datasets. Again, we also observed that the performance is not sensitive to small variations of hyper-parameters (i.e. the performance does not significantly vary when hyper-parameters remain in the same order of magnitude). Surprisingly, small CNN networks (about ten filters and two layers) already reached relatively high performance. Finally, we chose the CNN architecture as protein sequence encoder, since a conv-biLSTM requires more time to train. In addition, we set the batch size to 5, the learning rate decay factor to 0.99, the number of filters to 20 and 40 respectively in the first and second convolutional layers, the filter size to 6, the stride to 3, the dropout to 0.2 and the weight decay to 0.01.

To investigate multitask learning, we also tested training the protein encoder (CNN) jointly on the SCOPe and secondary structure datasets (following the hard-sharing strategy, as discussed in appendix A.8.3).

In table 8.3 we display the mean performance (and its standard deviation) on the three considered datasets. The performances are displayed in the format "mean score±score standard deviation".

	CellLoc	SCOPe	Secondary Structure
SVM (baseline)	ROCAUC: 0.9728 ± 0.0615 AUPR: 0.8448 ± 0.0238	ROCAUC: 0.9609 ± 0.00476 AUPR: 0.8683 ± 0.0095	NA
CNN singletask	ROCAUC: 0.9281 ± 0.0088 AUPR: 0.6313 ± 0.0041	ROCAUC: 0.8635 ± 0.0121 AUPR: 0.5928 ± 0.0242	ROCAUC: 0.7652 ± 0.0055 AUPR: 0.344 ± 0.002
CNN multitask	NA	ROCAUC: 0.8776 ± 0.0022 AUPR: 0.6143 ± 0.0123	ROCAUC: 0.7597 ± 0.0037 AUPR: 0.3328 ± 0.0023

Table 8.3. Performances obtained on three benchmark protein datasets

First, the ROCAUC scores appear too high to fairly compare methods, and therefore, we focus on the AUPR scores. The SVM approach significantly outperforms the CNN approach. This could be expected, since we could not find in the literature any study showing competitive performance of deep learning models compared to methods based on state-of-the-art protein kernels for such small datasets. Nevertheless, the performance reached by the SVM-based method is extremely high. We think it is due to the fact that the LAkernel is particularly suited for these prediction task. Indeed, this kernel compares very efficiently proteins from the evolutionary point of view, which is particularly suited for predicting secondary structure of protein folding and sub-cellular localisation.

Furthermore, we could expect that deep learning methods may benefit from multitask learning, since training a neuron network encoder on multiple tasks is an implicit data augmentation. Moreover, recall that the SCOPe and secondary structure prediction tasks are explicitly related since they both characterise protein folding, but at different level. However, the co-training of the protein sequence encoder on the SCOPe and secondary structure datasets did not yield to an improvement of performance, although these two tasks are strongly related. A similar observation was made in the NLP field [353]. Parameter hard-sharing multitask

is likely to be too restrictive and therefore, is probably efficient in few specific cases.

That being said, task-specific layers are too parameter-inefficient, so that sparse sharing or parameter-efficient modules are necessary (cf appendix A.8.3). At the same time, training all tasks simultaneously might not be the best choice for loosely-related tasks. More adaptive training strategies, such as curriculum learning introduced in the next section, seem more promising in this situation.

With the intention to boost data-driven approaches for virtual screening in the chemogenomics framework, the two preliminary works, reported in this first section, blazed a trail.

Therefore, we investigate in the next section the efficiency of curriculum learning and various relevant modifications of the neuron network architecture for chemogenomics.

8.2 Improving end-to-end extracted representations for chemogenomics

In this section, we explicitly investigate end-to-end data-driven approaches for chemogenomics.

The data-driven approach relies on the neuron architecture that we called "chemogenomic neuron network". Recall that it consists of an FNN taking as input the combination of molecular and protein data-driven representations extracted by end-to-end learnable molecular graph and protein sequence encoders.

8.2.1 Materials

We use the DrugBank database version 5.1.0 to build two druggable proteome-wide drug-target interaction prediction datasets. In the first dataset DBH, we only kept interactions including human proteins and their ligands, whereas in the second one $DBEC$, we only kept interactions including Escherichia Coli proteins and their ligands.

$DBEC$ is composed of 592 molecules targeting a total of 314 proteins, and including 874 protein-ligand interactions that correspond to the positive training pairs. DBH is composed of 4834 molecules targeting a total of 2561 proteins, and including 13070 protein-ligand interactions. In both datasets, the vast majority of the targets and drugs are involved in only one or two known interactions.

As we later examine curriculum learning to augment the performance on these datasets, we also consider a PubChem-based and the protein sub-cellular localisation datasets already introduced earlier.

We pre-processed the PubChem datasets included in the MolNet dataset collection (see section 2.3.2) to keep the bioassays composed of a maximum of common molecules. The resulting dataset consists in a total of $439,863$ unique molecules and 90 binary classification tasks (bioassays). For each bioassay, molecules are labelled as active, inactive or unknown. When a molecule is labelled as unknown for a bioassay (because it has not been experimentally tested), the prediction error for the corresponding task and molecule is set to zero.

The sub-cellular localisation dataset [279] is a multi-class classification problem, labelling 5917 human proteins with one of the twelve sub-cellular localisation considered classes.

8.2.2 Methods

Methods: the chemogenomic neuron network

Based on our preliminary works, we used stacked 1D convolutional layers as the protein sequence encoder, in which each amino acid is one-hot encoded according to the amino acid type.

To encode molecular graphs, we used graph neuron network as defined in section 6.2. We first used the minimal formulation of graph neuron networks defined earlier in this chapter. In a second step, we are to test promising architecture modifications. To encode atoms and bonds of molecular graph, we again used RDKit. We described atoms with the following features (refer to Coley et al. work [298] for further details): "atom identity", "number of hydrogens", "number of heavy neighbours", "formal charge", "is in ring", "is aromatic", "polar surface area", "partial charge", "chirality tag". The chemical bond description contains: "bond type", "is in ring", "is aromatic", "is conjugated".

As for the preliminary work, we optimised three sets of hyper-parameters (architecture, learning and regularisation hyper-parameters) by recording the performance on multiple train/validation/test data splits to explore hyper-parameter spaces manually, since a grid-search or random-search would have required too much calculation time. In the following, we detail the optimised hyper-parameters and the range of values we considered are given in parentheses.

The architecture hyper-parameters are the number of filters both for the graph and sequence encoders ($[10, 1000]$), the number of layers for both the protein and molecule encoders ($[\![2, 5]\!]$), the convolutional filter size for the sequence encoder ($[6, 12]$), the convolutional stride for the sequence encoder ($[2, 6]$), and the number of neurons in the last hidden fully connected layer which processes the pairwise-level encoding ($[10, 1000]$).

The learning hyper-parameters are the batch size ($([1, 100])$), initial learning rate ($[10^{-5}, 10^{-2}]$) and learning rate decay factor ($[0.8, 0.99]$).

The regularisation hyper-parameters are the probability of dropout ($[0., 0.9]$) and the weight decay ($[10^{-4}, 0.1]$). We set the number of epochs to 100, and we also considered early stopping such that training stops if the performance (AUPR score) on the validation set does not increase in ten successive epochs. In practice, training was always stopped early.

After exploring the hyper-parameter spaces, we set, for both datasets, the number of filters to 100, the number of convolutional layer to 3, the stride to 3, the filter size to 8, the weight decay and dropout to 0, batch size to 20, initial learning rate to 10^{-3}, learning rate decay factor to 0.9 and number of neurons in the last prediction layer to 100. Again, we also found that training was not sensitive to relatively small variations of hyper-parameters. In particular, adding a small quantity of dropout (until 0.5) leads progressively to a longer training and a loss of training performance, whereas it did not significantly improve the performance on the validation and test data.

Furthermore, we repeated training and testing procedures of GNN three times for $DBEC$ and two times for DBH (because the training was computationally too expensive), and kept only the best achieved performance. Indeed, we noticed that the performance of the chemogenomic neuron network varied significantly depending on the initial random seed.

Reference state-of-the-art methods

Based on the conclusions of the review of the literature in chapters 3 and 7, we compared the chemogenomic neuron network to three state-of-the-art approaches considered as baselines: feed-forward neuron networks -FNN- (see appendix A.8.2), SVM (see appendix A.5) and the *NRLMF* approach (see section 3.2.2). We used the Keras library to implement the neuron network models.

The FNN approach requires numerical feature vectors as input. We extracted 1021-dimensional structural Morgan fingerprint vectors (analogue of ECFP4) with the RDKit library for each molecule. We extracted 1920-dimensional feature vectors (sequence composition, topological and physical properties) for each protein sequence via the protR library (see section 2.1).

We manually optimised the hyper-parameters of FNN (number of layers, number of neurons in each layer, learning and regularisation hyper-parameters) on a fixed train/validation/test data split. Notably, we did spend little time in fine-tuning these hyper-parameters, compare to those of the chemogenomic neuron network.

Finally, we set the number of neurons to 2000, 1000 and 100 for the successive three stacked fully connected layers, weight decay and dropout probability to 0 (no regularisation), the initial learning rate to 10^{-3}, the batch size to 100, the learning rate decay to 0.9, the number of epochs to 100 and early stopping patience to 20. The early stopping patience is the number of successive epochs, for which the performance on the validation data decreased, required to stop training.

The SVM and *NRLMF* approaches take as input molecule and protein kernel matrices. We used the local alignment kernel, presented in section 2.2, to build the protein kernels (centred and normalised). We used the RDKit Python library to extract Morgan fingerprints (analogue of ECFP4 structural fingerprints, see Section 2.1) and consider the Tanimoto metric to build the kernel matrix on molecules based on these fingerprints. The (protein, molecule) pairwise kernel is defined as the Kronecker product of the protein and molecule kernels as motivated in chapter 4. The regularisation hyper-parameter of SVM was automatically optimised when assessing the performance via 5-fold nested cross-validation. We set the five hyper-parameters of *NRLMF* to their default value, as stated in the original paper [136].

Methods: evaluation procedure

We evaluated the performance by 5-fold nested cross-validation. Although we recorded the ROCAUC, F1-score, accuracy and AUPR scores for each test fold, we predominantly refer to the AUPR. The AUPR is

considered as a more significant quality measure than the ROCAUC when negative interactions are in fact unknown interactions (which is the case in DTI prediction), and when there are more negative samples than positive ones in the test set (which is also the case in DTI prediction).

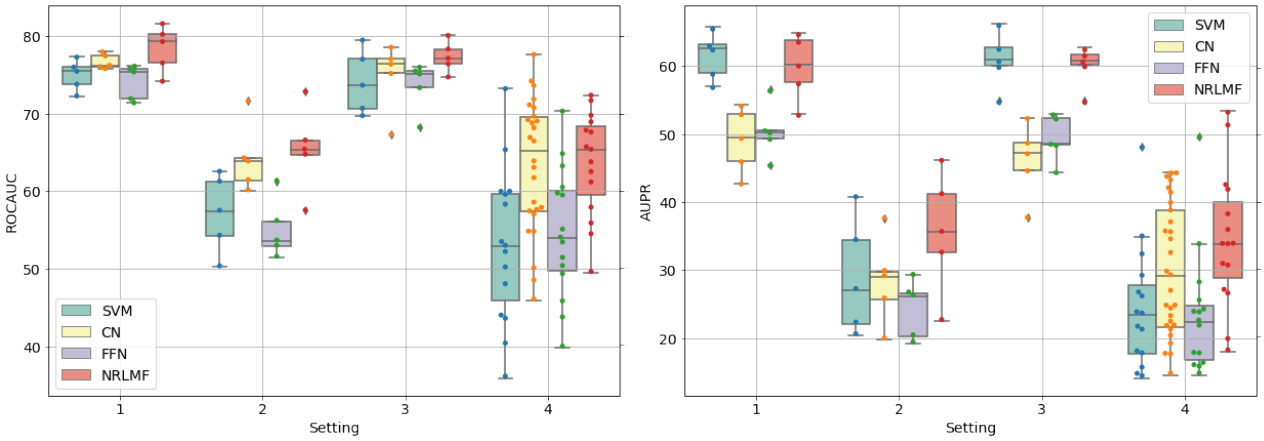
Indeed, it emphasises on the recovery of the positive samples and penalises the presence of false positive examples among the best-ranked samples. Therefore, we used AUPR as the evaluation metric for the hyper-parameters optimisation and in the early stopping procedure used in deep learning approaches.

We also evaluated each method in various settings. First, we considered the S_1, S_2, S_3, and S_4 settings that correspond to random, orphan ligand, orphan protein and double orphan prediction scenarios, as in section 4 (motivations are detailed in section 1.2 and 4, as well as in Pahikkala et al. paper [168]). Note that the folds of S_4 were built by intersecting those of S_2 and S_3, so that there are 25 folds in the S_4 setting instead of 5. Independently, we considered scenarios in which the "positive:negative" sample ratio in the test set is 1:1, 1:2 or 1:5. These scenarios evaluate whether methods remain robust when predicting interactions at the proteome level (i.e. when there are much more non-interacting pairs than interacting ones).

8.2.3 Results and discussion: comparison of the chemogenomic neuron network to baselines

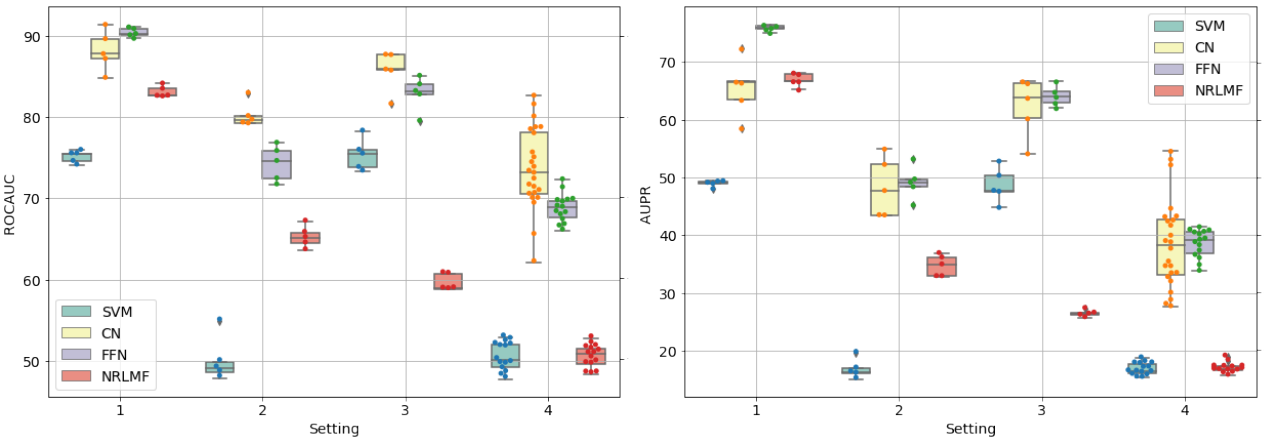
We first compared the performance of standard chemogenomic neuron network (CN) to those of the considered baselines. Performances are reported in tables 8.4, 8.5, 8.6 and 8.7. The performances are displayed in the format "mean score±score standard deviation".

Also, figures 8.1 and 8.2 displays the ROCAUC and AUPR performance obtained on the $DBEC$ and DBH datasets for a "positive:negative" test samples ratio set to $1:5$.



(a) ROCAUC scores (b) AUPR scores

Figure 8.1. ROCAUC and AUPR performance on *DBEC* for the four settings with a test and train sample positive:negative ratio set to 1:5



(a) ROCAUC scores (b) AUPR scores

Figure 8.2. ROCAUC and AUPR performance on *DBH* for the four settings with a test and train sample positive:negative ratio set to 1:5

	raw (S_1)			orphan proteins (S_2)			orphan molecules (S_3)			double orphan (S_4)		
	1:1	1:2	1:5	1:1	1:2	1:5	1:1	1:2	1:5	1:1	1:2	1:5
SVM	**75.14 ± 1.7**	74.95 ± 2.04	74.99 ± 1.76	54.97 ± 4.33	56.5 ± 4.29	56.67 ± 4.44	74.03 ± 3.8	73.89 ± 3.78	74.14 ± 3.72	53.84 ± 8.86	54.46 ± 8.43	53.06 ± 9.63
NRLMF	**78.8 ± 3.68**	**78.88 ± 2.73**	**78.41 ± 2.69**	**65.05 ± 7.1**	**64.96 ± 6.32**	**65.4 ± 4.9**	**77.19 ± 1.65**	**77.21 ± 2.06**	**77.38 ± 1.81**	**65.2 ± 7.26**	**64.59 ± 6.26**	**63.61 ± 6.51**
FNN	**75.92 ± 1.17**	74.63 ± 1.04	74.88 ± 1.52	**58.25 ± 5.34**	55.12 ± 4.67	54.89 ± 4.98	74.67 ± 2.38	**75.52 ± 2.4**	73.96 ± 3.15	56.03 ± 9.22	55.91 ± 8.07	54.68 ± 8.12
chemogenomic neuron network	**77.1 ± 1.83**	**76.5 ± 0.89**	**76.74 ± 0.85**	**61.9 ± 4.41**	**59.64 ± 8.55**	**64.25 ± 4.0**	73.49 ± 3.99	74.04 ± 4.91	74.91 ± 3.98	**59.23 ± 9.5**	**58.4 ± 8.74**	**58.62 ± 9.1**

Table 8.4. ROCAUC scores on the *DBEC* dataset in the four settings and for a test sample positive:negative ratios in {1:1, 1:2, 1:5}

	raw (S_1)			orphan proteins (S_2)			orphan molecules (S_3)			double orphan (S_4)		
	1:1	1:2	1:5	1:1	1:2	1:5	1:1	1:2	1:5	1:1	1:2	1:5
SVM	**81.75 ± 1.75**	**73.25 ± 2.4**	**61.55 ± 3.11**	59.62 ± 5.55	**45.89 ± 6.73**	**28.95 ± 7.66**	**80.79 ± 2.37**	**72.02 ± 3.06**	**60.96 ± 3.73**	58.52 ± 9.21	43.37 ± 9.15	24.4 ± 8.8
NRLMF	**82.56 ± 2.82**	**73.89 ± 2.98**	**59.89 ± 4.35**	**67.56 ± 7.04**	**53.18 ± 8.12**	**35.62 ± 8.07**	**81.49 ± 1.48**	**72.35 ± 2.2**	**60.06 ± 2.73**	**67.93 ± 8.63**	**52.97 ± 8.7**	**34.5 ± 9.75**
FNN	**79.82 ± 0.98**	68.07 ± 2.35	51.55 ± 2.55	**61.92 ± 5.11**	42.24 ± 5.27	24.26 ± 3.91	78.57 ± 2.22	68.14 ± 3.43	49.34 ± 3.11	**60.04 ± 8.83**	43.55 ± 9.84	22.97 ± 6.59
chemogenomic neuron network	79.04 ± 2.24	66.5 ± 2.51	49.08 ± 4.31	**60.79 ± 6.24**	**45.11 ± 8.65**	**28.38 ± 5.81**	76.23 ± 2.23	64.61 ± 4.78	46.14 ± 4.92	**61.97 ± 8.72**	**45.81 ± 8.5**	**27.0 ± 6.96**

Table 8.5. AUPR scores on the *DBEC* dataset in the four settings and for a test sample positive:negative ratios in {1:1, 1:2, 1:5}

	raw (S_1)			orphan proteins (S_2)			orphan molecules (S_3)			double orphan (S_4)		
	1:1	1:2	1:5	1:1	1:2	1:5	1:1	1:2	1:5	1:1	1:2	1:5
SVM	74.6 ± 0.57	74.6 ± 0.55	74.62 ± 0.54	50.26 ± 3.05	49.88 ± 3.06	49.88 ± 2.95	75.16 ± 1.51	75.28 ± 1.76	75.14 ± 1.69	49.79 ± 1.61	50.25 ± 1.84	50.5 ± 1.68
NRLMF	82.98 ± 0.55	83.06 ± 0.65	83.12 ± 0.63	65.18 ± 1.36	65.18 ± 1.28	65.23 ± 1.2	59.49 ± 0.96	59.56 ± 0.81	59.6 ± 0.93	50.37 ± 1.48	50.29 ± 1.55	50.47 ± 1.35
FNN	**90.39 ± 0.42**	**90.65 ± 0.46**	**90.45 ± 0.52**	74.74 ± 0.4	74.08 ± 2.29	74.25 ± 1.96	83.06 ± 1.48	**83.52 ± 1.52**	82.9 ± 1.8	**69.12 ± 2.3**	**69.17 ± 1.99**	68.79 ± 1.69
chemogenomic neuron network	88.67 ± 1.59	88.21 ± 1.47	88.21 ± 2.23	**81.5 ± 1.47**	**80.83 ± 2.7**	**80.28 ± 1.4**	**85.76 ± 1.5**	**85.4 ± 2.02**	**85.76 ± 2.23**	**73.34 ± 5.89**	**71.21 ± 5.53**	**73.68 ± 4.87**

Table 8.6. ROCAUC scores on the *DBH* dataset in the four settings and for a test sample positive:negative ratios in {1:1, 1:2, 1:5}

	raw (S_1)			orphan proteins (S_2)			orphan molecules (S_3)			double orphan (S_4)		
	1:1	1:2	1:5	1:1	1:2	1:5	1:1	1:2	1:5	1:1	1:2	1:5
SVM	78.34 ± 0.44	66.85 ± 0.68	48.99 ± 0.5	50.35 ± 2.64	33.46 ± 2.33	16.82 ± 1.54	78.2 ± 1.77	66.86 ± 2.12	48.66 ± 2.74	49.84 ± 1.7	33.26 ± 1.64	16.81 ± 1.02
NRLMF	86.14 ± 0.45	78.73 ± 0.75	66.93 ± 1.06	68.19 ± 1.55	53.68 ± 1.72	34.75 ± 1.65	61.88 ± 0.99	45.93 ± 0.72	26.42 ± 0.52	50.45 ± 1.31	33.71 ± 1.25	16.99 ± 0.8
FNN	**91.45 ± 0.45**	**86.5 ± 0.44**	**75.91 ± 0.49**	78.16 ± 0.44	66.02 ± 1.86	**49.13 ± 2.58**	**85.69 ± 1.05**	**78.15 ± 1.18**	**64.06 ± 1.6**	**71.46 ± 2.02**	**57.93 ± 2.7**	**38.61 ± 2.29**
chemogenomic neuron network	88.74 ± 1.68	80.29 ± 2.65	65.46 ± 4.53	**81.82 ± 1.9**	**70.39 ± 3.66**	**47.68 ± 4.38**	**86.0 ± 1.85**	**77.02 ± 3.41**	62.22 ± 4.66	**72.26 ± 6.42**	**55.98 ± 7.67**	**38.48 ± 7.5**

Table 8.7. AUPR scores on the *DBH* dataset in the four settings and for a test sample positive:negative ratios in {1:1, 1:2, 1:5}

To summarise the results, we can first notice that the "best" method depends on the dataset and the setting. On *DBEC*, the four methods are competitive, but the *NRLMF* method performs slightly better in all cases, and even significantly better in the S2 and S4 settings. Overall, since *DBEC* is a relatively small dataset, it favours to the *NRLMF* and SVM-based approaches (not involving deep learning), which perform the best in the various settings.

On the larger dataset *DBH*, deep learning-based approaches significantly outperform *NRLMF* and SVM. FNN performs the best in the S_1 setting, whereas CN seems slightly better for the S_2, S_3 and S_4 settings. Moreover, the SVM-based approach is not robust on the S_2 and S_4 settings for both datasets. Most of all, on the *DBH* dataset, the SVM-based approach fails to leverage the information in the training data, as its performance is that of a random classifier.

Overall, CN and FNN appear as the best default choice, since they seem the most robust to a panel of settings. In particular, CN outperforms other methods on the S_2, S_3 and S_4 settings for the *DBH* dataset.

Regarding the datasets, we observed that the performance reached on the *DBH* are 10 points higher than those reached on *DBEC*, which was expected since there are much more data (more information) to train the models in *DBH* than in *DBEC*.

Furthermore, all methods appear equivalently robust to an imbalanced test set. Indeed, when changing the "positive:negative" ratio in the test set from 1 to 5, all methods lost about twenty points of AUPR on S_1, and up to 30 points on S_2, S_3, and S_4.

The performance reached for a ratio of 1:5 humbles, while it is the most realistic scenario when evaluating the interactions at the proteome level. However, we expect that a relatively large number of drug-like molecules interact with more than one protein [24], whereas most of the molecules in the database have one or two known targets. This means that a significant amount of false predicted positives are expected to correspond to real interactions.

Regarding the settings, the best scores are obtained for S_1, the worst for S_4, and the scores obtained in the S_2 and S_3 datasets are intermediate between those observed on S_1 and S_4. This was to be expected since no pairs of the training set contain the protein or the ligand of the tested pair to guide the predictions in S_4, whereas the models can rely on training pairs containing either the same proteins in S_2 or the same ligands in S_3.

The loss of performance between the raw (S_1) and the double orphan (S_4) settings is about 10 points of ROCAUC and 20 points of AUPR. More importantly, the prediction on S_4 varies more depending on the test folds than in the other settings. This can be understood as a consequence of the small size of the test set in the S_4 setting (any test fold is $1/25^{th}$ of the total amount of data whereas they represent $1/5^{th}$ in the other settings), resulting into test sets of various difficulties.

Interestingly, the performance reached by all models is better for S_3 than for S_2, which suggests that predicting ligand for new protein targets is more laborious than predicting targets for new compounds. Therefore, there is relatively more loss of information for orphan proteins than for orphan molecules. We think it is due to the sampling bias discussed in section 1.3.

To conclude this first phase of the current study, note that we also considered several "positive:negative" samples ratios in the train set, as all approaches may benefit from more negative training data. Recall that such "negative" training data are unknown interactions, so that we must not use of all of them to train the models. Moreover, it would be hardly computationally tractable.

Before discussing our results, note that we did not weight the sample loss to correct for the imbalance proportions of labels in the training set in the case of deep learning-based approaches. Indeed, we found that it damaged their final performance on the test set.

The SVM and *NRLMF* methods did not benefit from an increasing number of negative samples in the train set, whereas deep learning-based methods, as expected, did strongly benefit from an increasing number of training samples. More precisely, increasing the number of negative samples in the training set, up to five times the number of positive sample, increased the performance of CN and FNN by about ten points of AUPR (only two points of ROCAUC) for the *DBH* dataset, and by twenty points of AUPR (ten points of ROCAUC) for the smaller *DBEC* dataset. We also observed that CN benefits slightly less than FNN from an increasing number of negative in the training set. Overall, we tested four "positive:negative" train sample ratios (1:1, 1:2, 1:5 and 1:10). The performance of CN and FNN did increase between a ratio of 1:5 and 1:10 in some cases, meaning the neuron network-based models may still benefit more from more training negative samples.

In the following, we tested several changes in the "standard" chemogenomic neuron network discussed in the two preliminary works described earlier in this chapter.

For each of the modifications, we explored the associated hyper-parameter spaces to find the best reachable solution. To this end, we measured 5 times the performance on a single train/validation/test split for the S_1, S_2, S_3 and S_4 settings, since an evaluation of all of these modification by nested cross-validation was neither computationally feasible for us, nor relevant. The mean and standard deviation of the performance we obtained are reported in Table 8.8, and are discussed below. The performances are displayed in the format "mean score±score standard deviation".

8.2.4 Evaluation of graph convolutional network architectures

In this section, we investigate promising modifications to the standard GNN formulation for chemogenomic virtual screening. We only considered the modifications that appeared the most relevant, both in terms of intuition and performance improvements, as discussed in section 6.2.

Among proposed $AGGREGATE^{(l)}_{node}$ functions, the GAT function proposed by Velickovic et al. [295] (see section 6.2) seemed the most promising to us. Indeed, it resulted in significant performance improvement, and is intuitively relevant since it is intended to prioritise relevant neighbours via a multi-head attention mechanism at each node update.

We also assayed the "wbond" function that takes into account the bond attributes for the aggregation, in order to test whether bond attributes can bring representation power and flexibility. Here, the "wbond"

$AGGREGATE^{(l)}_{node}$ function refers to (as in Coley et al. [298]):

$$\mathbf{h}_i^{(l+1)} = \sigma(\mathbf{W}_0^{(l)} \cdot \mathbf{h}_i^{(l)} + \sum_{j \in \mathcal{N}(i)} \mathbf{W}_1^{(l)} \cdot [\mathbf{h}_j^{(l)}, \mathbf{x}_{ij}])$$

where $\mathbf{h}_i^{(l+1)}$ is the representation of atom i at step $l+1$, $[\mathbf{h}_j^{(l)}, \mathbf{x}_{ij}]$ is the concatenation of the representation $\mathbf{h}_j^{(l)}$ of atom i at step l and the bond attribute vector $\mathbf{x}_{ij}$ between atoms i and j, and $\mathbf{W}_0^{(l)}, \mathbf{W}_1^{(l)}$ are learnable parameters.

For both of them, we varied the number of convolutional filters (in {10,20,50,100,1000}) and the keeping probability of dropout (in {0,0.3,0.6}). We also tested several numbers of attention heads (in {1,5,10}) for the GAT function. Besides, we used the original implementation provided by the authors of the GAT method.

None of these two modifications lead to significant performance improvement for the four settings, as reported in Table 8.8.

Although the GAT function led to substantial improvements in gold standard datasets of graph representation learning (see section 6.2), it does not in chemogenomics. This may be due to the the fact that molecular graphs are small, and that all atoms of molecules are informative when learning an end-to-end molecule encoding. In the case of the "wbond" function, the standard GNN may already leverage bond information based on the nodes attributes and degrees (i.e., based on the topology of the graph).

Independently, we tested several options to replace the summation, in the minimal GNN, that builds the graph-level representation from node representations. From the simplest to the most sophisticated, we tested the max function [350, 323], jumping connections [323] and hierarchical differential pooling [320, 321, 301], all discussed in Section 6.2.

When used as the $AGGREGATE^{(l)}_{graph}$ function, the sum function captures the distribution of node features in a single value. Alternatively, the max function is expected to capture representative elements in the node features' distributions, with no additional parameters to learn. This is relevant for molecular bio-activity prediction, as bio-activity may rely on representative elements such as pharmacophores or functional groups. The chemogenomic neuron network, whose GNN's $AGGREGATE^{(l)}_{graph}$ function is set to the max function, is mentioned as "CN-MaxAgg" in Table 8.8. Note that even if the sum function can theoretically capture representative elements, in particular when the dimension of the feature vector is high, the max function enforces capturing representative elements, which may improve the performance.

Regarding the $COMBINE^{graph}$ function, we tested to concatenate graph-level representations after each update step $\mathbf{m}^{(l)}$ in order to build the final graph-level representation $\mathbf{m}$. We expect that it could help learning by skipping deleterious layers and by representing molecules via substructures of different sizes. The chemogenomic neuron network, whose GNN's $COMBINE^{graph}$ function is set to the concatenation function, is mentioned as "CN-ConcatCombine" in Table 8.8. Note that the sum function can, theoretically, adopt the same behaviour if the dimension of the feature vector is high enough, and if the network learns to store information in different locations in the feature vector at each update step.

The hierarchical pooling procedure introduces in section 6.2, which takes the role of both the $AGGREGATE^{(l)}_{graph}$ and $COMBINE^{graph}$ functions, seems particularly relevant as it is the analogue of the pooling operator

in image processing. Indeed, it reduces the spatial information step by step by differentially coarsening the input graph after each node update step, until the graph is reduced to a single node. The chemogenomic neuron network whose GNN is based on hierarchical pooling is mentioned as "CN-pool" in Table 8.8. Again, note that a naive sum over node feature vectors can mimic the same behaviour, but the hierarchical pooling procedure is expected to enforce learning the graph-level representation in a hierarchical fashion and, hence, to improve encoding.

For all of these modifications of the molecular graph encoder, we varied the quantity of dropout ({0,0.3,0.6}). In the case of hierarchical pooling, that we implemented based on Ying et al. [320], we also varied the dimension of the nodes embedding (i.e. the number of convolutional filters) and the quantity of dropout when building the soft assignment matrices.

None of these modifications led to substantial performance improvements for the four settings on the *DBEC* dataset.

Intuitively, this means that, in the case of small chemical compounds, the sum function has the same representation power than the max for the $AGGREGATE^{(l)}_{graph}$ function and concatenation for the $COMBINE^{graph}$ function. Therefore, in the case of small molecular graphs, the sum function can capture representative elements in the graph (as well as the max function) and can learn to store information of subgraphs of different sizes (as well as the concatenation function), if these strategies are relevant for DTI prediction. Regarding the hierarchical pooling, it seems that standard GNN can also leverage information of substructures of different sizes in a hierarchical manner. Indeed, the iteration of the nodes update is already providing neighbouring information to each node in a hierarchical manner.

8.2.5 Evaluation of protein sequence network architectures

Up to now, we only explored the modification of the neuron network architecture encoding molecular graphs. However, the performance bottleneck could be related to the protein sequence encoder rather than the molecular graph encoder.

Therefore, we examined a protein sequence encoder that consists of a biLSTM layer on top of a convolution layer, instead of stacked convolution layers. Indeed, the bi-LSTM can locally integrate the detected patterns by the convolutional layer to provide local contextual information, both forward and backwards, whereas stacked convolutional layers process protein sequences by detecting local patterns, patterns of patterns and so on.

We varied the dimension of the amino acid learnt representation ({10, 50, 100, 500}) and the amount of dropout ({0., 0.1, 0.3, 0.6}) to search for the best performance.

Finally, the best reached performance was identical to those obtained while using stacked convolutional layers (cf. table 8.8 at the row named "CN-biLSTM"). At this stage, we conclude that encoder architecture refinements does not appear as a promising direction to enhance DTI prediction with the chemogenomic framework.

8.2.6 Evaluation of the neuron architecture combining end-to-end extracted representations of molecules and proteins

We now propose to investigate the use of an attention mechanism to combine molecular graph and protein sequence representations, as in [348, 349], compared to a standard concatenation.

We first implemented the attention mechanism proposed by Tsubaki et al [349] which resulted in a significant decrease of performance for the four settings. This might be due to the fact that this attention mechanism is highly non-convex, since it relies on the dot product of the two learnt representations, resulting in a rugged optimisation landscape.

Therefore, we turned to a self-attention mechanism that compute the pairwise representation $\mathbf{h}_{pair}$ based on the concatenation $[\mathbf{h}_{prot}, \mathbf{h}_{mol}]$ of the protein $\mathbf{h}_{prot}$ and molecule $\mathbf{h}_{mol}$ learnt representations, following:

$$\mathbf{h}_{pair} = \sigma(\mathbf{W}_{att} \cdot [\mathbf{h}_{prot}, \mathbf{h}_{mol}] + \mathbf{b}_{att}) \odot [\mathbf{h}_{prot}, \mathbf{h}_{mol}]$$

. As reported in Table 8.8 (at the row named "CN-pairwiseAtt"), this did not result in performance improvement, meaning that such soft attention mechanism does no provide better representation power. It is most likely that the self-attention mask "$\sigma(\mathbf{W}_{att} \cdot [\mathbf{h}_{prot}, \mathbf{h}_{mol}] + \mathbf{b}_{att})$" produces a uniform attention weight distribution in most cases.

Up to now, we only explored modifications of the neuron network architecture to encode (protein,molecule) pairs from their raw representation. We conclude that searching for more sophisticated architectures is not the right direction to enhance drug virtual screening with the chemogenomic neuron network, at least as observed on our DrugBank-based benchmark dataset.

In other words, since the performance does not seem limited by the representation power of the chemogenomic neuron network architecture, the bottleneck seems to be related to data featurisation, diversity and quantity.

In particular, a feed forward neuron network on top of data-blinded features is highly competitive. Therefore, we evaluate the simultaneous use of data-blinded and data-driven features in the next section.

8.2.7 Evaluation of the combination of data-blinded and data-driven features

Since FNN outperforms the chemogenomic neuron network in some settings, we tested a model architecture that can differentially leverage two forms of inputs using appropriate neuron network components: (i) the expert-based features and (ii) the raw data (molecular graph and protein sequence). Intuitively, such a network should benefit from the best of both forms of representation, and assess the degree with which each representation is efficient for each individual interactions. Therefore, it is expected to perform better than the two type of representations (i.e. data-blinded and data-driven) taken separately.

Similarly to Paul et al. work [354], we consider an MLP to learn an abstract representation from expert-based features.

In the following discussion, we considered the same data-blinded features that we used as input of the FNN-based model.

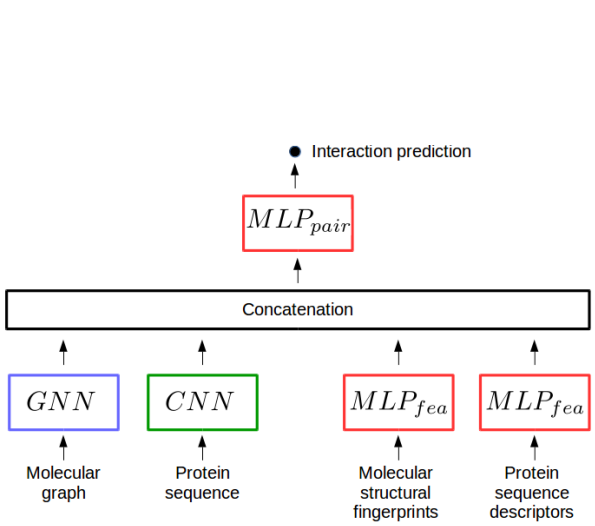
Furthermore, there are several possibilities of neuron network architectures to integrate multiple views of protein and molecule, in particular, data-blinded and data-driven features, into a final pairwise representation. We considered the two architectures displayed in Fig 8.3.

The architecture in Figure 8.3a combines in one step the four data representations extracted from: (i) the molecular graph, (ii) the data-blinded molecule features, (iii) the protein sequence and (iv) the data-blinded protein features. Regarding the multi-layer perceptron which processes molecule and protein data-blinded features (MLP_{fea} in fig. 8.3a) and the one processing the concatenation of the four representations (MLP_{pair} in fig. 8.3a), we explored the hyper-parameter spaces by varying the number of layers (from 1 to 3), the number of hidden units (in {10,100,200,1000,2000}) while promoting pyramidal architectures [333] and the quantity of dropout (in {0,0.3,0.6,0.9}).

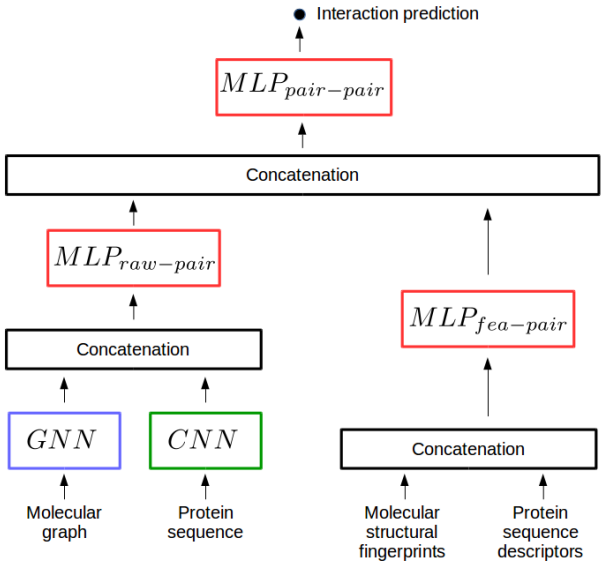
To rule out the possibility of an implementation error, we performed two sanity checks. First, we checked that we recover the performance obtained by an MLP with data-blinded features when we set to one the dimension of the data-driven features (i.e. we silent the protein sequence and molecular graph encoders while keeping the same implementation). Similarly, we also checked that we recover the performance obtained with data-driven features while silencing MLP_{fea} and MLP_{pair} in the same manner.

The performance reached by such a model is significantly lower than that obtained individually by the chemogenomic neuron network and an FNN with data-blinded features. A possible explanation is that simultaneous training of the MLP_{fea} and MLP_{pair} networks with molecular graph and protein sequence encoders is an ill-conditioned non-convex optimisation problem.

To address this issue, we first separately pre-trained the pairs of neuron modules encoding data-blinded and data-driven features before re-training them together on the same training data. With this procedure, we could only retrieve the performance reached when considering the data-driven representation alone, even when increasing the width and depth of MLP_{pair}. Therefore, such architecture fails to positively leverage the information contained in data-driven and data-blinded features.



(a) Direct combination of the molecule and protein data-blinded and data-driven features into a pairwise representation.



(b) Hierarchical combination of the pairwise data-driven and data-blinded representations.

Figure 8.3. Two types of chemogenomic neuron networks combining data-driven and data-blinded features.

To further explore joint optimisation of neuron modules for data-driven and data-blinded feature extraction, we considered the architecture in figure 8.3b. It combines the pairwise data-driven and data-blinded representations such that data-driven and data-blinded representations are combined only through a single relatively small layer. Typically, $MLP_{pair-pair}$ (cf. Figure 8.3b) is composed of a single hidden layer of 50 hidden neurons. In the following, we call this model the $CN\text{-}feaMLP$ approach.

We varied the neuron architecture and dropout probability of $MLP_{fea-pair}$ similarly as for MLP_{fea}, and the ones of $MLP_{raw-pair}$ and $MLP_{pair-pair}$ similarly as for MLP_{pair}, in the architecture displayed in Fig. 8.3a.

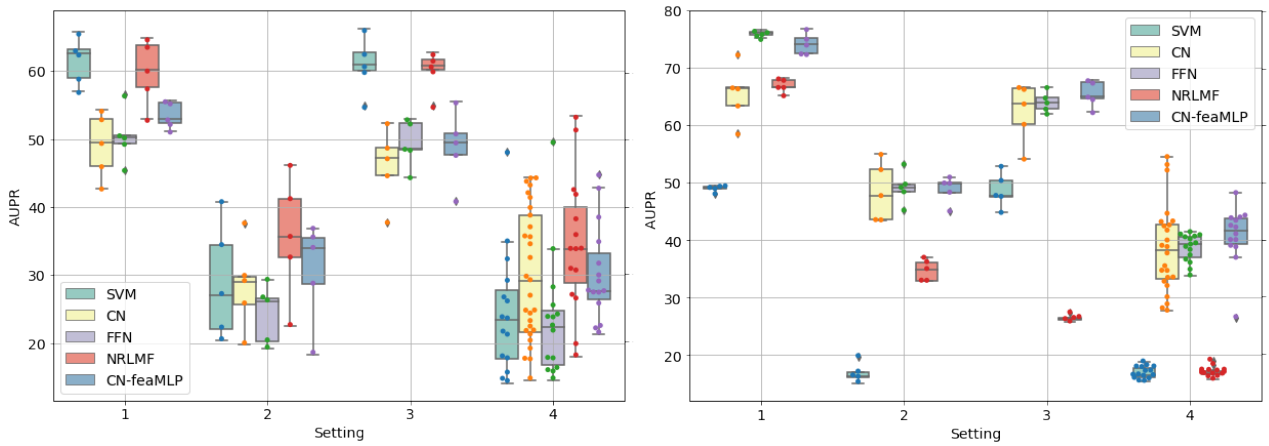
The performance of such a network are reported in Table 8.8 at the row named $CN\text{-}feaMLP$. Remarkably, this approach significantly outperformed the standard chemogenomic neuron network for the S_1 setting, while performing similarly on the others. More precisely, the best-reported performance on the S_1 setting is the one reached by an FNN using only data-blinded features as input, so that we can expect that the architecture in fig. 8.3b benefits from the best of data-driven and data-blinded features.

To further challenge this approach, we assessed the performance on the four settings via a 5-fold nested cross-validation with a test and train "positive:negative" ratio of 1:5. The performance on $DBEC$ and DBH are reported in figure 8.4 next those reached by the baselines.

In the case of the $DBEC$ dataset, the $CN\text{-}feaMLP$ model outperforms both the standard chemogenomic neuron network and the FNN-based model on S_1 and S_2, and performs similarly, yet slightly better, for the S_3 and S_4. However, the performance is still significantly lower than that of $NRLMF$, on this small dataset. Therefore, we conclude that the $NRLMF$ approach should be used on relative small datasets.

On the DBH dataset, the $CN\text{-}feaMLP$ model outperforms the standard chemogenomic neuron network model for the S_1, but it does not overtake the FNN approach. For S_2, S_3 and S_4, the $CN\text{-}feaMLP$ approach reaches slightly better performance than both the standard chemogenomic neuron network and the FNN-based model.

Therefore, for relatively large datasets like DBH (that are widely available our-days), we advocate for the use of the $CN\text{-}feaMLP$ model. Moreover, we believe that more efforts should be done in the future in integrating multiple description views of molecules and proteins in the same model.



(a) $DBEC$ dataset (b) DBH dataset

Figure 8.4. AUPR scores obtained via 5-fold nested cross-validation for the four settings. The performance of $CNN\text{-}feaMLP$ must be compared to the others (the baselines).

In the next subsection, we explore another relevant "data-oriented" direction to enhance the performance of the chemogenomic neuron network: transfer learning for implicit data augmentation.

8.2.8 Evaluation of curriculum learning for chemogenomics

We now investigate the efficiency of transfer learning in the context of chemogenomics.

First, recall that the chemogenomics formulation of proteome-wide DTI prediction is a form of multitask learning since, in this framework, we predict interactions for a wide variety of molecules and proteins.

Moreover, pre-training or co-training molecular and protein encoders on separated larger prediction tasks (even unsupervised ones) may enhance DTI prediction performances by leveraging the information in more massive datasets.

Co-training refers to parameter hard-sharing, soft-sharing or partial-sharing (see appendix A.8.3) in which we train several prediction tasks simultaneously, while task-specific neuron encoders share information in order to enable them to learn a richer set of features. As we saw in our preliminary work, such multitask

co-training procedures may be hard to tune and relevant only for closely related tasks.

Pre-training refers to the so-called "curriculum learning" procedure in which the model is first trained on a large dataset (called the source dataset) to learn a rich set of data-driven features, which is useful to make predictions in large training data.

Next, this pre-trained model is fine-tuned by re-training the weights on the smaller dataset of interest (named the target dataset). In such a procedure, the model fits the input data distribution of the large source dataset and fine-tunes its weights in a second phase to adapt to the dataset of interest, rather than learning all the weights from scratch on this target dataset.

Alternatively, it is also possible to "freeze" (i.e. do not retrain) the pre-trained molecule and protein encoders, and train only the last layers that take as input the extracted representations from the pre-trained encoders. In the case where the source and target datasets are related "enough", this could improve predictions on the target dataset.

For instance, Paul et al [355] performed such transfer learning in chemoinformatics in the case of organic solar cell screening.

Before discussing our results, we start with general observations about implementation of curriculum learning with the chemogenomic neuron network.

First, we performed a sanity check to challenge our implementation of curriculum learning. More precisely, we checked that a pre-trained model reaches immediately best reachable performance when re-using the pre-trained network on the same train and test data, with or without freezing all the layers.

Second, we found that curriculum learning made often the training much shorter, in terms of the number of training epochs in the second phase. Indeed, pre-trained encoders are expected to be closer to the optimal solution than encoders whose weights are randomly initialised.

Third, we considered to pre-train and load per block three sub-networks of the chemogenomic neuron network: (i) the protein sequence encoder, (ii) the molecular graph encoder, and (iii) the prediction sub-network, i.e. the stacked fully connected layers outputting the interaction prediction from the learnt pairwise data-driven representation. Furthermore, we varied the number of layers and neurons in the prediction sub-network in order to search for the best architecture when freezing one or the two encoders. Indeed, an MLP may need a relatively large width and depth to differentially leverage the representations extracted by pre-trained frozen encoders.

In a first attempt, we considered to pre-train the three parts of the chemogenomic neuron network (the protein sequence encoder, the molecular graph encoder and the prediction sub-network) on the DBH dataset in order to use it for prediction on the $DBEC$ dataset. This led to a loss of performance on S_1 (0.2 of AUPR instead of 0.4).

Interestingly, we noticed that the bottleneck was the protein encoder pre-trained on DBH. Indeed, pre-training on DBH and freezing the protein sequence encoder, while training the molecular graph encoder and prediction sub-network from scratch, led to the loss of performance mentioned above. In contrary, when pre-training and freezing the molecular graph encoder while training "from scratch" the protein sequence

encoder and the prediction sub-network, we obtained similar or better performance than with direct training of the chemogenomic neuron network, as shown in table 8.8 at the row named "curriculum learning". In particular, the performance obtained on S_1 and S_3 were significantly better. This means that DTI prediction with the chemogenomic neuron network may benefit from a molecular graph encoder pre-trained on larger datasets.

The two prediction tasks (DBH and $DBEC$) are formally identical, but they differ because their data have different characteristics. Indeed, $DBEC$ contains E. Coli proteins and their ligands, whereas DBH contains human proteins and their ligands. Therefore, the two datasets may present critical statistical differences regarding protein sequences distribution. However, both datasets deal with small organic drug-like molecules, so that a molecular graph encoder pre-trained on the DBH dataset is also suitable to predict DTI on the $DBEC$ dataset.

When we pre-trained the encoders without freezing them in the second phase, and use an initial learning rate in the second phase one order of magnitude lower than the one used to train directly the chemogenomic neuron network (i.e. initial learning rate set to 10^{-4} instead of 10^{-3}), the performance was lower than the one obtained when training the chemogenomic neuron network from scratch. In particular, the average AUPR is 0.3 instead of 0.4 on S_1. This means that encoders' weights pre-trained on DBH, in particular those of the protein encoder, are not suitable starting points for fine-tuning on $DBEC$.

To investigate these conclusions, we selected large and complex chemical and protein datasets to pre-train the encoders.

The PubChem dataset is an excellent choice for pre-training a graph neuron network encoder for molecules since it contains the information about hundreds of bioactivities for hundreds of thousands of molecules. To our knowledge, there is no such evident choice for protein datasets. We considered the sub-cellular localisation dataset [279], containing six thousand proteins labelled with one of the twelve sub-cellular localisation.

Thus, we considered pre-training only two parts of the chemogenomic neuron network, (i) the protein sequence encoder and (ii) the molecular graph encoder, on the $PCBA$ and $CellLoc$ datasets respectively, in order to use them for prediction on the $DBEC$ dataset. When pre-training the molecular graph and protein sequence encoders on the $PCBA$ and $CellLoc$ datasets respectively, and used them as frozen to fit a model on the $DBEC$ dataset, we recovered the performance reached by direct training of the chemogenomic neuron network on $DBEC$. More precisely, we recovered a performance of 0.4 in AUPR only with a prediction neuron sub-network with three hidden layers of 1000, 500 and 100 hidden neurons each. However, we obtained 0.28 of AUPR with a prediction neuron sub-network made of a single hidden layer of 100 hidden units. This means that the prediction sub-network needs to be complex enough to leverage the information in the representations extracted by the pre-trained encoders. In other words, these pre-trained encoders are informative with respect to the $DBEC$ prediction task, but not as suited as encoders trained on the $DBEC$ prediction task itself.

To challenge curriculum learning for DTI prediction with the chemogenomic neuron network, we evaluated the pre-training of the molecular graph encoder on DBH and $PCBA$ via 5-fold nested cross-validation with a

test and train "positive:negative" ratio of 1:5 on the four settings. The performance on $DBEC$ are reported in figure 8.5 next to those reached by the baselines. The final model obtained when pre-training the molecular graph encoder on the $PCBA$ (resp. DBH) dataset is called $CN\text{-}currPCBA$ (resp. $CN\text{-}currDBH$) in Fig.8.5

Pre-training the molecular graph encoder on $PCBA$ yielded to similar performance than direct training on $DBEC$ for the four settings. We thus confirmed the observations previously made on a single train/validation/test split for the four settings.

Interestingly, pre-training the molecular graph encoder on DBH resulted in a better model than the one trained from scratch on S_1, S_2 and S_3 for the $DBEC$ dataset. Although the proteins involved in DBH and $DBEC$ follow different statistics, these two prediction tasks are indeed strongly related.

Future studies could consider larger source datasets. For instance, we could consider an unsupervised learning task as a source task such as a reconstruction task like in auto-encoders [322, 280] (predicting the original input from the hidden representation), or an adversarial task [356] (distinguishing real data from fake data, such as non-valid molecules or randomly generated protein sequences).

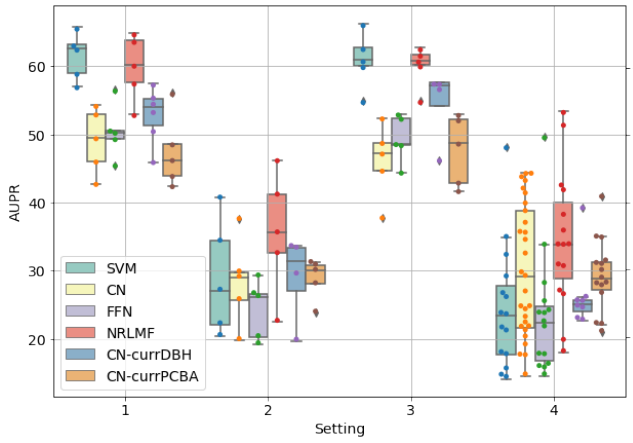


Figure 8.5. AUPR scores obtained via 5-fold nested cross-validation performance for the four settings on the $DBEC$ dataset. The performance of $CNN\text{-}currPCBA$ and $CNN\text{-}currDBH$ is to be compared to the others (the baselines).

		raw (S_1)	orphan proteins (S_2)	orphan molecules (S_3)	double orphan (S_4)
	standard chemogenomic neuron network (CN)	39.57 ± 4.17	26.74 ± 2.49	43.74 ± 2.35	24.63 ± 1.89
change in $AGGREGATE_{node}$	CN-GAT	39.97 ± 4.64	25.84 ± 3.58	43.53 ± 2.35	26.28 ± 3.72
	CN-wbond	35.65 ± 1.47	23.27 ± 1.97	34.38 ± 3.87	25.01 ± 4.14
change in graph-level encoding	CNN-MaxAgg	38.33 ± 2.85	26.24 ± 2.29	46.06 ± 2.28	21.57 ± 2.41
	CN-ConcatCombine	43.77 ± 3.59	26.04 ± 2.29	44.31 ± 2.45	24.97 ± 3.50
	CN-pool	40.54 ± 6.82	18.71 ± 0.53	36.63 ± 7.95	19.24 ± 2.60
change in protein encoding	CN-biLSTM	40.65 ± 1.88	28.69 ± 1.55	41.55 ± 3.08	22.79 ± 2.72
change in protein and molecular representations combinations	CN-pairwiseAtt	35.76 ± 2.15	23.84 ± 5.15	41.96 ± 2.92	22.69 ± 7.21
adding extra-information	curriculum learning	45.06 ± 2.64	21.43 ± 3.62	51.45 ± 3.03	20.97 ± 2.70
	$CN - feaMLP$	50.616 ± 2.71	30.89 ± 4.93	42.04 ± 2.95	25.77 ± 3.60

Table 8.8. AUPR score of various modifications of the chemogenomic neuron network on a single train/validation/test split of the *DBEC* dataset. Standard deviations are obtained after repeating 5 times the evaluation procedure.

8.3 Conclusions

First, we emphasise that a singletask feed-forward neuron network taking as input standard molecular fingerprints and protein sequence descriptors is a good default method for prediction of DTI on relatively large datasets like DBH. Indeed, a standard FNN reached state-of-the-art performances on the DBH dataset although we did not fine-tune the FNN architecture or select the most relevant task-specific data-blinded descriptors (proteochemometric descriptors [143]).

Based on our study and the review of the literature in chapter 7, we state that FNN should be considered as a reference simple method that leads to state-of-the-art performance, and to which future chemoinformatic studies should be compared.

When using such models to guide pharmaceutical research, more efforts should be involved in the design and selection of suitable descriptors. Furthermore, in the case of FNN, recall that multitask learning was reported to be only beneficial for two tasks when they share a large pool of input compounds with correlated or anti-correlated bio-activities.

In the present work, we report that the chemogenomic neuron network is competitive with FNN models, particularly in orphan settings like S_2, S_3 and S_4, for which we found that the chemogenomic neuron network reached the best performance.

We conclude from our observations that a promising future research direction is to integrate heterogeneous sources of data such as various bioactivity datasets, and independently, multiple molecule and protein attribute views. Multiplying data sources is expected to enable the integration of chemical knowledge that covers a larger sub-space of the chemical space, while improving the signal to noise ratio.

About data sources integration, we investigated in this work two promising direction. That is, the combination of data-blinded and data-driven features in a single neuron network and curriculum learning with larger source datasets. Integrating data-blinded and data-driven features in the same network yield to substantial improvements so that we recommend, in the future, to involve more efforts in the design of neuron network to integrate multiple views of molecules and proteins in the same model. In contrary, we report that curriculum learning did not result in strong performance improvement on the relatively small $DBEC$ dataset, meaning that a promising direction to improve the chemogenomic neuron network is at first to specifically vary data views rather than increasing the amount of auxiliary data.

Furthermore, we conclude that the latest algorithmic development about graph convolutional network did not prove to enhance DTI prediction with the chemogenomic neuron network. Indeed, we observed that both promising modifications of the molecular graph encoder and pairwise representation encoder do not yield to significant superior performance. Our result, in addition to the literature review in chapter 7, strongly suggest to qualify the current trend of graph neuron network, and more generally of end-to-end representation learning, for chemoinformatics. Moreover, we encourage further studies to justify more the design of their approaches, most of all compared to state-of-the-art baselines, i.e. $NRLMF$ and FNN.

Nevertheless, GNN is a very dynamic field and new developments might lead to improvements in chemogenomics.

In practical applications of DTI prediction, we suggest to consider an ensemble learning strategy (majority voting or max voting, see section 3), combining the predictions of *NRLMF*, FNN and the chemogenomic neuron networks, since they individually perform the best on different types of the problems.

The reachable performance depends greatly on the test "positive:negative" sample ratio. Nevertheless, several strategies are conceivable to determine the probability threshold up to which we can consider prediction interactions as highly probable (see section 5).

Besides, neuron network-based models were not efficient on the small dataset *DBEC* on which shallow methods performed the best. In particular, *NRLMF* appeared to be the method of choice in this case. Nevertheless, we think that most models should be evaluated on relatively large datasets, with thousands of recorded interactions, since the publicly available databases now reach this size. In particular, evaluating new methodologies on small datasets scrambled the examination of new methodologies, like in [349].

To conclude, the most promising research directions for improving DTI prediction in chemogenomics is, from our point of view, the integration of heterogeneous data sources, since the latest most relevant algorithmic development did not prove to enhance DTI prediction power. To this end, multi-kernel learning, in particular, was reported to be efficient for DTI prediction [165, 167].

Part IV

Perspectives

Chapter 9

Short-term perspectives for chemogenomics

Abstract: *The problem of predicting drug-target interactions has been addressed from multiple points of view with relative success, when looking at the achieved performance. However, it is still a challenging problem. Indeed, only a relatively small portion of the chemical space of molecules has been explored and assayed so that the data statistics limit the predictive power of machine-learning algorithms. In this final chapter, we introduce potential ways of improving drug virtual screening. We first discuss the use of 3D dimensional data with data-driven representation learning approaches. Indeed, such 3D information, when it is available, is expected to engender more plausible predictions. Second, we debate the integration of heterogeneous data sources with deep neuron networks. Integrating different types of features within a hybrid setting is expected to increase both the quality and the coverage of DTI predictions since different features complement each other to produce a complete picture.*

Résumé: *Le problème de la prédiction d'interactions entre des médicaments et des cibles a été abordé à partir de multiples points de vue avec un succès relatif. Cependant, c'est toujours un problème qui n'est pas bien résolu. En effet, seule une partie relativement petite de l'espace chimique des molécules a été explorée et testée, de tel façon que la statistique des données disponibles limitent le pouvoir prédictif des algorithmes d'apprentissage statistique. Dans ce dernier chapitre, nous présentons des pistes pour l'amélioration du criblage virtuel de molécule. Nous discutons d'abord de l'utilisation de données 3D avec des approches d'apprentissage de représentation. En effet, ces informations 3D, partiellement disponibles, devraient résulter en des prédictions plus plausibles. Deuxièmement, nous débattons de l'utilisation de données hétérogènes dans le cadre d'apprentissage de la représentation. L'intégration de différents types de descriptions des données devrait permettre d'améliorer à la fois la qualité et la couverture des predictions d'intéraction entre molécule et protéine. En effet, différents types de descriptions peuvent se complèter pour produire un modèle plus global.*

In this chapter, we will shortly present a few ideas that we suggest to explore for future researches in drug-target interaction prediction, conceived from our experience during this thesis. These suggestions mainly concern the use of new types of data together with representation learning in the context of deep learning.

9.1 Representation learning for drug-target interaction prediction with 3-dimensional data

In principle, one would expect methods that handle information in the 3D space to improve the prediction performances, since binding of a ligand into the pocket of a protein is an event that occurs in the 3D space. We will not consider here 3D information about the protein, because this is not available proteome-wide, as already mentioned. However, considering 3D conformers of the ligand may be useful, since the conformation of the bound ligand bears information about the complex that is not available from the protein sequence or from the 2D graph of the ligand molecule. Nevertheless, as discussed in chapter 2, the "active" conformation (i.e. the conformation of the bound ligand) is usually unknown, and attempts to use the most stable conformer (which might not be the active conformer) did not perform better than methods based on 2D descriptors [96, 97].

Therefore, one could describe molecules based on an ensemble of conformers, with the underlying idea that the active conformation could be close to one of the sampled conformers. Then, it may be relevant to investigate the extension of graph convolutional networks to several 3D conformers of the same molecule..

However, considering molecular conformers comes with two essential problems: conformer evaluation (how to choose a conformer, or a set of conformers relevant to the prediction at hand) and conformer combination (managing the high correlation between different conformers). The former problem can be solved using various molecular modelling programs, which contain tools that can generate as many conformers as needed, by systematically sampling rotamers around all rotatable bond (for example via the program Omega from the suite of OpenEye programs).

Then, using conformers for a DTI prediction task can be formalised as a multiple instance learning problem. In this framework, each sample is a bag of instances. In our case, each molecule is a bag of multiple possible conformers.

Each conformer can be describe by a same 2D undirected graphs, but whose atoms' and bonds' descriptors encode 3D information (inter-atomic distance for instance).

A naive solution to build a bag-level (i.e. molecule-level) representation could be to concatenate or average the learnt representation of individual conformers. However, it is clearly not an ideal solution as only one instance (one conformation), or few instances, may contain the information for the prediction task.

A solution could be to add an attention mechanism after the encoding step to guide the network to learn what graph-instance representation (what conformer) to focus on.

To handle the correlation information among different graph conformers, we could investigate passing some information across different graphs during the learning process.

Another idea could be to treat all graph conformers simultaneously, by taking each conformer as a channel (analogously with image RGB channels) in a molecule tensor representation.

Several studies have been published on multiple-instance learning with neuron networks.

In Tibo et al. [357], authors define a "bag-layer" to aggregate the numerical descriptions of the multiple instances of a bag. First, the bag-layer projects each instance representation z_i into the same k-dimensional space via $h_i = \sigma(W.z_i + b)$, using a weight matrix $W \in \mathbb{R}^{k*m}$, a bias vector $b \in \mathbb{R}^k$, and an activation function σ (such as ReLU, tanh, or linear). Then, the bag-level representation $g(z_1, \ldots, z_n)$ results from the aggregation of the instance representation following $g(z_1, \ldots, z_n) = aggregate(h_1, \ldots, h_n)$, where $aggregate$ is an element-wise aggregation function (such as max or average).

From another perspective, Jiang et al. [358] proposed to learn a single representation from multiple graphs, via an adversarial learning architecture. In this study, they used the same encoder to encode several graph instances of the same sample. The concatenation of these encoded representations is then fed into an MLP to solve a supervised learning task. The encoder is trained to simultaneously solve the supervised learning task and to fool a discriminator network which aims at recovering the original graphs corresponding to each encoded representation. The goal of this adversarial task is to force the encoder to generate a consistent representation across different graphs of the same bag in a common subspace.

In the next subsection, we introduce another interesting direction to increase chemical information in order to improve both the quality and the coverage of DTI predictions.

9.2 Integration of heterogeneous data sources

One of the most stimulating promising direction in chemogenomics is the integration of various types of information, in particular large-scale omics data (e.g. transcriptomics, interactomics, epigenomics, metabolomics or functional genomics), as input of the model. It is expected to increase both the quality and the coverage of DTI predictions by integrating different types of features that complement each other to produce a complete picture of the physical and biological phenomena that are related to drug-target interactions. For example, differential mRNA expression after incubation in presence of a drug with respect to incubation with no drug provide a list of differentially expressed genes that can be used as descriptors of the drug, in combination with structural features and drug target profiles. This additional information can compensate for unknown targets, or can help reveal, in the case of the considered cell line, which target drives the mechanism of action, thus guiding feature selection algorithms.

Computationally speaking, the generation of attribute vectors with the standardisation of the information is a critical issue since chemical data and all omics data have different structures.

Besides, feature selection methods, that aim at eliminating irrelevant and redundant features, are also of crucial importance since omics data are extensive, and since the raw feature vectors produced by ensemble methods are also usually quite large. Indeed, too many features increase the computational cost and may

decrease the performance because of high dimensional issues, or correlation of features.

We survey multi-kernel learning for DTI prediction in appendix B. It is a powerful and efficient method to combine multiple heterogeneous views (i.e. data sources) for chemogenomics. This method would be well suited to the integration method proposed here.

To our knowledge, a study explored combinations of heterogeneous views in the representation learning framework [359]. This study tackles the question of modelling brain connectivity patterns, and we have not seen any attempts in the domain of DTI prediction.

The authors used a variational auto-encoder -VAE- (see appendix A.8.2) to learn a latent representation from the original features, such that the similarities between instances based on each views guide the representation learning. More precisely, they used an additional loss term based on the side-views to take into account the similarity between samples, and guide the learning of the intermediate latent representations.

To detail this example, let us note h_i the latent representation of the instance i and $s_{ij} = \prod_{k=1}^{\mathcal{K}} s_{ij}^k$ the total similarity value between instances i and j based on the K side views, such that view-specific similarities s_{ij}^k are typically defined with the RBF function $s_{ij}^k = exp(\gamma||z_j^k - z_i^k||^2)$ (z_i^k is the descriptor vector of sample i in the k^{th} side-view). Then, the total loss is simply:

$$\mathcal{L} = \mathcal{L}_{VAE} + \lambda \sum_{i=1}^{N} \sum_{j>i}^{N} s_{ij} ||h_j - h_i||_2^2$$

Managing heterogeneous data types by manipulating similarities between samples instead of attribute vectors is particularly convenient, since it fixes the issue of the standardisation of the information. However, it is not integrated into the representation learning framework directly.

Much works remain to be done.

Even though large amount of data are generated every year, publicly available chemical and biological data still face the problem of data scarcity (consistency of the panel of properties measured for a panel of samples), and quality (consistency of measures between databases). This is particularly critical, since the of quality of deep learning models significantly depends on the quality of the input data [360].

Chapter 10

Conclusion

In this thesis, we have explored how machine learning methods can be adapted to study drug specificity as a problem of predicting interactions between drugs and proteins at the proteome scale.

In the first chapter, we explained how computational approaches for drug specificity prediction could assist the drug discovery process. Indeed, they can identify drug candidates for known therapeutic targets, identify the target, and the mechanism of action of active molecules discovered by phenotypic tests, anticipate potential side effects or drug-drug interactions by identifying secondary protein targets, or suggest repositioning of known drugs in new therapeutic indications.

In the second chapter of the manuscript, we discussed the numerical representation of molecules and proteins, since it is a crucial prerequisite for machine learning methods. We also introduced available toolkits and databases that are considered throughout the manuscript.

In the second part of the manuscript, we first reviewed methodological advances in machine learning algorithms for drug virtual screening, based on expert-based extracted descriptors or similarity measures (chapter 3).

More precisely, we first discussed the main frameworks in drug-target interaction prediction: docking, ligand-based and the chemogenomics frameworks. Then, we reviewed data-blinded approaches whose input feature vectors are either handcrafted in an ad hoc manner, or calculated by toolkits that were designed based on chemists expertise.

This allowed us to propose *NN-MT* in chapter 4, a multitask Support Vector Machine algorithm that is trained on a limited number of data points, in order to solve the computational issues of proteome-wide SVM for chemogenomics. We showed that the prediction performances of *NN-MT* are, at an efficient calculation cost, similar or better than various state-of-the-art methods. *NN-MT* was particularly efficient when predicting (protein, ligand) interactions in the most difficult double-orphan case, i.e. when no interactions are previously known for the protein and for the ligand. The *NN-MT* algorithm appears to be an appropriate default method providing state-of-the-art or better performances, in a wide range of prediction scenarios : proteome-wide prediction, protein family prediction, test (protein, ligand) pairs dissimilar to pairs in the

train set, and orphan cases.

However, predicting drug-target interactions is still a challenging problem. Indeed, only a relatively small portion of the chemical space of molecules has been explored and assayed so that the data statistics limit the predictive power of machine learning algorithms. In this perspective, we thus emphasise that machine learning methods should be evaluated and optimised toward unexplored chemical subspaces, and should particularly focus on the double orphan prediction case.

In chapter 5, we present our contributions to two collaborative projects with medical applications: cystic fibrosis and triple negative breast cancer. The cystic fibrosis project illustrates how chemogenomics approaches can propose realistic targets for drugs identified by phenotypic tests, and how analysis of the predicted targets can explain and predict drug-drug interactions. At a fundamental level, it shows how chemogenomics can help identify molecular mechanisms at the origin of the disease symptoms, and explain the relations between the pathways perturbations. Most of all, it shows how target predictions provides comprehensive interpretation of biological results reinforcing the interest of the corresponding experimental results. This project is also a striking example of how chemogenomics can guide future experiments, and even in our case, suggest a whole research program in translational medicine based on drug repositioning.

The triple negative breast cancer project also illustrates the interest of chemogenomics to propose targets for drugs identified by phenotypic tests. It also illustrates how different types of data can be used to complement chemogenomics results. In this case, independent biological data were not integrated within the prediction step, but used to help analysis of the predictions and identify the most promising biological pathways to be targeted in TNBC. It again illustrates how prediction of targets can help interpretation of biological experiments and help them "make sense". These results were obtained at the very end of this PhD, and we hope that we will soon be able to publish them, and that they will be the starting point of a specific research project with our biologist collaborators. More particularly, we hope that understanding how some marketed drugs for psychotic diseases might target a key potassium channel in TNBC, could pave the way a new translational medicine in drug repositioning.

In other words, drug virtual screening is a tool that provides an entry point to the understanding of biological phenomena, and diseases in particular. Indeed, drug virtual screening is a creative force that provides different visions and proposes other research directions.

In the third part of this manuscript, we surveyed and investigated end-to-end feature extraction of molecules and proteins based on deep learning for drug-target interaction prediction.

We first defined neuron network-based encoders to extract data-driven features on molecular graphs and protein sequences, so that the data representation itself is learnt from the raw description in order to maximise prediction performance. Furthermore, we notably provided a historical review of graph representation learning for chemoinformatics in appendix C. This allowed us to offer a general view of graph convolutional neuron network, i.e. neuron network-based encoder of undirected graphs.

Based on this critical appraisal, we investigated the latest developments in graph representation learning, and point out critical issues that question the efficiency of data-driven approaches for chemogenomics. Although machine learning topics covered in the present thesis, and their applications in virtual screening

are not exhaustive, we can conclude that the current research in the neuron network architecture for encoding molecular graphs and protein sequences does not seem a promising direction to improve drug virtual screening in the chemogenomics framework. Indeed, we tested a variety of recent promising neuron network architectures for molecular graph and protein sequence encoders as part of the chemogenomic network. None of them enhanced the performance of the chemogenomic network on a benchmark dataset built from the DrugBank database.

In contrast, based on our results, integration of heterogeneous data sources appeared more encouraging. Indeed, we reported that integration of data-blinded and data-driven features, and more generally of different data views, is a promising direction to boost the chemogenomic network, as well as the integration of heterogeneous data sets. Besides, we reported that feed-forward neuron networks applied to expert-based descriptors are competitive and even yield to the best performance in the raw setting (S_1) of proteome-wide DTI prediction. Moreover, remember that we did not fine tune this model, and most importantly, we did not considered the most relevant expert-crafted descriptors for the DTI prediction task, such as proteochemometric descriptors [143], which leaves some space for improvement and reinforces the interest of this approach.

Our main contributions in this work was to discuss and tackle proteome-wide drug target interaction prediction from multiple perspectives. In particular, we experimented and developed with success SVM-based and deep learning-based approaches for proteome-wide DTI prediction.

In chapter 9, we proposed some perspectives to improve drug-target interaction predictions with non-linear methods. Indeed, many interesting perspectives that were not covered in the present thesis remain to be explored.

In particular, the use of 3D dimensional data for data-driven representation learning could provide some relevant descriptors for the DTI prediction problem, since the binding of a ligand into the pocket of a protein is an event that occurs in the 3D space.

Again, we think that the most promising perspectives to enhance performances rely more on mixing heterogeneous data. Indeed, all data sources are expected to be noisy (missing data, experimental noise) but with different noise pattern, and to cover different sub-parts of the chemical space. Therefore, combining data sources may have a regularisation effect as well as provide information from a wider sub-part of the chemical space.

To use multiple views of chemical data (chemical structure, bioassay signatures, side-effects profiles, drug-induced phenotypes, pharmacophore patterns) in an integrated framework, multi-view learning [361] is a branch in machine learning that studies how to combine different and heterogeneous views of a sample.

Alternatively, within the paradigm of kernel-based learning, if a kernel is defined for each view of the data, multi-kernel learning has been shown relevant for chemical and genomic data fusion [362, 54, 166, 167, 165]. In particular, Olayan et al. [165] significantly outperformed state-of-the-art approaches because they relied on a fusion of similarity measures to describe molecules and proteins, but not on algorithmic refinements.

It should be emphasised that one important contribution of this thesis is to propose an extensive review of methods for drug virtual screening, with efforts to structure the considerable amount of recently published works. In chapter 3 we reviewed and organised drug virtual screening approaches based on expert-based features. Later in chapter 7, we reviewed and organised data-driven feature-based approaches for virtual screening. Moreover, we proposed a general survey of graph convolutional network. In particular, we critically appraised and structure the variants of spatial graph convolutional networks in section 6.2. Moreover, in appendix C, we proposed a historical perspective on graph representation learning. Furthermore, in appendix D, we surveyed and draw conclusions from the few available studies on the fundamental analysis of graph neuron networks. Finally, in appendix B, we introduce the most prominent works considering multi kernel learning for DTI prediction, since we think it is a valuable direction of research. These reviews on some basic but also some recent and advanced topics, aimed at providing a "self consistent" manuscript for the reader. We hope that these review contributions could help new comers in these fields to find their way in an abundant literature that does not always help to embrace a global overview.

Finally, all along this thesis, we considered computational approaches that predict candidate drugs' targets and mechanisms of actions in the intention to assist the drug discovery process. In the same objective, the reverse operation is of great interest. It consists in proposing ligands for specific targets, or more generally, proposing organic compounds with the desired biological and pharmacokinetics profiles. Automatic generation of molecules with specific properties is a very dynamic field which has undergone very important developments in the last two years. Although we did not tackle this question, because it is very promising and popular topic, it seemed crucial to us to review current progresses in this field. Thus, we propose, in appendix E, a review of the field that is organised to answer, in different ways, the following questions: how to generate molecules in silico, how to bias the generation process towards specific properties, and how to evaluate and compare molecular generative approaches. We conclude this survey by a brief comparison of the reviewed methods. To our opinion, it is also a contribution of this thesis that we let in appendix E for the interested reader.

As a concluding and somewhat provocative remark, we found that much exciting work can be done to interface drug-target interaction prediction and therapeutic research. Competing for minor (meaningless ?) performance improvements through algorithmic considerations must not be the ultimate goal of virtual screening.

Part V

Appendices

Appendix A

Machine learning in brief

In this appendix, we introduce all the machine learning notions that are required to understand this thesis work.

More precisely, we first give a general perspective on machine learning and on supervised learning in particular. Then, we briefly introduce linear models, tree-based models and kernel-based models. Following this, deep learning models are introduced and discussed at length. Note we also consider the framework of multitask learning in the context of linear-based, kernel-based and deep learning models.

We do not aim at providing an extensive discussion, but intuitively present the methods to which we refer in the main manuscript. Interested readers can refer to the following textbooks [363, 176].

A.1 Machine learning basics

Given enough data, machine learning is about algorithms and processes providing an efficient framework to discover the underlying structure of the data.

More precisely, we consider n *examples* (or, equivalently, *data points*, or *observations*, or *samples*) $\{\boldsymbol{x}^1, \dots, \boldsymbol{x}^n\}$ that form the *data set*.

All data points are described by p-dimensional vectors $\boldsymbol{x}^i \in \mathbb{R}^p, i \in \{1, \dots, n\}$. The p attributes describing data samples are called *features*, or equivalently *descriptors*, *variables* or *attributes*. In the general case, we note $\mathcal{X}$ the space where the data points live.

In the rest of the appendix, when mentioning the structure of the data, we refer to the native latent organisation of the data, for example the spatial, sequential, tree-like, or graph-like organisation.

A.1.1 Supervised learning

In the case of *supervised learning*, the aim is to learn a rule which assigns labels to the data. Thus, each data point is assigned to a *label* (or equivalently a *target*, or *outcome*). In other words, the goal of supervised learning is to use the data set to define a function $f : \mathcal{X} \to \mathcal{Y}$ where $\mathcal{Y}$ denotes the space where the targets

live. $\mathcal{Y}$ can be either qualitative (or categorical) or quantitative. f can be viewed a *predictor*, but is often called a model. Let us note $\{y^1, \ldots, y^n\}$ the labels assigned to each $\{\boldsymbol{x}^1, \ldots, \boldsymbol{x}^n\}$.

In the case of classification, each sample is assigned to a class. f is then called a classifier. In particular, in the case of binary classification, there are two available classes, and $y^i \in \{0, 1\}\ \forall i$. In the case of a regression problem, each sample is assigned to d-dimensional real vectors to be predicted with the rule-discovery algorithm: $y^i \in \mathbb{R}^d\ \forall i$. f is then called a regression model.

The predictor f is chosen depending on the assumptions made on the data. Typically, a predictor is composed of weights which are learnt in order to minimise the errors made by f when predicting labels for unseen samples (plus some additional constraints that we discuss later).

In particular, the weights are learned on a subset of the data called the training set, and we assess the prediction performances of the learned function on another subset of the data called the test set. Thus, the training and tests set must not overlap, in order to avoid overestimation of the prediction performance. Evaluation procedures and metrics are introduced in appendix A.2.

Moreover, predictors often require tuning a set of values of internal variables called hyper-parameters. Changes in the values of hyper-parameters can considerably improve or impair learning. The selection of optimal values usually relies on systematic grid search or random search (see appendix A.2). The development of automatic hyper-parameter selection algorithms is an active area of investigation and we introduce this domain in the context of deep learning in appendix A.8.4.

For example, one of the most intuitive machine learning algorithms is the *k-nearest-neighbour* method (k-NN), where the predicted output value for a given point relies on the label values of its k "nearest neighbours" data points, where k is an integer. In the case of classification, the prediction is determined by the most common class of the k nearest points. In the case of regression, the output value is determined by the average of the output values of the k nearest points. The hyper-parameters of such approaches are the metric chosen to compute distances between sample points, and the number of considered neighbours k.

Linear methods refer to approaches which compute the outcome y_i of data points x_i based on a linear combination of its p-features, such that $y_i = \sum_{j=1}^{p} w_j.x_{ij}$ where w_j are weights to be optimised. Linear methods are introduced in appendix A.3.

Kernel methods are a class of algorithms which use a *kernel* function. A kernel is a similarity function which takes two data points as inputs and outputs their similarity. In kernel theory, considering a specific kernel is equivalent to considering a transformation of the input data into a higher-dimensional representation space where the problem is expected to be easier to solve. We give more detail on kernel methods in appendix A.5. The best known members of this class of algorithm are support vector machine (SVM), kernel ridge regression (KRR) and Gaussian Processes (GP).

Tree-based methods are made of tree-structured decision-making algorithms called *decision trees*. Decision trees are made of a root node (starting point of the tree), branches with intermediate nodes, and terminating leaf nodes. Each node represents a decision point such as "is feature f_i bigger than or strictly lower than a specific value". The branches exiting a node are the mutually exclusive decisions from the nodes criteria.

The final leaf nodes are assigned to a predicted label depending on how it flowed from the root node to a leaf node, that is, on how it fulfilled the criteria at each encountered node. Tree-based methods are detailed in appendix A.4.

Last but not least, *artificial neuron networks* (ANN) are built with computational neurons which somehow mimic the functioning of biological neurons. Indeed, a computational neuron receives an input from other neurons, much like dendrites, linearly combines them, and emits an output to other neurons if the results reaches some threshold. ANNs are introduced and discussed in their dedicated appendix A.8.

A.1.2 Other machine learning settings.

We introduce here a non-exhaustive list of machine learning settings that we consider in the main manuscript.

Unsupervised learning (detailed in appendix A.6) corresponds to the case where no labels are assigned to the data points. The goal of unsupervised learning is problem specific. For example, we can seek for groups of data points among their overall distribution, which is called *clustering*. We may seek to estimate the distribution of the data, which is called *density estimation*. Last but not least, we might need to project the data in a lower dimensional space (for visualisation purposes for instance) which is called *dimensionality reduction*.

Dimensionality reduction may also be useful for extracting few informative features to ease interpretation. Indeed, specific issues emerge in high dimensional spaces (hundreds or thousands of dimensions), referred to as the *curse of dimensionality*, that do not occur in low-dimensional settings. For instance, we might model behaviours that are locally smooth, but the number of local regions in the space (for example, in which the prediction function is constant) increase with the space dimension. Therefore, the number of required parameters may increase with the number of dimensions.

Semi-supervised learning is the setting where only a subset of the data points used for training is labelled. In this case, the unlabelled data can be used to improve a predictor, for example to better estimate the data distribution.

The goal of reinforcement learning is to find a succession of actions that maximise a reward (a "score"). Data are coming online with no labels (like in the unsupervised learning setting), and the algorithm seeks to optimise itself to maximise a reward through a process of trial and error. Reinforcement learning models have to deal with the trade-off of exploration-exploitation, that is respectively, exploring new actions and exploiting actions close to those known as best ones.

Multi-instance learning is a setting in which samples are *instances* grouped into *bags*. In other words, only the bags are independent objects. For example, a set of sub-image of a global large image is a bag (the single entity is the large image composed of instances which are sub-images). In the case of chemoinformatics, the set of the most likely 3D-conformations of a molecule is a bag of instances, and the single entity is the molecule.

Multi-view learning is a setting where the samples are described by heterogeneous sets of features which are different *views* of the data. For instance, a cell line can be described by different views like mRNA-level gene expression, mutation status, methylation status, CN variation and so on. Molecules can be described from different views, like their molecular graph, targets profile, secondary effect profile, ATC code profile and so on. In practice, considering simultaneously heterogeneous descriptions of samples in a single model can be a significant issue.

Active learning is a setting in which the data points from the training set are added one by one, in such a way that each new data point best improves the prediction model being built. This setting is used when it is expensive (in some way) to label data point that are necessary to train the model. This situation is encountered in pharmacological companies. For example, when learning a rule to predict ligands for a protein target, active learning can help to reduce the number of wet lab experiments by guiding which interactions should be tested. A typical active learning strategy is the "Query by Committee" (QBC) strategy that uses an ensemble of machine learning models to make predictions. Such a procedure automatically samples regions of the chemical space where most machine learning models fail to accurately predict a given chemical property [19].

A.1.3 Feature pre-processing and engineering

In most cases, machine learning approaches' performances strongly rely on designing relevant features to describe the data. The design of handcrafted or automatically extracted features is usually referred to as *featurisation*.

On the other hand, representation learning [269] finds a data-driven appropriate representation of data, in order to perform a given machine learning task, rather than to rely on expert-based data-blinded features. In the case of supervised learning, the representation is often learnt in an end-to-end manner, which refers to the joint optimisation of the feature encoder and the prediction model itself. Deep learning-based representation learning is introduced in appendix A.8.

Also, *feature pre-processing* and *feature engineering* are often the main levers to significantly increase the models' performances, more than the choice of an algorithm.

A common pre-processing method is *standardisation* which involves enforcing each feature distribution over the available data to be a Gaussian with zero mean and unit variance. This is achieved by removing the empirical mean value of each feature and by dividing by its empirical standard deviation. Many machine learning estimators may misbehave if features are not standardised. Indeed, if a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly. If there are too many outliers that influence the mean and variance, we can standardise the input using the median instead of the mean, and "3^{rd} quantile - 1^{rst} quantile" value instead of the variance.

Furthermore, standardisation of the output labels y_i is also important when there are several outputs, so that they have the same influence on model evaluation.

Normalisation, another widely used pre-processing procedure, is the process of scaling individual samples to have unit norm. It can be useful when using machine learning algorithms that consider the distance between samples, like the widely used SVM algorithm (see appendix A.5).

Feature selection consists in selecting a restricted number of features that are essential for the problem at hand. It is useful to increase the predictive power of the model and to reduce the computation cost. In particular, when the number of features is much higher than the number of instances, this issue is of vital importance. Multiple strategies can be considered depending on the problem. They often select features based on:

- the most varying features.
- the features most correlated to the output.
- transforming the data to a new space whose dimension is smaller, while keeping the variance of the data as much as possible.
- using feed-forward selection procedure: adding features one by one, selecting those which triggered the largest improvement of model performance.
- using univariate statistical test: only consider features which are not independent from the output, according to a cut-off criterion. Some tests look for linear dependency, such as chi^2-test for classification and F-test for regression. Others look for non-linear dependency, such as mutual information-based test.
- using an embedded feature selection model like lasso or ElasticNet. Models based on matrix factorisation also learn a compact representation of the data.
- computing feature importance, for example via bagging procedures like the randomised lasso or bagging trees, and consider only features whose importance is above a given threshold.

Another feature pre-processing is feature engineering which includes any kind of manual modification and creation of new features. All are, of course, problem dependent. For instance, some continuous features can gain predictive power when binarised. As in the case of the prediction of metro usage, the days of the week could be split into *weekdays* and *weekends* and the hour of the day could be binarised into *working hours* and *not working hours*. Time (or more generally periodic features) is also likely to benefit from a periodic transformation $x \mapsto \sin(\frac{2\pi x}{T})$ (where T is the period). In the case of hours of the day, it is $x \mapsto \sin(\frac{2\pi x}{24})$.

Lastly, missing feature values can be imputed through multiple strategies if they represent an issue. When there are few missing values, the naive approach is to replace a missing feature by the mean, the median or the most frequent values of this feature in the dataset.

A.1.4 Model complexity: bias-variance trade-off and regularisation

To conclude this brief introduction to machine learning approaches, we introduce the notion of bias-variance trade-off.

The overall prediction error made by a model is composed of three components: the *model bias*, *model variance* and irreducible errors (remaining after the two others).

In particular, the mean squared error of the prediction $\hat{y}$ compared to the ground truth y, MSE $= \mathbb{E}(\hat{y}-y)^2$, can be rewritten as MSE $= \text{Var}(\hat{y})+\text{Bias}^2(\hat{y})$. This is referred to as the *bias-variance decomposition*, and illustrates that there is a trade-off between bias and variance.

In statistics, *bias* is the difference between the expected value of an estimator $\hat{y}$ and the actual value being estimated y: $\mathbb{E}(\hat{y}-y)$. Model bias is the error from incorrect assumptions, leading to missing the underlying relationships between inputs and outputs. High bias can cause *underfitting*: the model makes too many mistakes on the training data. A too simple model is likely to have a high bias.

In statistics, the *variance* of an estimator $\hat{y}$ is its deviation from the expected value of an estimator $\mathbb{E}(\hat{y})$: $\mathbb{E}(\hat{y}-\mathbb{E}(\hat{y}))$. Model variance is the error coming from the sensitivity to small fluctuations in the training set. These fluctuations come from the noise, or simply outliers or missing data in the training data (measurement uncertainties for instance). High variance can cause *overfitting*: the model makes few mistake on the training data, but cannot generalise well on unseen data. A too complex model is likely to have a high variance.

In summary, increasing model complexity towards overfitting decreases the bias and increases the variance. Conversely, decreasing model complexity towards underfitting increases the bias and decreases the variance. Model complexity is related to model flexibility and often relies on the number of parameters. Overfitting can be observed when the performance of a model on train set continues to improve with model complexity, while the performance on unseen test data plateaus or declines. This behaviour is illustrated in Fig. A.1.

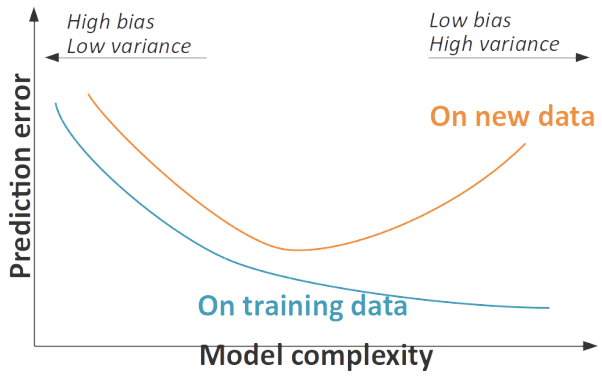


Figure A.1. Bias-variance trade-off. (picture from [364])

In most cases, the training set represents only a small portion of the overall possible data points of the consider problem, and it is critical that the selected model has good generalisation properties. Therefore, the best parameters and hyper-parameters are selected based on their prediction performance on unseen test data.

To improve generalisation, model complexity is usually penalised by adding a contribution to the empirical error which quantifies the model complexity. This allows to decrease model prediction error on unseen data and model complexity simultaneously. An additional hyper-parameter λ often control this trade-off such that the total loss function can be written as: $total\ error = empirical\ error + \lambda * complexity\ penalisation$.

Among equally performing models, we favour simpler models (still not likely to underfit), because they are easier to use (lower computational complexity), easier to train (lower space complexity), easier to interpret, and tend to generalise better. Moreover, the widely accepted Occam's razor principle suggests that simple explanations are more plausible.

In the next sections, we introduce common procedures to select models leading to the best performance and common metrics to assess the performances.

A.2 Evaluation metrics and procedures for supervised learning

A.2.1 Model evaluation

Let us first define the terms: *model evaluation* discusses the metrics used to assess the performance of a model, while *model selection* discusses the process which selects the best model.

Performance metric for regression problems

In the following, we recall the most widely used metrics for assessing performances on a regression task. In the following y^{pred} refers to the predicted values and y^{true} to the actual true values.

- explained variance: quantifies how much of labels variance can be recovered by the model. The best score is 1.0, lower values are worse.

$$\text{explained variance}(y^{true}, y^{pred}) = 1 - \frac{Var(y^{true} - y^{pred})}{Var(y^{true})}$$

- mean squared error (MSE): sums over the data points the squared differences between predicted and the actual outputs. Best value is zero, higher values are worse.

$$\text{MSE}(y^{true}, y^{pred}) = \frac{1}{n_{samples}} \sum_i (y_i^{true} - y^{pred}{i}^2_)$$

- mean squared log error: similar to MSE but the predicted values far from the real values are less penalising the performance.

$$\text{log-MSE}(y^{true}, y^{pred}) = \frac{1}{n_{samples}} \sum_i (log(1 + y_i^{true}) - log(1 + y_i^{pred}))^2$$

- R^2 score, also called "coefficient of determination": considers the distance between the predicted and the true values, relatively to the distance between the true values and their mean. Therefore, a constant model that always predicts the mean value of the output, disregarding the input features, would get a R^2 score of 0.0. The best possible score is 1.0, and it can be negative.

$$R^2(y^{true}, y^{pred}) = 1 - \frac{\sum_i (y_i^{true} - y_i^{pred})^2}{\sum_i (y_i^{true} - mean(y^{true}))^2}$$

Performance metric for a classification problem

A common way of assessing the performance of a classification problem is via its confusion matrix. In the case of binary classification, defined by a positive labelled class P and a negative labelled class N, the confusion matrix is defined by:

	True	Class
Predicted class	Number of **T**rue **N**egatives (TN)	Number of **F**alse **N**egatives (FN)
True class	Number of **F**alse **P**ositives (FP)	Number of **T**rue **P**ositives (TP)

TP is the number of times a positive sample is correctly predicted. TN is the number of times a negative sample is correctly predicted. FP is the number of times a positive sample is incorrectly predicted. FN is the number of times a negative sample is incorrectly predicted.

From the test theory point of view, false positives can also be named type I errors (or false alarms) and false negatives can also be called type II errors (or misses).

From the confusion matrix, we can define a set of metrics to evaluate a classifier.

- Accuracy: is the fraction of correctly predicted samples among all samples in the training set. Best value is 1, and lower is worse.

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN} \in [0, 1]$$

  Selecting the model with the best accuracy may result in high bias, especially when there are not the same number of samples from each class.

- MCC: is the equivalent of the accuracy metric, but it is corrected for unevenly distributed data. Best value is one, and lower is worse.

$$MCC = \frac{TP * TN - FP * FN}{\sqrt{(TP + FP) * (TP + FN) * (TN + FP) * (TN + FN)}} \in [-1, 1]$$

  However, MCC still depends on reliable labelling of the data.

- Precision: refers to the fraction of the correctly predicted positive samples (TP) among all positively predicted ones (overlook the number of FN)

$$\text{Precision} = \frac{TP}{TP + FP} \in [0, 1]$$

  However, precision does not inform us on whether we have missed occurrences of the positive class, which would be reflected by the number of false negatives. Therefore, a very high precision might be associated to limited accuracy.

- Recall: denotes the fraction of correctly positive predicted samples among all true positive samples.

$$\text{Recall} = \frac{TP}{TP + FN} \in [0, 1]$$

  However, it overlooks the number of false positives. Indeed, the simple strategy which consists in always predicting the positive class give perfect recall for that class, because the misclassified points are only captured by the number of false positives.

- F_1 statistic: is the harmonic mean of precision and recall (equal weights to precision and recall), in order to consider both the FPs and FNs. Furthermore, a significant imbalance between precision and recall results in a lower F_1 score. In other words, it is necessary to have both good precision and good recall to have a good F_1 score. The harmonic mean of any data is always upper-bounded by its arithmetic mean. Thus, the F_1 score addresses precision and recall shortcomings simultaneously.

$$F_1 = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

All the metrics defined above measure the performance of a classifier at a selected "classification score" threshold. Indeed, most models assign each data point to a probability, or a classification score, relatively to each class. The higher the score between a sample and a class, the more likely the sample belongs that class. To claim what samples belong to what classes, we must define a threshold such that a sample is assigned to a specific class if the classification score is above this threshold. For instance, when a classifier outputs probabilities, this threshold is usually 0.5 in binary classifiers.

However, the generalisation of the performance over the whole threshold spectrum is relevant, especially to fairly compare the performance of multiple methods. For that purpose, we mainly consider the two following metrics:

- Area under Receiver-Operator Characteristic curve (ROC curve). The ROC curve is a 2D plot in which the horizontal and the vertical axes correspond respectively to FPs rate -number of FP among all true negatives- and the TPs rate -recall-. Each point in the curve reports the FP rate and recall, measured at different arbitrarily selected classification score thresholds.

  By definition, a good model should have a ROC curve localised in the high left corner. A good model should then maximise the area under the ROC curve (ROCAUC). 1 is the best performance, and lower is worse.

Note that two models with the same accuracy but different ROCAUC would mean that the model with higher ROCAUC is more accurate for high probability of positiveness and models are equivalent when considering a positiveness threshold of 0.5.

- Area under the Precision-Recall curve (PR curve). The Precision-Recall curve is a 2D plot where the horizontal and the vertical axes correspond respectively to the precision rate and the recall rate. Each point in the curve reports the precision and recall, measured at different arbitrarily selected classification score thresholds.

  By definition, a good model should have a PR-curve localised in the high right corner. A good model should then maximise the area under the curve; 1 is the best performance, lower is worse.

  A high AUPR (area under the PR curve) score means that, when considering samples with a high probability of positiveness, there are few FP, and TP are recovered. In other words, the probability of positiveness (respectively negativeness) of TP (respectively TN) is higher than the one of FP (resp. FN).

In the case of highly imbalanced datasets (for example, the number of negative examples greatly exceeds that of positive examples), the AUPR score gives a more realistic picture of the performance. Indeed, a large change in the number of false positives can have a small impact on the false positive rate used in the ROCAUC score, whereas precision is sensitive to a large number of negative examples as it compares them to true positives rather than true negatives.

Despite these differences between AUPR and ROCAUC, Davis et al.[365] found an important connection between the ROC space and the PR space. They proved that a curve (associated with a model) dominates in the ROC space if and only if it dominates in the PR space. Conversely, they showed that algorithms that optimise the ROCAUC score are not guaranteed to optimise the AUPR score.

Furthermore, they explained why it is incorrect to linearly interpolate between points to build the PR curve. Indeed, precision does not necessarily change linearly with Recall since FP replaces FN in the denominator of the Precision metric compared to the Recall metric. Based on the proven connection between the ROC space and the PR space, the interpolation of the convex hull in the ROC space (which is the best curve that can be built from a set of ROC points) to the PR space provides the best curve in the PR space, and therefore, an appropriate interpolation.

Among all previously defined classification metrics, we can wonder what is (are) the most relevant one(s) for virtual screening of drugs.

Considering several metrics provides a general idea about the biological relevance of the results of the new method and its added value over the previous approaches.

However, there is no consensus on the best metrics, and most studies use different evaluation metrics. As a result, it is often difficult to compare the performances of methods proposed in different studies.

In the case of hit discovery, reducing the number of false positive is of major importance, in order to reduce the number of experimental tests. Hence, the precision and the PR curves are suitable measures of performance. However, DTI prediction is a retrieval problem which seeks to find positive points among the

set of negative points, but non-interacting drug-target pairs could be pairs for which interaction has not been experimentally proven yet.

Therefore, focusing on avoiding false discoveries could be counterproductive because the aim is to find new interactions among the "non-interacting pairs". Thus, finally, evaluating the performance of DTI classifiers from different angles remains the best choice even if a particular attention may be put on the AUPR score.

Note that more specific ways to measure DTI prediction performance have been proposed in the literature. For instance, some authors attempted to retrieve the DTI prediction made on a given database in other databases or by literature search. An alternative (and costly) way is also to directly test the prediction in-vitro.

A.2.2 Model selection

At first sight, it seems logical to merely divide the available data into two parts to fairly assess and compare performances of several models: one is the training set used to adjust the weights of the model, the other is the test set used to assess the performance we can expect on unseen data. Indeed, a model might work for the training set, but fail to predict on unseen data (see generalisation issues in appendix A.1). Therefore, models' performance is assessed on data which have not been used for training.

However, data should be used in a more efficient way, as they can be scarce or difficult to obtain. Moreover, performance measures are reliable only when the test samples are representative of the whole population. Assessing performances on a particular sub-part of the data may bias the estimation of the performance. Thus, there exists several methods to make the best from the available data: these methods divide the data into several train and test sets in such a way that all samples are used as validation data once.

In particular, cross-validation (CV) is a method which splits the available data into k separate folds. We then successively train k-times on (k-1) folds and validate on the remaining one (see Fig. A.2). Therefore, it estimates the performance on all the available data, while never testing the model on training data. The final performance is the mean of the estimated performances measured on the different folds.

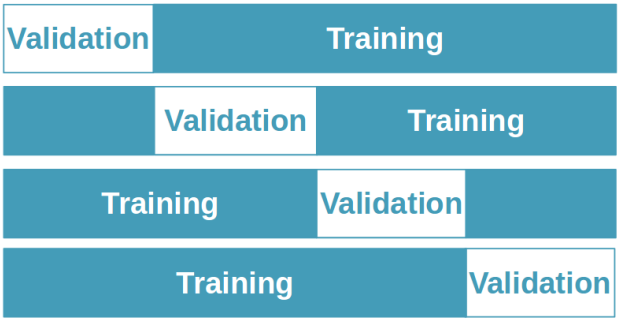


Figure A.2. Cross-Validation with $k = 4$. (picture from [364])

The problem of choosing an appropriate k arises. Considering a to large k may result in highly variable performances, since some of the small folds may contain easily predictable points while others may contain difficult predictable points. In contrary, considering small k may result in inaccurate models as a too small fraction of the available data is used for training.

The specific case of $k = N$, where N is the number of samples in the dataset, is called *leave-one-out cross-validation* (LOO-CV) and may require burdensome computations. However, it is interesting when the number of labelled data is scarce.

Usually, k is set to 5 (at each step, 80% of the available data are used to fit the model, 20% for evaluation), or to 10 (90% for fitting, 10% for evaluating).

In parametrised models, we often chose the parameter which gave, on average, the best performances on all folds of the cross-validation scheme. However, this best parameter may not be suited for new unseen data if the folds are biased compared to new data.

In this case, nested cross-validation is suitable. It consists in considering an *inner* CV nested in an *outer* CV scheme. It consists in a (k-1) folds cross-validation (*inner-CV*) nested in a k folds cross-validation (*outer-CV*). At each step of the outer-CV, the inner-CV is repeated for all considered values of the hyper-parameters. The values leading to the best prediction performance are retained as optimal, and the performance is finally assessed on the outer test fold, which represent the new data unseen when optimising the parameter.

Bootstrap is another widely used procedure to split the data into training and testing set. This method builds training sets with slightly different statistics by randomly sampling with replacement from the available data. For each training set, the samples which have not been picked form the testing set. Typically, training sets are sampled a hundred times.

In the next sections, we introduce machine-learning algorithms and procedures.

A.3 Linear models

Linear models correspond to the class of approaches that build a linear relationship between the inputs and the outputs. In the case of regression, the outputs are learnable linear combination of input features. In the case of classification, the outputs are learnable linear combination of input features transformed into a probability of belonging to each class.

This section is by no means an exhaustive discussion on linear models, but an intuitive introduction. Interested readers can refer to the following textbook [176].

A.3.1 Linear regression

Let us consider the problem in which we want to linearly model the relationship between $Y \in \mathbb{R}^d$ and $X \in \mathbb{R}^p$. Let $w \in \mathbb{R}^p$ be a weight vector and $\sigma^2 > 0$. We make the following assumption:

$$Y \mid X \sim \mathcal{N}(\mathbf{w}^\top X, \sigma^2),$$

which can be rewritten as:

$$Y = \mathbf{w}^\top X + \epsilon,\ \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Note that if there is an offset $w_0 \in \mathbb{R}^p$ such that $Y = \mathbf{w}^\top X + w_0 + \epsilon$, one can redefine a weighting vector $\tilde{\mathbf{w}} \in \mathbb{R}^{p+1}$ such that $Y = \tilde{\mathbf{w}}^\top \begin{pmatrix} X \\ 1 \end{pmatrix} + \epsilon$.

Let the dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$ be independent (a data point y_i is independent of the other $\{y_j\}_{j \neq i}$ given x_i and a model w) and identically distributed ($y_i \mid x_i \sim \mathcal{N}(\mathbf{w}^\top w_i, \sigma^2), \forall i$).

Based on this assumptions, the conditional distribution of all outputs given all inputs is a product of independent terms:

$$p(y_1, \cdots, y_n | x_1, \cdots, x_n; \mathbf{w}, \sigma^2) = \prod_{i=1}^{n} p(y_i | \mathbf{x}_i; \mathbf{w}, \sigma^2).$$

Typically, the logarithm is used to transform the product into a sum. In addition, an optimisation problem is usually set as a minimisation problem. Therefore, we consider the so-called associated log-likelihood of the data which is written as:

$$-l(\mathbf{w}, \sigma^2) = -\sum_{i=1}^{n} \log p(y_i | \mathbf{x}_i) = \frac{n}{2} \log(2\pi\sigma^2) + \frac{1}{2} \sum_{i=1}^{n} \frac{(y_i - \mathbf{w}^\top \mathbf{x}_i)^2}{\sigma^2}$$

Indeed, the model $\mathbf{w}$ (or more precisely, the model weight vector) is usually obtained by maximising the "likelihood", that is, adjusting the weights $\mathbf{w}$ so that the output probabilities $p(y_i|x_i)$ correspond to the true labels y_i. Minimising the log-likelihood with respect to $\mathbf{w}$ can be reformulated as:

$$\hat{\mathbf{w}} = \text{argmin}_{\mathbf{w}} \tfrac{1}{2n} \textstyle\sum_{i=1}^{n} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2 = \text{argmin}_{\mathbf{w}} \tfrac{1}{2n} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2.$$

Thus, optimising a linear regression model is equivalent to minimising the classical sum of square errors.

The function $f : w \mapsto \frac{1}{2n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2$ is equivalent to $\frac{1}{2n}(\mathbf{y}^\top\mathbf{y} - 2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w})$. By definition, this function is strictly convex if and only if its Hessian matrix is invertible. In such a case, $\hat{\mathbf{w}}$ can be directly obtained by setting the gradient of f to 0:

$$\nabla f(\mathbf{w}) = \frac{1}{n}\mathbf{X}^\top(\mathbf{X}\mathbf{w} - \mathbf{y}) = 0 \iff \mathbf{X}^\top\mathbf{X}\mathbf{w} = \mathbf{X}^\top\mathbf{y} \iff \hat{\mathbf{w}} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$$

To ensure that $(\mathbf{X}^\top\mathbf{X})^{-1}$ is invertible, we pre-process the data matrix X to avoid numerical issues. However, this pre-processing does not guarantee that the resulting data matrix is well-conditioned, for example if it is a low rank matrix (i.e. some features are co-linear).

We can also derive σ^2 by minimising the $-l(\mathbf{w}, \sigma^2)$ with respect of σ^2 by setting $\nabla_{\sigma^2} l(\mathbf{w}, \sigma^2)$ to zero. It results in the classical "empirical variance" of a linear model:

$$\hat{\sigma}^2 = \tfrac{1}{n} \textstyle\sum_{i=1}^{n} (y_i - \mathbf{w}^\top \mathbf{x}_i)^2.$$

A.3.2 Logistic regression

Logistic regression linearly models the relationship between $Y \in \{0, 1\}$ and $X \in \mathbb{R}^p$, i.e. segregates the two classes with a linear decision boundary.

First, let Us naively model the conditional outputs by a Bernoulli distribution:

$$y_i|x_i \sim Bernoulli(\theta_i), \ \forall i$$

with the parameter $\theta_i = \sigma(\mathbf{w}^\top \mathbf{x}_i)$, where $\mathbf{w}$ is a weight vector. The sigmoid function σ, displayed in figure A.3, takes values in $[0, 1]$ and is defined by: $\forall z \in \mathbb{R}, \sigma(z) = \frac{1}{1+e^{-z}}$.

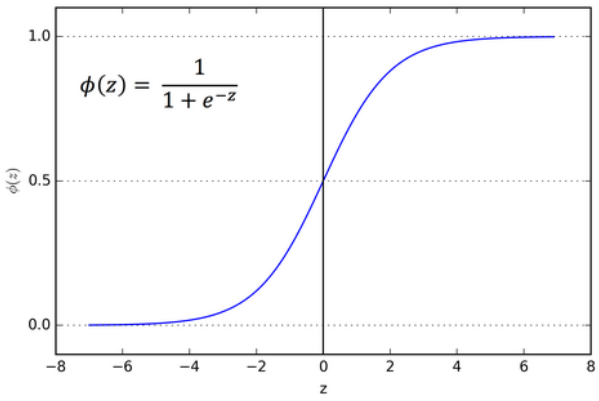


Figure A.3. Sigmoid function. (picture from [366])

Note that the sigmoid function is just transforming a linear combination of the inputs into a probability value (i.e. between 0 and 1). Thus, a sample x_i is assigned to the class 1 (resp. 0) if its linear combination $w^T x_i$ is positive (respectively negative) because its associated probability $\sigma(w^T.x_i)$ is above (respectively below) 0.5.

The sigmoid function has also convenient computational properties. In particular, $\forall z \in \mathbb{R}$, $\sigma(-z) = 1 - \sigma(z)$, and $\sigma'(z) = \sigma(z)(1 - \sigma(z)) = \sigma(z)\sigma(-z)$.

As in the regression case, the weights $\mathbf{w}$ are optimised to minimise the negative log-likelihood on a dataset of i.i.d. random variables $\mathcal{D} = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_n, y_n)\}$, which is given by

$$l(\mathbf{w}) = \sum_{i=1}^{n} y_i \log \sigma(\mathbf{w}^\top \mathbf{x}_i) + (1 - y_i) \log \sigma(-\mathbf{w}^\top \mathbf{x}_i)$$

However, unlike linear regression, there is no closed-form solution. Therefore, gradient descent based algorithm are typically used to find the optimum weights.

A.3.3 Regularised models

As introduced in appendix A.1, the more variables in the model, the more complex it is, and the more likely it overfits the training data. Thus, controlling model complexity usually benefits to the performance on

unseen data. In the case of linear models, as in many cases, regularisation (e.g. reducing model complexity) is performed by adding to the cost function $\mathcal{L}$ a penalty term on the model complexity:

$$\mathcal{L}_{regularised} = \mathcal{L}_{not-regularised} + \lambda * complexity \tag{A.1}$$

In a nutshell, $\lambda > 0$ favours less complex models by trading bias in exchange of smaller variance. The hyper-parameter λ is usually set by cross-validation (see appendix A.2).

Ridge penalty

Among regularised models, the so-called "Ridge" estimator penalises model complexity by minimising adding the L^2 norm of the model weights to the loss. Such a penalty shrinks the value of the weights as it enforces small absolute values. If all weights are small, the modelled function is smooth, potentially improving better the generalisation properties. In particular, correlated variables get similar weights. However, it does not ensure that the resulting model is sparse.

In particular, the Ridge regression estimator is defined as:

$$\hat{w}_{\text{ridge}} = \arg\min_w ||y - w^T.X||_2^2 + \lambda||w||_2^2 \Leftrightarrow \hat{w}_{\text{ridge}} = \arg\min_w ||y - w^T.X||_2^2 \text{ such that } ||w||_2^2 \leq 1$$

The ridge regression estimator has an analytical solution: $\hat{w}_{\text{ridge}} = (X^T X + \lambda I)^{-1} X^T y$.

Lasso penalty

Both the standard linear regression and the Ridge regression can use all available features, which can render the solution hard to interpret. Moreover, we can often often assume that only a subset of key features are necessary to explain data outcome.

Building a model on a sparse selection of essential features is enabled by the Lasso penalty (L^1 regularisation). The Lasso regression estimator is defined as:

$$\hat{w}_{\text{lasso}} = \arg\min_w ||y - w^T.X||_2^2 + \lambda||w||_1 \Leftrightarrow \hat{w}_{\text{lasso}} = \arg\min_w ||y - w^T.X||_2^2 \text{ such that } ||w||_1 \leq 1$$

Unlike the Ridge estimator, there is no explicit solution. We can however transform the Lasso optimisation problem into a quadratic problem which can be solved by standard gradient descent-based algorithms.

To intuitively understand why the Lasso estimator provides a sparse solution, we can look at the feasible set, which is the set where the parameters are enforced to live. The feasible set triggered by $||w||_2 \leq 1$ (Ridge) is a ball, the feasible set triggered by $||w||_1 \leq 1$ (Lasso) is a square. The feasible sets of the L^q norms (e.g. $||w||_q^q = \sum_{i=1}^n w_i^q$) for different values of q are displayed in figure A.4.

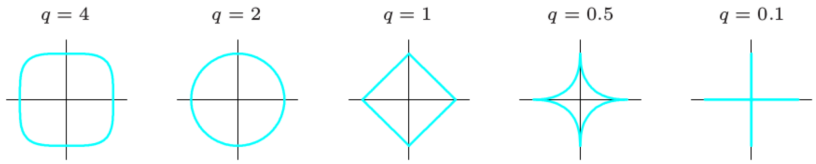


Figure A.4. Boundaries of the feasible set associated with the constraint $||w||_q \leq 1$ for different values of q. (picture from [364])

In the case of the L^1 feasible set, we can observe that the projection of the unconstrained minimum on the feasible set is likely to "hit" a corner of the square, where only one feature has a weight not equal to zero (see Fig. A.5), which explains why the obtained solution is sparse.

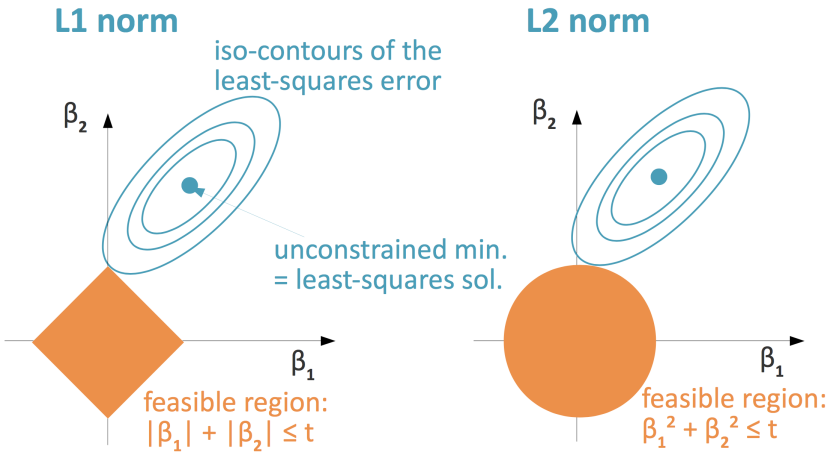


Figure A.5. Comparison of the Lasso-estimator and the Ridge-estimator. (picture from [364])

Data structure-based penalty

In the case where data have a latent structure, we can also enforce the model to fit this structure by penalising the model accordingly.

In particular, when the variables are partitioned into predefined groups of variables that are known or expected to work together (being active or inactive together), we can define a penalty of the form $\sum_{k=1}^{K} \sqrt{p_k}||w_k||_2$ to enforce this structure into the model. Here, K is the number of groups and p_k is the number of features in each group. An example of such partition in a biological problem can be genes belonging to the same biological pathway.

Another example is node classification for graph-structured data, where we can enforce two neighbouring nodes to have similar labels by adding the following penalty to the model: $w^T Lw$ (where L is the Laplacian of the graph).

Finally, in the case of a multi-output prediction task for which all prediction tasks are assumed to use of the same subset of features (although with different weights), we can consider the following penalty to enforce the corresponding multitask sparse solution: $||w||_{21} = \sum_i \sqrt{\sum_j w_{ij}^2}$.

A.4 Tree based models

Tree-based models are distinct from the previously discussed models in that they are non-parametric. This means that the model is not simply a mathematical function with a fixed form, and that the size of the model changes depending on the size of the data. Non-parametric methods could be said to "let the data talking".

This section is an introduction on tree-based models, and interested readers can refer to the following textbooks [176].

A.4.1 Decision Tree

Let us consider a dataset $\mathcal{D} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$.

Decision trees are based on hierarchically structured decision-making nodes. To learn a decision tree, we need to choose the rules for defining splitting criteria (i.e. decision criteria at every node), stopping growing the tree, and pruning the tree depending on a cost-complexity trade-off.

Splitting criteria

When growing a tree, all nodes are split according to the "best" possible splits at each step (i.e. tree depth). A split relies on a feature value f_i such that, a sample (whose feature vector is x) is assigned to one of the children node if $x_i < f_i$, or otherwise to the other children node.

The quality of a split is assessed by a score that evaluates the reduction in uncertainty, e.g. the variance of sample label values falling in the two children nodes. Thus, the best split is the one that minimises this uncertainty.

In the case of classification, both the classification error or the entropy can be considered. The classification error is $1 - \max_k(\hat{p}_k)$, where $\hat{p}_k$ is the proportion of training instances from class k, and quantifies the minimum probability that a training point is to be misclassified. The entropy $-\sum_k \hat{p}_k \log_2(\hat{p}_k)$ is a measure of lack of information and thus, quantifies the information gain provided by a split. However, the Gini impurity criterion (i.e. $\sum_{k=1}^K \hat{p}_k(1-\hat{p}_k)$) is usually considered. It is a measure of how often a randomly chosen sample would be incorrectly labelled if it was randomly labelled according to the distribution of labels in the subset.

Stopping criteria

Furthermore, a strategy deciding when to stop growing a tree is required.

Two strategies are usually considered:

- setting the *Maximum Tree Depth*. This is the maximum number of nodes from the root node of the tree. Once the maximum depth of the tree is met, splitting and adding new nodes is stopped. Deeper trees are complex and likely to overfit the training data.
- setting the *Minimum Node Records*. This is the minimum number of training samples that a given node is responsible for. Below this minimum, we must stop splitting and adding new nodes. Nodes that account for too few training patterns are expected to be too specific and are likely to overfit the training data.

Pruning criteria

Note we can we also prune the tree afterwards using cost-complexity pruning strategies which aim at finding an efficient sub-tree, in order to balance model complexity and classification performance. Thus, it can be seen as a form of regularisation.

Decision trees are easily interpretable as they can be displayed graphically and mimic human decision making. In addition, they naturally handle quantitative variables and multi-class problems.

However, they are known to exhibit poor predictive accuracy in general. Intuitively, a decision tree has a low bias since the conditions that are checked at each node becomes multiplicative, so that the tree is continuously narrowing down on the data. In other words, the tree is likely to be highly tuned to the training set. A decision tree also has a high variance, since small changes in input variable values might result in very different tree structure. Forests of decision trees fitted on the data with different statistics are usually considered to fix this high variance issue.

A.4.2 Bagging trees

In the *bagging trees algorithm*, different trees are fitted on different subsets of the original data built by sampling data points with replacement from the original data. In other words, the bagging procedure considers many predictors by bootstrapping the data (randomly sub-sample with replacement the dataset several times).

Bagging is efficient when used with a low bias and high variance base estimator, which is the case of decision trees, as mentioned before. Indeed, by averaging such estimators, we get a smoother version (low variance), still centred around the true density (low bias).

A.4.3 Random Forest

In practice, bagging alone is typically not enough. In order to get a better reduction in variance, aggregated models need to be uncorrelated, so that they make "different errors". Bagging usually results in highly correlated models that make the same errors, and therefore, it does not reduce the variance of the combination of the base predictors.

The idea behind random forests is improving the variance reduction of bagging by reducing the correlation between the trees, without increasing the bias too much. Variance reduction is achieved during the tree-growing process through random selection of the input variables: before each split, q (out of the p available) input variables are selected at random as candidates for splitting. A value of $q = \sqrt{p}$ is typically chosen.

In other words, the decorrelation of the individual trees is two-fold. First, the training set is sampled (with replacement) to create M bootstrapped training sets of size N used in an ensemble model. This is known as bagging, e.g. bootstrapping and aggregating. Secondly, each learner is fitted on a random subset of the available features.

The final prediction is obtained by aggregating the predictions of individual trees. This implies averaging predictions for a regression problem, averaging probability prediction for a classification problem. In practice, this method has excellent predictive power.

Note that bagging trees and random forests are useful when using a large number of trees (several hundreds). They are thus computationally intensive, especially when the number of features is large.

Finally, let us more formally discuss the intuition behind the fact that both bagging and sampling features reduce the variance of predictions of the ensemble model.

First, recall that,

$$\begin{aligned}\mathrm{Var}[X+Y] &= \mathbb{E}[(X+Y)^2] - \mathbb{E}[X+Y]^2 \\ &= \mathbb{E}[X^2 + 2XY + Y^2] - \mathbb{E}X^2 - 2\mathbb{E}[X]\mathbb{E}[Y] - \mathbb{E}[Y]^2 \\ &= \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}[X,Y]\end{aligned}$$

We can then decompose the variance of the aggregated prediction $\frac{1}{M}\sum_i^M f_i$ of M base learners on x_i following:

$$\mathrm{Var}\Big[\frac{1}{M}\sum_i f_i\Big] = \frac{1}{M^2}\sum_i \mathrm{Var}[f_i] + 2\frac{1}{M^2}\sum_i\sum_{j\neq i}\mathrm{Cov}(f_i, f_j).$$

We can consider that each learner has the same variance σ^2. The covariance between the prediction of two base learners i and j can be written as $\mathrm{Cov}(f_i, f_j) = \rho\sigma^2\ \forall(i,j)\ s.t.\ i \neq j$ where ρ is a positive real number quantifying the correlation between any two learners (on average, all pairs of learners are considered to be correlated the same way). Then,

$$\mathrm{Var}\Big[\frac{1}{M}\sum_i f_i\Big] = \frac{\sigma^2}{M} + \frac{\rho(M-1)}{M}\sigma^2.$$

Thus, the variance σ^2 of individual learners is reduced by a ratio of $M/(1+\rho(M-1))$. Moreover, the less the learners are correlated (e.g., the lower ρ), the more the variance of the ensemble model is reduced.

A.5 Similarity based models

Kernel methods have been widely considered for the last 15 years and have been found extremely successful in numerous applications and enjoy great popularity in the machine-learning community [169, 33, 367].

Beyond this success, kernel methods have a two-fold motivation. First, data are not often described easily with real numbers, and kernel methods enable line fitting algorithm without any assumptions regarding the type of data, since the predictor takes as input the pairwise similarity matrix, with the constraint of being positive definite. Second, a kernel corresponds to a specific and potentially non-linear transformation of the input data, and kernel methods are particularly efficient when transforming the data into a space where the prediction problem is easier to solve. In addition, there is a total modularity between the choice of the algorithm and the choice of similarity, i.e. kernel. Importantly, kernel methods rely on a strong and well established mathematical theory, which is a competitive property compared to random forests and deep learning algorithms.

The main issue of this framework is its poor scalability. Indeed, kernels are $n * n$ matrices, n being the number of samples. Improving the scalability of kernel methods is an active area of research.

This section aims at introducing the only main results on kernel methods needed for this thesis. The mathematical proofs are not to be reported, and interested readers may refer to a more exhaustive presentation [176].

A.5.1 Kernels and Reproducing Kernel Hilbert Space

In a mathematical language, a valid kernel K, known as positive definite kernel, is a symmetric positive definite functional $\mathcal{X} * \mathcal{X} \to \mathbf{R}$. It can be viewed merely as a similarity measure. For instance, the simplest p.d. kernel for vector $x \in \mathbb{R}^d$ is $K(x, x') = \langle x, x' \rangle_{\mathbb{R}^d}$.

"Symmetric" and "positive definite" refer to the following properties:

$$\text{K is symmetric } \Leftrightarrow \forall (x, x^{'}) \in \mathcal{X}^2,\ K(x, x^{'}) = K(x^{'}, x);$$

$$\text{K is positive definite } \Leftrightarrow \forall (x_1, \cdots, x_N) \in \mathcal{X}^N \text{ and } (a_1, \cdots, a_N) \in \mathbb{R}^N,\ \sum_{i=1}^{N} \sum_{j=1}^{N} a_i a_j K(x_i, x_j) \geq 0.$$

The Aronszajn theorem states that there is an equivalence between a valid kernel and a transformation ϕ from the original space $\mathcal{X}$ to a Hilbert space $\mathcal{H}_{\phi(\mathcal{X})}$, such that K is the scalar product in this Hilbert space.

$$\text{K is a valid kernel on } \mathcal{X} \Leftrightarrow \exists \text{ a mapping } \phi : \mathcal{X} \to \phi(\mathcal{X}) \text{ and a Hilbert space } \mathcal{H}_{\phi(\mathcal{X})} \text{ such that}$$
$$\forall x, x' \in \mathcal{X} :\ K(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}_{\phi(\mathcal{X})}}$$

In addition, a p.d. kernel is uniquely linked with a Hilbert space of functions $\mathcal{H}_K$, generally called a Reproducing Kernel Hilbert Space (RKHS).

The RKHS $\mathcal{H}_K$ is the *continuous* space of functions from $\mathcal{X}$ to $\mathbb{R}$.

$$\mathcal{H}_K \subset \mathbb{R}^{\mathcal{X}} \text{ is a RKHS } \Leftrightarrow \forall x \in \mathcal{X}, \left| F: \begin{array}{l} \mathcal{H}_K \to \mathbb{R} \\ f \to f(x) \end{array} \right. \text{ is continuous}$$

The terms "reproducing kernel" refers to the fact that the point-wise evaluation of any function $f \in \mathcal{H}_K$ at any $x \in \mathcal{X}$ can be written as the projection of f on $K_x : t \in \mathcal{X} \to K(t,x) \in \mathbb{R}$.

$$K : \mathcal{X}^2 \to \mathbb{R} \text{ is called a reproducing kernel of } \mathcal{H}_K \text{ if } \left| \begin{array}{c} \mathcal{H} \text{ contains all function of the form:} \\ \forall x \in \mathcal{X}, \forall f \in \mathcal{H}_K,\ f(x) =< f, K_x >_{\mathcal{H}_K} \\ \text{with: } \forall x \in \mathcal{X},\ K_x : t \to K(x,t) \end{array} \right.$$

A.5.2 Kernel trick

The kernel trick states that any algorithm processing finite-dimensional vectors that are expressed only in terms of pairwise inner products, can be applied to potentially infinite-dimensional vectors in the feature space of a p.d. kernel, by replacing each inner product evaluation by a kernel evaluation.

It is intuitive because a kernel is precisely the inner product in its associated feature space. This trick has important practical applications as shown below.

Two straightforward, practical and handy application of the kernel trick is the kernel-based centring and normalisation operations operated in the feature space. First, note that the mean of data points $x_1, \ldots, x_n$ in the feature space is: $\mu = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)$. Thus, centring the data in the feature space be expressed only in terms of kernels, as shown below:

$$\begin{aligned} K^C_{i,j} &=< \phi(x_i) - \mu, \phi(x_j) - \mu >_h \\ &=< \phi(x_i), \phi(x_j) >_h - < \mu, \phi(x_i) + \phi(x_j) >_h + < \mu, \mu >_h \\ &= K_{i,j} - \frac{1}{n}\sum_{k=1}^{n}(K_{i,k} + K_{j,k} + \frac{1}{n^2}\sum_{k,l=1}^{n} K_{k,l}) \text{ since } \mu = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i) \end{aligned}$$

Therefore, the kernel K^{Center} of the data centered in the feature space is:

$$\begin{aligned} K^{Center} &= K - UK - KU + UKU \text{ where } U_{i,j} = \frac{1}{n} \text{ for } 1 \leq i,j \leq n \\ &= (I - U)K(I - U) \end{aligned}$$

Normalising the data in the feature space can also be expressed only in terms of kernels, as shown in below:

$$\begin{aligned} K^{Norm}_{i,j} &=< \phi(x_i)/||\phi(x_i)||_h, \phi(x_j)/||\phi(x_j)||_h >_h \\ &= \frac{1}{\sqrt{< \phi(x_i), \phi(x_i) >_h < \phi(x_j), \phi(x_j) >_h}} < \phi(x_i), \phi(x_j) >_h \\ &= \frac{K(x_i, x_j)}{\sqrt{K(x_i, x_i)K(x_j, x_j)}} \end{aligned}$$

A.5.3 Representer Theorem

The Representer theorem states that the solutions to a wide range of optimisation problems have the form of a sum of kernel functions.

More formally, let $\mathcal{X}$ be a set endowed with a valid kernel K and $\mathcal{H}_K$ its corresponding RKHS. Let S be a finite set of data points: $S = \{x_1, \cdots, x_n\} \subset \mathcal{X}$. Let $\Psi : \mathbb{R}^{n+1} \to \mathbb{R}$ be a function of $n+1$ variables, strictly increasing for the last variable. In these conditions, the representer theorem states that any solution to the following optimisation problem:

$$\min_{f \in \mathcal{H}_K} \Psi(f(x_1), \cdots, f(x_n), ||f||_{\mathcal{H}_K})$$

has a solution of the form:

$$\forall x \in \mathcal{X},\ f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x).$$

In the case of machine-learning, the typical form of Ψ is ($\mathcal{L}$ being a loss function):

$$\min_{f \in \mathcal{H}_K} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(y_i, f(x_i)) + \lambda ||f||^2_{\mathcal{H}_K}.$$

Then, if f is the solution to this classic optimisation problem:

$$f(x_j) = \sum_{i=1}^{n} \alpha_i K(x_i, x_j) = [K\alpha]_j$$
$$||f||^2_{\mathcal{H}_K} = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) = \alpha^T K \alpha$$

The representer theorem is rather intuitive, because if $f \in \mathcal{H}_K$ then $\forall x,\ f(x) =< K_x, f >_{\mathcal{H}_K}$, so that, for a finite subset of points $(x_1, \cdots, x_n)$, the orthogonal part of f with respect to K_{x_i}s is useless to explain the data.

Furthermore, minimising $||f||_{\mathcal{H}_K}$ has a regularisation effect because it penalises non-smooth solutions. Besides, thanks to the representer theorem, we know that the solution lies in a subspace of dimension n (n is the number of data point), even if the feature space is of infinite dimension.

A.5.4 Kernel Ridge regression

Let us write the input space $\mathcal{X}$, and the output space $\mathcal{Y} \subset \mathbb{R}$. We consider an RKHS $\mathcal{H}_K$ and its associated kernel K on $\mathcal{X}$.

Kernel Ridge Regression (KRR) optimisation problem is defined as:

$$\hat{f} = \operatorname*{argmin}_{f \in \mathcal{H}_K} \frac{1}{n} \sum_{i=1}^{n} (y_i - f(x_i))^2 + \lambda ||f||^2_{\mathcal{H}_K} \tag{A.2}$$

The Representer theorem gives the form of the solution: $\hat{f} = \sum_{i=1}^{n} \alpha_i K(x_i, x)$. Thus, we can rewrite the KRR optimisation problem as:

$$\operatorname*{argmin}_{\alpha \in \mathbb{R}^n} \frac{1}{n} (K\alpha - Y)^T (K\alpha - Y) + \lambda \alpha^T K \alpha \tag{A.3}$$

It has a unique closed form solution as:

$$\alpha = (K + \lambda I)^{-1} Y \tag{A.4}$$

Note that if we choose the linear kernel $K(x, y) = x^T y$, the solution of KRR is equivalent to that of the standard ridge regression.

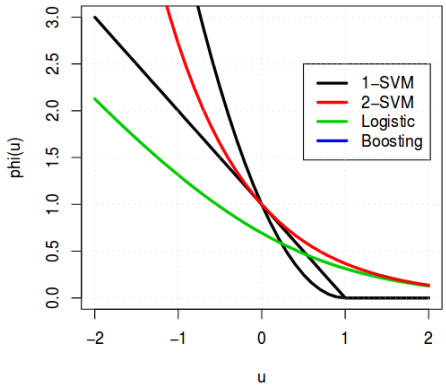
Furthermore, we can also choose other loss functions for regression, like the Huber loss, which is less sensitive to outliers.

A.5.5 Large Margin Classifier (general framework for classification with kernels)

In the case of binary classification $\mathcal{Y} \in \{-1, 1\}$, the margin between the predictions $f(x)$ and the actual labels y is defined as $y^T.f(x)$. A general class of loss functions for binary classification can be derived by non-increasing functions of the margin:

$$\mathcal{L}(f(x), y) = \varphi(yf(x)), \text{ with } \varphi \text{ non-increasing} \tag{A.5}$$

Fig A.6 displays typical margin-based loss functions for classification.



Method	$\varphi(u)$
Kernel logistic regression	$\log(1 + e^{-u})$
Support vector machine (1-SVM)	$\max(1 - u, 0)$
Support vector machine (2-SVM)	$\max(1 - u, 0)^2$
Boosting	e^{-u}

Figure A.6. Margin-based loss functions φ for classification. (picture from [368])

An intuitive large margin-based classifier is the Kernel Logistic Regression (KLR) optimisation problem, that is written as:

$$\hat{f} = \underset{f \in \mathcal{H}_K}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} log(1 + e^{-y_i f(x_i)}) + \lambda ||f||^2_{\mathcal{H}_K} \tag{A.6}$$

The representer theorem states that the form of the solution is: $\hat{f} = \sum_{i=1}^{n} \alpha_i K(x_i, x)$. Thus, we can rewrite the KLR optimisation problem as:

$$\underset{\alpha \in \mathbb{R}^n}{\operatorname{argmin}} J(\alpha) = \frac{1}{n} log(1 + e^{-y_i [K\alpha]_i}) + \frac{\lambda}{2} \alpha^T K \alpha \tag{A.7}$$

This problem has no closed form solution, but is a smooth convex optimisation problem so that it can be solved efficiently by many numerical methods.

A.5.6 Support Vector Machines (SVM)

SVM is an extremely popular algorithm. It belongs to kernel large margin classifiers. Its associated loss is the "Huber loss" (see Fig. A.6).

Let us consider a set of labelled samples $S = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ where $(x_i, y_i) \in \mathcal{X} \times \{-1, +1\}$. For example, the data points x_i represent molecules, and their labels y_i are equal to $+1$ for compounds that bind to a given protein and -1 for compounds that don't.

In the simplest case where the two classes of data points are linearly separable, Support Vector Machines [169] (SVM) is an algorithm that learns to separate these two classes based on the optimal hyperplane separating the two classes.

Such hyperplane H is defined by a normal vector w and a constant b such that: $\forall x \in H,\ \langle w, x \rangle + b = 0$. Among the infinity of possible separating hyperplanes, the optimal hyperplane maximises the "margin". This margin is defined as the closest distance from the hyperplane to any of the data points. It can be shown that the search for this optimal hyperplane can be formulated by the optimisation problem stated in eq. A.8.

$$\underset{w,b}{\operatorname{argmin}} \ ||w||^2 \tag{A.8a}$$

$$\text{subject to } y_i \langle w, x_i \rangle + b \geq 1, \forall i = 1, \ldots, N. \tag{A.8b}$$

Indeed, the solution of this problem is the hyperplane maximising its distance to the closest data points, equal to $2/||w||^2$. The constraint ($y_i \langle w, x_i \rangle + b \geq 1\ \forall i$) states that the hyperplane separated perfectly negative from positive samples.

Then, the decision function f classifies any new point x depending on its position to the hyperplane, i.e. based on the sign of $f(x) = \langle w, x \rangle + b$.

The optimisation problem in eq. A.8 is strictly convex and admits a unique solution. The Lagrangian associated to the optimisation problem leads to the equivalent dual problem stated in eq. A.9, in which the coefficients α_i are known as the Lagrange multipliers associated to the constraints $y_i \langle w, x_i \rangle + b \geq 1$).

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{A.9a}$$

$$\text{subject to } \alpha_i \geq 0, \forall i \tag{A.9b}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{A.9c}$$

In practice, this quadratic problem can be solved efficiently using a dedicated algorithm, known as Sequential Minimal Optimisation (SMO) [154].

When the optimum α^* is met, the decision function allowing predictions for any new point x depends on its position with respect to the hyperplane :

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i^* y_i \langle x, x_i \rangle + b^*\right) \tag{A.10}$$

However, the two classes of data points may not be linearly separable. In such situations, kernel methods are widely-used sets of techniques that allow adapting linear methods to non-linear models. Let us consider a semi-definite positive kernel function $K : X \times X \to \mathbb{R}$. The Mercer theorem states that there exists a non-linear function $\phi : X \to \mathcal{H}$ that maps data points in X into a high dimensional feature Hilbert space $\mathcal{H}$, where K can be expressed as a scalar product: $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle_{\mathcal{H}}$.

In practice, $\mathcal{H}$ is often $\mathbb{R}^d$. Although the two classes of data points might not be linearly separable in X, they might become linearly separable in the high dimensional space $\mathcal{H}$ where the SVM is solved. Since the images of the data point $\phi(x_i)$ are used only in scalar products, the principle of kernel trick is that finding the α_i coefficients to solve the SVM can be done by replacing all occurrences of the scalar product $\langle \phi(x_i), \phi(x_j) \rangle_H$ by the kernel function $k(x_i, x_j)$ (see eq. A.11)

$$\alpha^* = \underset{\alpha}{\text{argmin}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j k(x_i, x_j) \tag{A.11a}$$

$$\text{subject to } \alpha_i \geq 0, \forall i \tag{A.11b}$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \tag{A.11c}$$

In other words, finding the separating hyperplane in H does not require explicit definition of the nonlinear mapping function ϕ or calculation of the image vectors $\phi(x_i)$.

Then, the label of a new data point x is predicted by a function $f(x)$ defined as:

$$f(x) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i^* y_i k(x, x_i) + b^*\right) \tag{A.12}$$

In the case where the two classes of points are not linearly separable even while considering a non-linearly transformation of the data via a specific kernel, we need to allow some of the training points to be misclassified, i.e. to be on the side of the separating hyperplane corresponding to points affected to the opposite label.

To this end, we introduce penalty terms $\epsilon_n (\forall n = 1, \dots N)$, also called slacked variables, defined by: $\epsilon_n = 0$ for data points that are in the correct margin boundary and $\epsilon_n = |y_n - (\langle w, x_n \rangle + b)|$ for misclassified points.

Points on the decision boundary have $\epsilon_n = 1$, and misclassified points would be penalised by $\epsilon_n > 1$, proportionally to their distance to the separating hyperplane. Thus, the penalty terms can be written as $\epsilon_n = max(0, 1 - y_n(\langle w, x_n \rangle + b))$.

Then, the exact classification constraints of equation A.8b are replaced by $y_i\langle w, x_i\rangle + b \geq 1 - \epsilon_i$. In addition, the penalty terms must satisfy $\epsilon_n \geq 0 \ \forall n = 1, \dots N$.

The new objective function displayed in eq. A.13 aims both at maximising the margin and minimising the penalty terms, i.e. minimising the number of misclassified points.

$$\underset{w,b,\epsilon}{\text{argmin}} \ ||w||^2 + C\sum_{i=1}^{N} \epsilon_i \tag{A.13a}$$

$$\text{subject to } y_i\langle w, x_i\rangle + b \geq 1 - \epsilon_i, \forall i = 1, \dots, N, \tag{A.13b}$$

$$\epsilon_i \geq 0, i = 1, \dots, N. \tag{A.13c}$$

The parameter C in the objective function of equation A.13a introduces a trade-off between maximisation of the margin, expressed by the term $\frac{1}{2}||w||^2$, and classification error on the training set, expressed by the penalty terms. This parameter is usually determined by cross-validation.

As for the separable case, SVM in the non-separable case can also be solved in the dual space using a kernel instead of feature vectors.

A.6 Unsupervised learning

Unsupervised learning is the setting in which no labels are assigned to the data points. In this case, we can seek for groups of data points within their overall distribution, which is called clustering. On another hand, we can also estimate the distribution of the data, which is called density estimation. Also, we can project the data on a lower dimensional space (dimensionality reduction).

This section is only a brief introduction and interested readers can refer to the following textbook [176].

A.6.1 Dimensionality reduction

Dimensionality reduction is a well-developed area. There are numerous techniques and variants. Some of the most popular ones are the Principal Component Analysis (PCA), Non-negative matrix factorisation (NMF), auto-encoder and more recently t-Distributed Stochastic Neighbour Embedding (t-SNE) and U-map. Auto-encoders use artificial neural networks that we present in appendix A.8.2.

There is no way to map high-dimensional data into a low dimensional space such that all the structure of the data is preserved. None of these techniques is better than the others, but they are complementary: PCA and NMF try to preserve the linear structure, t-SNE tries to preserve topology (neighbourhood structure); U-map tries to preserve both the proximity of objects like t-SNE but also the remoteness of objects.

Principal component analysis

PCA is the classic and historical method for dimensionality reduction. It linearly projects the data on a lower dimensional space such that the variance of the projected data is maximised. It is a linear projection because the vectors of the basis of this space are linear combinations of the basis of the original space. Equivalently, this linear projection also minimises the average projection cost, i.e. the mean squared distance between data points in the original space and their projection to the principal space.

By definition, PCA is a variance maximising exercise. Hence, it requires the data to be centred and standardised before we can apply it correctly.

Let us consider the projection of the data on a vector $\mathbf{u}_1$. We define the following optimisation problem:

$$\text{Find } \mathbf{u}_1 \in \mathbb{R}^d \text{ such that } \text{Var}(x^T\mathbf{u}_1) \text{ is maximal, with } ||\mathbf{u}_1|| = 1.$$

As the data is centered i.e. $(\frac{1}{N}\sum_{n=1}^{N} x_n = 0)$ the empirical variance is

$$\text{Var}(x^T\mathbf{u}_1) = \frac{1}{N}\sum_{n=1}^{N}\{\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\bar{\mathbf{x}}\}^2 = \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 \ ,$$

where $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T$ is the data covariance matrix.

The optimisation problem can be rephrased as: the vector $\mathbf{u}_1$ is optimised such that the variance of the projected data is maximised under the constraint that $||\mathbf{u}_1|| = 1$. The Lagragian problem associated with this optimisation problem is:

$$\mathbf{u}_1 = \arg\max_{\mathbf{u}_1} \mathbf{u}_1^T S\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}_1)$$

If we set the derivative of the objective function of the Lagrangian problem, it results in:

$$\nabla_{\mathbf{u}_1}\Big(\mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 + \lambda_1(1 - \mathbf{u}_1^T\mathbf{u}_1)\Big) = 0 \Leftrightarrow \mathbf{S}\mathbf{u}_1 = \lambda_1\mathbf{u}_1 \Leftrightarrow \mathbf{u}_1^T\mathbf{S}\mathbf{u}_1 = \lambda_1$$

Therefore, the variance of the projected data is maximum when we projection direction $\mathbf{u}_1$ is the eigenvector associated with the largest eigenvalue λ_1 of the empirical variance matrix. By induction, it generalises to the projection on a space of dimension M by considering the eigenvectors associated with the M largest eigenvalues.

In other words, the M dimensional vector space resulting from the PCA projection is defined by the M eigenvectors of M largest eigenvalues of the data covariance matrix.

A.6.2 Clustering

There is also a wide variety of complementary clustering approaches. We focus now on two simple methods used in this thesis, namely the *K-means* and *hierarchical clustering* methods.

K-means clustering

For a set of observed data points $x_i \in \mathbb{R}^p$ $i \in \{1, ..., n\}$, the K-means algorithm clusters the data by minimising the *distortion* $J(\mu, z)$ defined by:

$$J(\mu, z) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \|x_i - \mu_k\|^2 \tag{A.14}$$

The $\mu_k \in \mathbb{R}^p$, $k \in \{1, ..., K\}$ are the centre of K clusters (K being a hyperparameter), and z_i^k are indicator variables associated to x_i such that $z_i^k = 1$ if x_i belongs to the cluster k, and $z_i^k = 0$ otherwise. Thus, the distortion quantifies the distance between each data point and the centre of its cluster.

It is a non-convex objective function, and therefore, it has the problem of initialisation. Usually, centres are initialised by uniformly sampling among the data points. The optimisation proceeds with an alternating minimisation, as shown below.

Algorithm 2 K-means algorithm

Require:

Set of observations: $x_i \in \mathbb{R}^p$, $i \in \{1, ..., n\}$; cluster centre vectors: $\mu_k \in \mathbb{R}^p$, $k \in \{1, ..., K\}$; indicator variables z_i^k associated with x_i such that $z_i^k = 1$ if x_i belongs to the cluster k, and $z_i^k = 0$ otherwise.

Objective: Identify K clusters minimising the distortion $J(\mu, z) = \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \|x_i - \mu_k\|^2$..

repeat

Step 1 : minimisation of J with respect to z: $z_i^k = 1$ if $\| x_i - \mu_k \|^2 = \min_s \| x_i - \mu_s \|^2$.

In other words, we associate each x_i to the neerest cluster

Step 2 : we minimise J with respect to μ: $\mu_k = \frac{\sum_i z_i^k x_i}{\sum_i z_i^k}$.

μ_k is obtained by setting to zero the k-th coordinate of the gradient of J with respect to μ, which is:

$$\nabla_{\mu_k} J = -2 \sum_i z_i^k (x_i - \mu_k)$$

until Convergence (e.g. $J < \epsilon$ with $\epsilon > 0$)

Hierarchical clustering

Agglomerative Hierarchical clustering is another clustering algorithm. It starts with n clusters (each data point being its cluster) and merges them successively two by two. More precisely, we need a criteria to define which are the two closest clusters in order to successively merged them step by step. Several inter-cluster distances are usually considered. Single linkage considers the inter-cluster distance as the distance between the closest points of the two clusters. Average linkage considers the inter-cluster distance as being the distances between all observations of the cluster pair. Ward linkage merges the two clusters for which the resulting cluster's internal sum of distance (internal compactness) is minimum. In other words, this strategy merges clusters to minimise the intra-cluster variance, similarly to the K-means objective function.

Evaluation of clustering

Clustering quality can be assessed by several means. When it is possible, clustering quality can be assessed qualitatively by manually checking if the clusters make sense.

There are also scores to assess clustering quality like the *silhouette score*. It compares the average intra-cluster distance with the inter-cluster distance. Indeed, a "good" clustering is expected to find compact clusters different from each other. The silhouette score can vary from -1 (inter-cluster distance is 0) to 1 (average intra-cluster distance is 0).

A.7 Multitask learning

A.7.1 The singletask and multitask learning frameworks

In singletask learning, a model makes predictions for one specific question, for instance predicting interactions between ligands and a given protein. Multitask learning addresses multiple questions simultaneously within one model. For example, a multitask drug virtual screening model predicts interactions between ligands with several proteins. The idea behind multitask learning is to improve the performance of all tasks, by leveraging and exploiting some shared knowledge.

Multitask learning has several advantages. First, it improves generalisation performance by leveraging the domain-specific information contained in the training samples of related tasks [369]. Uncorrelated tasks might even improve generalisation by acting as a source of noise. Second, sharing data between related tasks allows to increase the train set available for each task. Thus, it is expected to increase the statistical power. While these mechanisms (data augmentation and regularisation) were identified using artificial neuron networks [370], they are believed to apply to the whole multitask framework.

A.7.2 Tasks similarity

In general, tasks are intended to help each other when they are related in some ways. Tasks can be related intuitively, like when a task's output gives a direct information about another task (for example, predicting whether a name is present in a sentence or not can be an auxiliary task for name error detection [371]).

Beyond such a qualitative discussion, it is important to quantitatively assess task similarity, which is an ill-defined concept. To this end, similar tasks can be theoretically defined as tasks generated from a fixed probability distribution $p(y|x)$ [372]. It is a very strong assumption and we expect multitask learning to work for more loosely related tasks. Besides, several definitions of "task relatedness" has been proposed but none can be used directly in practice or does not encode all possibilities of task relatedness [353]. In the most general case, we can assess that two tasks are related when solving them jointly with a multitask model leads to better performance than solving the two tasks individually with two singletask models.

When task descriptors are available, tasks similarities can be evaluated by comparing their descriptors [373, 374]

A.7.3 Transfer learning

In the literature, we often find the similar yet distinct terms *multitask* and *transfer* learning. Multitask learning shares knowledge between tasks in order to improve prediction performances of all tasks, whereas *transfer learning* refers to transferring knowledge from a *source* task to a *target* task. In other words, the knowledge sharing in transfer learning aims at improving learning of the target task only. In particular, negative transfer (resp. positive transfer) may occurs when knowledge sharing decreases (respectively increases) the performances on the target task.

Transfer learning and multitask learning can both be sub-divided into two main categories [375]: "parameter-based" and "representation-based" (or equivalently "feature-based") transfer learning. Note that there is another subcategory of transfer learning referred to as "instance-based transfer learning", which considers the data as the medium for sharing knowledge. More precisely, specific data from the source task are selected to improve the model fitted on the target task [376].

A.7.4 Feature-based multitask and transfer learning without tasks descriptors

Feature-based transfer learning considers the data description as a medium to share knowledge. It is also called representation-based transfer learning, because it usually relies on learning a common feature representation of data for all tasks. Multitask lasso [377] (see appendix A.3), as well as shallow [369] and deep [378] neuron networks fall into this category.

Multitask lasso [379] is commonly considered as a baseline for multitask feature-based multitask learning. It assumes that tasks are related in that their outputs mainly depend on a common sparse set of features. We introduced multitask linear models in appendix A.3. To benefit from knowledge sharing with MT lasso while allowing task-specific models to rely on task-specific features, Jalali et al. [380] proposed a combination of block-wise sparse regularisation and task-wise sparse regularisation. More precisely, they decomposed the matrix of tasks' feature weights into two matrices, one shared by all tasks, and one task-specific, respectively regularised with l1/l2 norm (MT lasso) and l1 norm (ST lasso).

A.7.5 Parameter-based multitask and transfer learning without task descriptors

Parameter-based transfer learning links different tasks by using a task-related penalty term on the model parameters. In other words, the models of similar tasks are assumed to have similar parameters. Thus, this approach only applies to parametric models.

One of the most widely used approaches in this category is the "low-rank" matrix approach [381], which assumes that some task-specific models have co-linear weights. In the case of p data features, m tasks and n samples, the "low-rank" matrix approach seeks to find the minimum rank solution $W \in \mathbb{R}^{n*m}$ of the linear matrix equations $AW = B$, where $A \in \mathbb{R}^{p*n}$ and $B \in \mathbb{R}^{p*m}$.

This problem is NP-hard in general, and various approaches have been proposed to solve it approximately. One widely used approach entails regularising a standard objective function by the trace norm of the model parameters $||W||_*$, defined as the sum of the singular values of W. In particular, Pong et al. [381] proposed a reformulation that can be efficiently solved.

When dealing with tasks without known descriptors, we can assume tasks are related following a hierarchical tree structure [382, 383] (or any intended structure, like a graph structure [384]). Such approach can be considered both for linear [382] and non-linear [385] models.

Note that parameter-based transfer learning can also be performed by using Bayesian priors on model parameters, either by assuming that all models are sampled from an identical prior [386, 387], or by learning the latent hierarchy within the tasks [388].

A.7.6 Parameter-based multitask and transfer learning with task descriptors

In some cases, tasks descriptors are available, so that they can be used to compare tasks and to monitor information sharing between the tasks. For instance, in the case of drug-target interaction prediction, ligand-based virtual screening considers that each target interaction prediction defines a task. Therefore tasks can be describe by the associated protein sequence descriptors (see section 2.1).

A naive multitask approach with task features consists in clustering tasks based on their descriptors, and to fit cluster-wise models [389, 372].

Kernel methods offer a rich framework for multitask learning with task descriptors.

A preeminent approach is to use the Kronecker product of instance-specific and task-specific kernels [170, 135]. The resulting kernel lives in the space of the Kronecker product of the instance and task specific spaces. Using vanilla kernel methods (like vanilla SVM) with such a (instance-task) pairwise kernel offers an efficient and straightforward multitask approach.

Note this approach falls in the general multitask framework in which multitask convex problems are reformulated as singletask convex problems, as demonstrated in [390]. Indeed, they show that the following multitask optimisation problem (where $J(u)$ is the penalty term for task similarity, and m and n are the numbe of tasks and samples)

$$\frac{1}{mn} \sum_{l \in \mathbf{N}^m} \sum_{j \in \mathbf{N}^n} \mathcal{L}(y_{jl}, f_l(x_j)) + \lambda J(u)$$

is equivalent to the following singletask formulation

$$\frac{1}{n} \sum_{j \in \mathbf{N}^n} \mathcal{L}(y_j, f(x_j)) + \lambda ||f||^2$$

In the case of a linear model, we note $f_l(x_j) = u_l^T x_j$ and $J(u) = u^T E u$, where u_l are the model parameters for a task l and E captures the similarity between the tasks. The reformulation of the multitask problem uses a linear transformation of the data $x^* = B_l.x$, where x^* lives in the (sample, task) feature space. Extension to the non-linear case involves the representer theorem.

Besides, the Kronecker product of the instance and task kernels correspond to the following linear transformation:

$$B_l \in \mathbb{R}^{(L*d_T)*d_I} = \begin{pmatrix} B_{l0} \\ ... \\ B_{lK} \end{pmatrix}$$

where d_I is the dimension of the feature space of instances, d_T is the dimension of the feature space of tasks and $B_{lk} = \Phi_{lk}\mathbb{I}_{d_I}$ where $\Phi_{l,k}$ is the k^{th} feature of task "l".

A.8 Deep learning

Before introducing the deep learning, we should emphasise that it is the result of recent developments within the relatively old *Artificial neuron Networks* (ANN) framework. "Relatively" old because ANN appeared in the late 50s as one of the first AI algorithms ever considered.

The recent re-branding of ANN derives from several developments: hype (the impressive achievements of deep learning technologies, popularised by Google's AlphaGo, for instance, have raised the interest of researchers) and significant improvement in some popular yet challenging fields such as computer vision [51], speech recognition & translation [190] and bioinformatics [191, 192]. Such performance improvements were possible due to progress in two main areas: big data computational management, and computing power (primarily in the form of clusters of CPUs/GPUs (central/graphics processing units). Other crucial developments that enhanced the deep learning paradigm and ANN modelling are the answer to the vanishing gradient problem in backpropagation [324], and regularisation techniques [391, 392, 393]. These aspects are discussed in the following appendix A.8.4.

Note the successes of neuron networks have led to the development of various programming libraries. Leading examples are PyTorch, Caffe, Lasagne but most of all Keras and TensorFlow have become the most popular. These two latter are Python-based and therefore relatively easy to use for bioinformaticians. Moreover, their active community ensures that the latest neuron network algorithmic development and architectures are available for users. Keras is in fact a wrapper for TensorFlow and others, much easier to use than TensorFlow, whereas this latter is more flexible and slightly more up-to-date.

This section introduces the main results on artificial neuron networks relevant for this thesis. We direct interested readers to the following textbooks for an exhaustive presentation [394].

In the next sections, we introduce the deep learning framework and some extremely useful neuron architectures.

A.8.1 Introduction to Artificial neural networks

Artificial neurons

The core component of artificial neuron networks is the artificial neuron. An artificial neuron computes a weighted average of its inputs (mimicking synaptic weights) and applies a linear or nonlinear function called

activation function, analogously to neuronal cells, to this weighted average (see FigureA.7).

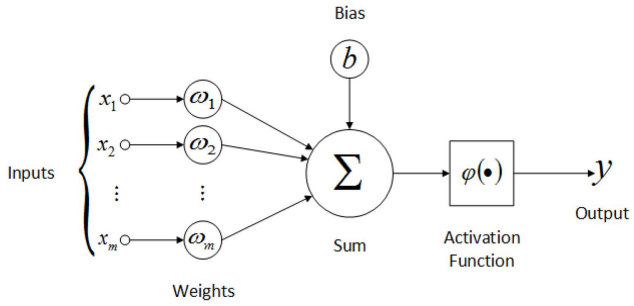


Figure A.7. A computational neuron. The ω_i is the weight associated with each input neuron x_i. ϕ is the activation function. (picture from [395])

Activation functions

The most common activation functions are the sigmoid, hyperbolic tangent (tanh) and Rectified Linear Unit (ReLU) (see Figure A.8). The sigmoid function was originally used as a continuous and differentiable approximation of the Heaviside function. The Heaviside function outputs a signal when its input signal is above a given threshold. A computational neuron with a sigmoid activation function is also called a *gate*, since it outputs a signal if the weighted average of the inputs is positive and is silent otherwise (like the Heaviside function). The hyperbolic tangent is another classical non-linear transformation used in ANN whose output is either a positive or a negative signal, which the main difference with the sigmoid function. However, due to the issue of disappearing gradients occurring when considering the sigmoid and tanh activation functions, the ReLU has become the most widely used activation function. Indeed the gradient of sigmoid and tanh functions tapers off to zero on both extremes of its activation, whereas the gradient of Relu is either 0 or 1, meaning that it never saturates on both extremes.

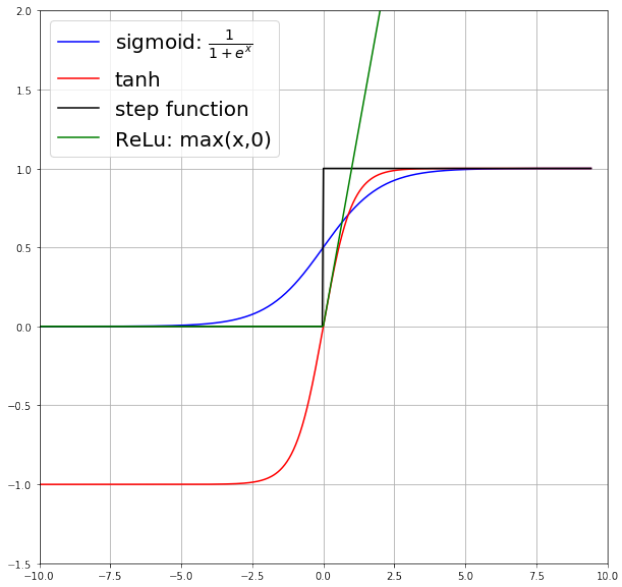


Figure A.8. The three most commonly used activation functions: sigmoid, tanh and ReLU. The Heaviside function, also called the step function, is drawn for comparison.

Multi-layer neuron networks

In ANNs, the computational neurons are connected into complex architectures, and organised into layered networks in which sets of neurons are partitioned into layers. Within a layer, each neuron is connected such that it receives its inputs from some or all neurons of the preceding layer, and gives its output to the next layer (from left to right in Figure A.9). The layered network can be considered "deep" as soon as there are several hidden layers. "Deep" models are to be contrasted with to "shallow" models. Linear models and kernel-based models like SVM (see appendices A.3 and A.5) are considered as shallow models. Indeed, linear models are a generalised form of weighted averages (as performed by a single neuron without any activation function), and kernel models are linear models on a transformation of the input space operated by the chosen kernel (see appendix A.5).

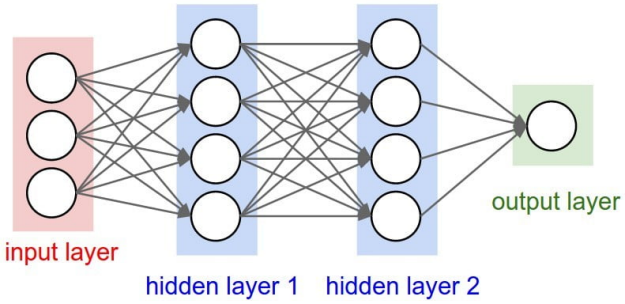


Figure A.9. An artificial neuron network made of an *input layer* with 3 *input units*, 2 *hidden layers* each made of 4 *hidden units* and an *output layer* made of a single *output unit*. (picture from [396])

The data descriptors are directly fed into the *input layer*. Then, the outputs of the input layer pass through intermediate layers call *hidden layers* performing successive non-linear transformations of the input data.

The predictions are generated at the final layer, called the *output layer*. The output layer may contain multiple nodes so that an ANN can natively perform multitask prediction.

When training neuron networks, we minimise a "loss" function describing the discrepancy between the prediction $\hat{y}$ of the neuron network from the true outcome y, as a function of the parameters. More precisely, we minimise the risk, which is the expectation of the loss. Since the true distribution of the data $P(x, y)$ is usually unknown, the weights of a neuron network are adjusted via the minimisation of the empirical risk, an approximation of the risk over the training data: $\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(f(x_i), y_i)$ (N refers to the number of samples in the data).

The most common loss functions are:

- for a regression problem: the mean square error loss function is commonly used to assess the error and optimise the neuron network: $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$. No activation function is considered to get a final unbounded real value $\in \mathbb{R}$ at the final layer.
- for a binary classification problem: the binary cross-entropy loss function is used as the loss function: $H_{binary} = \frac{1}{N}\sum_{i=0}^{N} y_i log(p(y_i)) + (1 - y_i) log(1 - p(y_i))$. Intuitively, $log(p(y_i))$ is added to the loss if sample i is positive, and $[1 - log(p(y_i))]$ if sample i is negative, which penalises wrong prediction. The sigmoid activation function is used for the output units, since it provides a probability value, and thus, a real value $\in [0, 1]$.
- a multi-class classification problem: the cross-entropy is used as the loss function, as a generalisation of binary cross-entropy to M classes: $H = -\sum_{i=1}^{N} y_i log(y_i)$ where y_i is the probability distribution of true labels (typically a one-hot vector), and $p(y_i)$ is the probability distribution of the predicted labels. A softmax function is applied over output neurons so that the final output units are a probability vector (all outputs summing to one).

For a given task and its associated loss function, parameters for each layer (named the "weights" of the network) are optimised by the back-propagation algorithm. More precisely, the derivatives of the empirical risk over all the parameters are calculated by the back-propagation algorithm based on a chain rule, and used in a gradient descent based algorithm [397].

In practice, for each layer l, the gradient of the loss function $\mathcal{L}$ is computed on local weights $W^{(l)}$, $\frac{\partial \mathcal{L}}{\partial W^{(l)}} = \frac{\partial \mathcal{L}}{\partial s^{(l)}}.{X^{(l)}}^T$ (to update them subsequently), and the gradient on the local input $X^{(l)}$, $\frac{\partial \mathcal{L}}{\partial X^{(l)}} = \frac{\partial \mathcal{L}}{\partial s^{(l)}} \cdot {W^{(l)}}^T$ to pass the error back to the previous layer. In the above definition, $\frac{\partial \mathcal{L}}{\partial s^{(l)}} = \frac{\partial \mathcal{L}}{\partial \sigma^{(l)}}.\frac{\partial \sigma^{(l)}}{\partial s^{(l)}}$ is the gradient of the scores $s^{(l)} = W^{(l)}x^{(l)} + b^{(l)}$ without activation. Note that $\frac{\partial \mathcal{L}}{\partial \sigma^{(l)}} = \frac{\partial \mathcal{L}}{\partial x^{(l+1)}}$ since $\sigma^{(l)} = x^{(l+1)}$. Finally, $\frac{\partial \sigma_i^{(l)}}{\partial s_i^{(l)}} = \partial \sigma_i^{(l)}(1 - \sigma_i^{(l)})$ for the sigmoid function and $\frac{\partial \sigma_i^{(l)}}{\partial s_i^{(l)}} = 1_{\sigma_i^{(l)} > 0}$ for ReLU for instance.

Then, based in the input data, deep neuron networks adapt the weights between two neurons by minimising the derivatives of the local loss for all input features, in order to correct for the prediction error made at the output layer. Training a deep neuron network usually requires a large amount of labelled data.

Deep learning pros and cons

At this stage, we can highlight the two main benefits of the deep learning framework:

1. it is particularly suited for multitask learning, since it inherently builds relationships between multiple prediction tasks by sharing hidden units among the tasks. We discuss multitask learning in the context of artificial neuron networks in appendix A.8.3.

2. it builds complex features from raw input data in a hierarchical manner, via successive non-linear transformations. Providing higher-level abstractions over the input samples, it might identify unknown structures in the data. Indeed, deep learning algorithms learn a transformation of the data at multiple levels, creating a new representation of the data better suited to learn a particular prediction model. This is considered to be the main reason why deep learning yielded to performance improvements in many fields.

.

Deep learning suffers in practice from two major drawbacks:

1. the overfitting issue: the computational power of such algorithms is so large that they overfit on the training data easily. Solutions to the overfitting issue are discussed in appendix A.8.4. Indeed, it is well known that neuron networks with at least one hidden layer can implement any Boolean function and approximate, arbitrarily well, any continuous function defined on a compact set, given enough hidden units [398, 399]. It can be easily justified in the case of a binary network in which neurons are equipped with the Heaviside activation function. Indeed we can enforce a neuron in the hidden layer to fire for each input individually, as in a look-up table. This also generalises to networks with sigmoid neurons or other activation functions, but it is trickier to show since we cannot isolate inputs like a look-up table. In the literature, this property is referred to as the universal approximation property. It means that a deep neuron network can fit any training data, but it does not mean that it is to generalise well to make predictions on new data points.

2. the black-box question: the learning process cannot be easily understood. In bioinformatics, the ability to provide predictions that are biologically interpretable is crucial in order to link these predictions with current knowledge, or to suggest new insights into the underlying biological processes. Besides, there is no mathematical theoretical result to prove or falsify the efficiency of deep learning-based algorithms. Moreover, as it is challenging to understand the behaviour of deep neuron networks, it is hard to understand what went wrong when the network fails. This concern is discussed appendix A.8.5.

In the following sections, we introduce some extremely widespread deep learning architectures and discuss how they perform representation learning.

A.8.2 Deep neuron networks useful architectures

Feed-forward Neuron Network

The most common deep neuron network architecture is the *stacked fully connected layers* or *Feed Forward neuron Network* (FNN) or *Multi-Layer Perceptron* (MLP) or simply *Deep neuron Network*. All these terms are found in the literature and refer to the same architecture.

It is the "traditional" deep neuron network represented in Figure A.9. In such a network, each neuron at a layer l takes as input all neuron outputs from layer $l-1$ and thus, directs its output to all neurons at layer $l+1$ (which is what "fully connected" refers to). The original data are subjected to non-linear transformations across several layers such that, when optimised, each intermediate (hidden) representation is meant to capture abstract transformed features of the original data. In this sense, such a network is a form of representation learning model, because the architecture is optimised to transform the original data into another representation that is more efficient for a given task (as discussed previously in Section A.8.1).

FNN is mainly used when data are structured (each input has a feature vector), but are not structurally arranged neither spatially nor chronologically. (e.g. features are not related in time or space). Indeed, since each neuron at layer l takes as input all the neurons in layer $l-1$, each neuron at layer l processes the layer $l-1$ as a whole. In the case where the data features are spatially arranged (like graphs, or images which are 2D grids of pixels etc.), Convolutional neuron Networks (CNN, see Section A.8.2) are a relevant neuron architecture to exploit such data structure. Recurrent neuron Networks (RNN, see Section A.8.2) however, are meant to handle time, and more generally, sequentially structured data.

Moreover, FNN has the significant drawback in requiring a large amount of parameters. Indeed, if the layers $l-1$ and l have both 100 neurons, the connections between the two already require 10,000 parameters. CNN and RNN also have the advantage of reducing the number of free parameters while exploiting the spatial or sequential arranged of the data.

Recurrent neuron Network

Recurrent neuron Network (RNN) is a particular form of neuron network architecture which uses many copies of the same neuron modules (or cell). Recurrent neuron network approaches can also be viewed as message-passing algorithms, as they propagate a signal along a sequence based on the local input and the previous output signal. The RNN model can then exploit the sequential structure of the data.

From a general point of view, an RNN is made of the same neuron module, that is repeatedly applied on the concatenation of the local inputs x_i with the output at the previous step h_{i-1} to generate the local output h_i. More precisely for the classic RNN, h_i is computed as presented in Figure A.10 by $h_i = tanh(W.(x_i, h_{i-1}) + b)$. The same neuron, and therefore the same learnable parameters W and b are used on every input i.

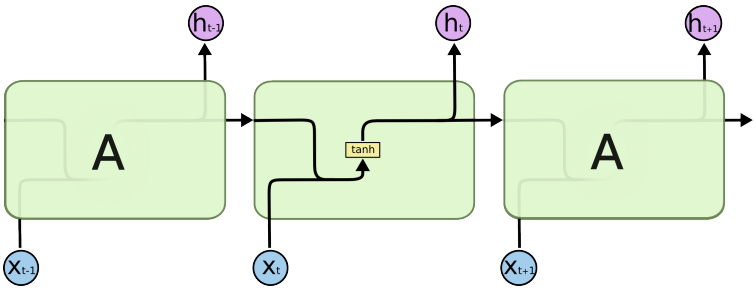


Figure A.10. The repeated module in a standard RNN contains a single layer: $h_i = tanh(W.(x_i, h_{i-1}) + b)$. (picture from [400])

An RNN takes each input one by one, and therefore, it can take a variable length list of inputs. For instance, an RNN can work on sentences in which word i is represented by the input x_i. The RNN can produce an output h_i for each word x_i and a chain output s_i (see Figure A.11).

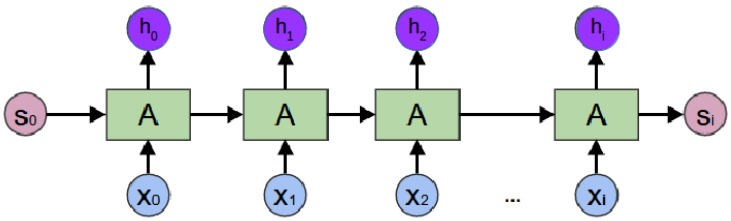


Figure A.11. RNN generating both positional h_i and sequence s_i outputs. (picture adapted from [400])

Note that RNN directionally treats the data so that the h_i value is based on x_i and on the preceding h_{i-1} (from left to right in Fig. A.11). However, in some cases, data are sequentially ordered with no preferential direction, like protein sequences. In such cases, a bidirectional RNN allows processing the input sequence forwards and backwards (see Figure A.12). Then, the predictions h_i are based on what came before and also after the current position i. More precisely, the final output h_i is usually the concatenation of the forward RNN output h_i^f and the backward RNN output h_i^b.

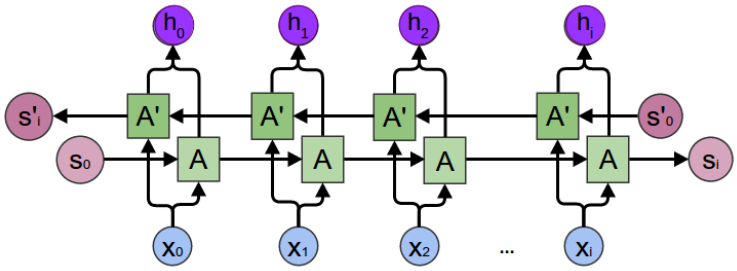


Figure A.12. Bidirectional RNN. (picture adapted from [400])

In the last few years, RNNs have been successfully applied to a variety of problems: speech recognition, image captions and so on [401, 402, 283]. This impressive success is mainly due to a special kind of RNN called Long short-term memory (LSTM) network. LSTM is a kind of recurrent neuron network whose building block is capable of efficiently storing contextual information over different positional length scales. In theory, classic RNN is capable of handling such "long-term dependencies", since it takes as input the previous output h_{i-1} which is supposed to convey information from previous positions. However, classic RNN fails to convey such "long-term dependencies" in practice, due to the vanishing gradient problem. In other words, the standard RNN cell fails to convey long-term dependencies mainly because it is equipped with a single tanh layer, so that an update is "forgotten" after several other updates of the layer. To address this issue, LSTM is conceived to connect previous information to the present task. We often talk about the "RAM" effect of LSTM to refer to that property.

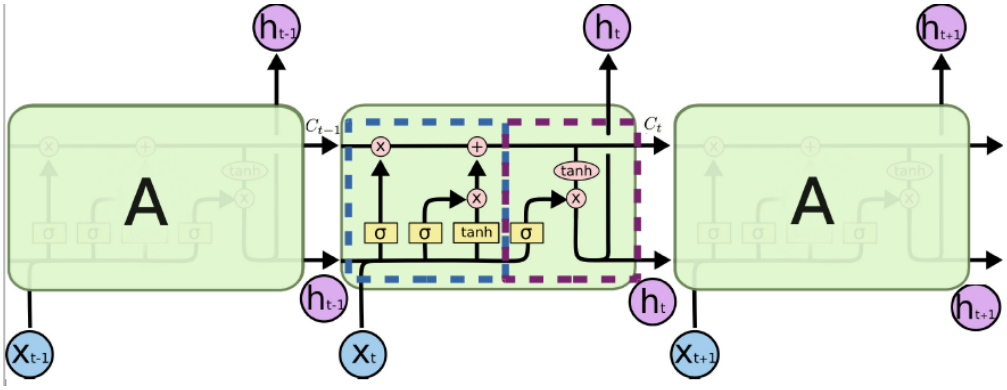


Figure A.13. The repeated module of LSTM. σ and *tanh* refer to the neuron network layers and $+$ and $\times$ refer to point-wise operation. (picture adapted from [400])

The LSTM module can be found in Figure A.13. The core idea of LSTM is to use a cell state vector C_t that runs straight down the entire chain to convey long term information. In the following, we first describe how this cell state C_t is updated (into the light blue dashed square in Fig. A.13), and then how the output h_t is computed (into the light violet dashed square in Fig. A.13).

The cell state is updated at each step based on linear interactions:

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \text{ used to compute } \hat{C}_t = f_t \times C_{t-1} \tag{A.15a}$$

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \text{ and } \tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \text{ used for } C_t = \hat{C}_t + i_t \times \tilde{C}_t \tag{A.15b}$$

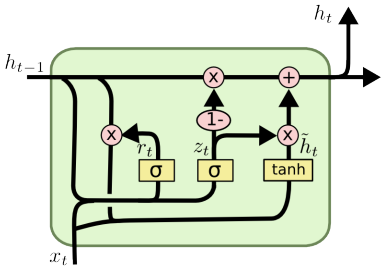
Eqs. A.15a erase some part of the cell state C_{t-1} by applying a gate based on the concatenation of the current input x_t and previous output h_{t-1} (a sigmoid layer can be considered as a gate as it mostly returns point-wise zeros or ones). Eqs. A.15b add filtered (by a gate i_t) modification $\tilde{C}_t$, to the cell state $\hat{C}_{t-1}$. Both the update $\tilde{C}_t$ and the gate i_t are based on the concatenation of the current input x_t and previous output h_{t-1}.

The output h_t is computed based on the current cell state C_t passing through a tanh layer and filtered by a sigmoid gate o_t, taking as input the concatenation of the current input x_t and previous output h_{t-1}.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{A.16a}$$

$$h_t = o_t \times tanh(W_h.[C_t] + b_h) \tag{A.16b}$$

All parameters $((W_f,b_f),(W_i,b_i),(W_C,b_C),(W_0,b_0),(W_h,b_h))$ are adjusted via back-propagation, which is computationally heavy. To address this issue, an increasingly popular variant of LSTM called the Gated Recurrent Unit (GRU) was introduced (see Figure A.14). Most importantly, it combines the forget and input gates (respectively f_t and i_t) into a single update gate. In addition, GRU merges the cell state C_t and hidden state h_t. The resulting model is less complex and in practice easier to train than standard LSTM while preserving the "RAM" property.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure A.14. Gated Recurrent Unit. (picture from [400])

Convolutional Neuron Network

A *Convolutional neuron Network* (CNN) is a widely used neuron network architecture on data with a spatial structure. A CNN can be viewed as a message passing-like algorithm, similar to RNN, since it learns to propagate locally detected information to hierarchically build a global representation of the input.

In signal processing, a convolution over two functions f and g is defined as $(f * g)(x) = \sum_a f(a)g(x-a)$. Therefore, we can think of convolution as sliding one function on top of another, multiplying and adding at each point.

Convolutional networks use convolutional units, which are neuron modules operating on patches of the input, and sliding over the input to detect patterns locally on the data. This is similar to a mathematical convolution.

Convolutions are used in many domains such as audio and image processing. For instance, in image processing, images are two-dimensional arrays of pixels (the features of an image are the pixels values). A convolution on an image refers to the operation of sliding an n*n pixel grid function on the input image and computing a new pixel value for each pixel (see Fig. A.15). The new pixel value results from the weighted value of the local pixel and its neighbours, based on the weight given by the "convolutional filter". In the literature, the function sliding and operating on local patches of the input is called a "kernel" or a "filter" and aims at detecting patterns locally over the entire input. The patch on which the kernel operates is called the "receptive field".

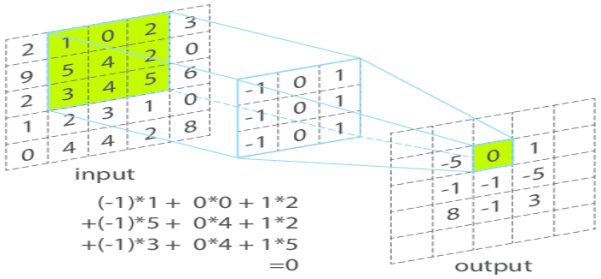


Figure A.15. A kernel computing a new pixel value for an input image based the weighted sum of its local grid pixel values, while sliding over the image. (picture from [403])

For instance, a 3*3 edge detection kernel can be defined by : $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ Indeed, its absolute output value is more significant when the central pixel is opposed to the side pixels. However, it gives approximately zero when the central pixel and side pixels are similar.

In the case of convolutional neuron networks, convolutional units are learnable filter functions ($\sigma(W.x+b)$) operating on local receptive fields which learn the filter weights themselves. In other words, it locally learns feature patterns as it slides over the data. More generally, a convolutional unit usually operates a set of filters in parallel, getting the same input but then trained to detect different patterns. In the case of 2D images, one neuron might detect edges, and another might detect blue-red colour contrasts and so on.

In practice, the convolutional unit is copied and applied simultaneously over the data instead of being slid over the data. Therefore, a convolution layer is filled with multiple copies of a single convolutional unit (see Fig. A.16).

Furthermore, a convolutional network is usually made of several stacked convolutional layers. Thus, more generally, the convolutional unit of layer l operates locally on the output of layer $l-1$, detecting patterns of the patterns detected at layer $l-1$ etc. Such a neuron architecture thus builds more and more abstract

representations of the original data.

In addition to extracting local patterns over the spatial arrangement of the input, convolutional layers reduce the total number of free parameters compared to FNN by using the same convolutional unit, preventing overfitting. Also, highly-efficient parallelised convolution implementations on GPUs brought a widespread usage.

A convolutional layer is usually followed by a "pooling layer" which aggregates neighbour detected patterns (see Fig. A.16). In addition to improving the signal to noise ratio, it contributes to greatly reduce the number of parameters, again preventing overfitting. Moreover, since we are interested into encoding the original image, aggregating neighbour detected patterns enables to forget the localisation of these patterns. The two main aggregation functions are the max and mean functions.

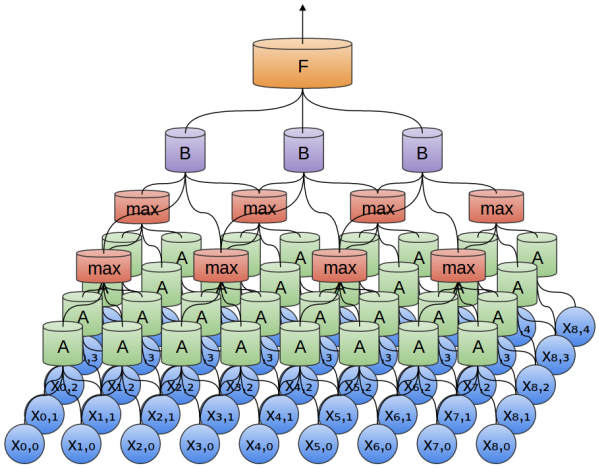


Figure A.16. A 2D convolutional neuron network with a first convolutional layer (whose convolutional unit is A - a bunch of neurons in parallel, getting the same inputs and detecting different patterns-) followed by a max pooling layer and a second convolutional layer (whose neuron is B) to finally build a global representation F of the input. (picture from [400])

To conclude this introduction on CNN, let us discuss some basic yet primordial technical details. We can vary some parameters to modify the behaviour of a convolutional layer. First, we can play with the number of filters, i.e. the number of patterns a single convolutional layer detects simultaneously. We can play on the filter size, i.e. the size of the receptive field on which the filter operates (for images, it is the size of the grid kernel). We can control how the filter convolves around the input volume by defining the stride value, i.e. defining the sliding step size in all directions. Finally, we can define the padding strategy, i.e. the strategy to operate the convolution filters on the edges of the input data. Most frequently, we fill the missing value in the receptive field at the edges with zeros, but others strategies are possible depending on use case.

Auto-Encoder Networks

Auto-encoders are introduced here and used in Section E.

The auto-encoder is a neuron network which aims at encoding the input data into a low-dimensional representation, and then decodes this latent representation to reproduce the input signal as well as possible. Encoding and decoding are performed via neuron layers. A classical auto-encoder network is shown in Figure A.17. However, an encoder/decoder based on convolutional or recurrent or both architectures is also conceivable, as the network performs a contraction of the input data before a reconstruction. Auto-encoders are part of a family of unsupervised methods used for dimensionality reduction, visualisation and also in the semi-supervised setting when labelled data are scarce. Indeed, an auto-encoder learns to encode unlabelled data by building a supervised prediction task which is the reconstruction of the original input.

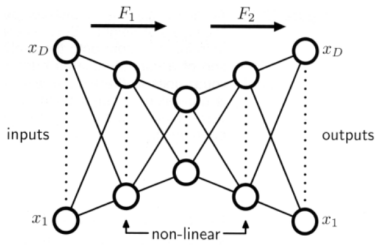


Figure A.17. One the simplest auto-encoder architectures. (picture from [363])

In practice, however, an auto-encoder can quickly learn to compress the data by assigning to each instance a single and isolated latent representation. The latent space may not be continuous, and so does not allow easy interpolation, since the decoder has no idea how to deal with the non-encoded regions of the latent space. This model cannot be used as a generative model, as we may need to encode a continuous representation in order to sample realistic instances from the latent space randomly. To extrapolate new instances, the uncovered parts of the latent space must be filled. A recent yet extremely popular neuron network auto-encoder, called Variational Auto-Encoder [404], achieves this by constraining the encoding network to generate latent vectors following a probability distribution on the latent space (usually a Gaussian distribution). Indeed, as each instance is not only represented by a single vector but by a wide distribution of vectors in the VAE framework, the latent space is more populated. Moreover, as the latent space is more populated while being constrained, it enforces similar instances to share a similar encoding, i.e. to be represented by overlapping distributions.

The VAE corresponds to a "classical" auto-encoder network except that the encoder in the VAE does not directly encode the latent representation of the input, but encodes the parameters of the distribution from which the latent representation is then drawn.

From a probabilistic model point of view, a variational auto-encoder models the observed data x as a latent representation z of instances living in a latent space (see Figure A.18).

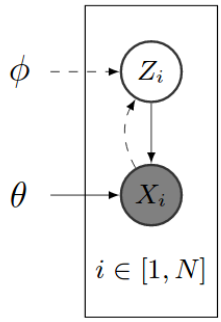


Figure A.18. The graphical model of the variational auto-encoder[404]. We note $p_\theta(z|x)$ and $q_\Phi(x|z)$. x is the observed random variable although it is assumed to be generated from an unobserved latent space z. N is the number of samples.

Following the graphical model in fig. A.18, we can write the joint probability of the model as $p(x, z) = p(x|z)p(z)$. It is a generative model since we can sample an instance $x_i \sim p(x|z_i)$ after sampling from the latent distribution $z_i \sim p(z)$.

Regarding inference in this model, the goal is to infer a proper latent representation z for given observed data x, or in other words, to calculate the posterior $p(z|x) = \frac{p(x|z)p(z)}{p(x)}$. However, $p(x) = \int p(x|z)p(z)dz$ is not computationally tractable as it needs to be evaluated over all configurations of latent variables. Variational inference approximates the posterior $p(z|x)$ with a family of distributions $q_\lambda(z|x)$ parameterised by λ. As discussed in the following, in practice, we want to model $p(z) \sim \mathcal{N}(0, I)$. Therefore, the approximation q of $p(z|x)$ is considered Gaussian, and λ refers to the mean and covariance matrix of the latent variables for each data point $\lambda_{x_i} = (\mu_{x_i}, \sigma^2_{x_i})$. $\lambda_{i=1,..,N}$ is found by minimising the KL divergence between $p(z|x)$ and its approximation $q_{\lambda_i}(z|x)$:

$$\lambda^* = \underset{\lambda}{\operatorname{argmin}}[KL(q_\lambda(z|x)||p(z|x))] = \underset{\lambda}{\operatorname{argmin}}\, \mathbb{E}[log(q_\lambda(z|x))] - \mathbb{E}[log(p(x|z)] + log(p(x)) \tag{A.17}$$

As $log(p(x))$ is still intractable but constant and the KL ≥ 0 (following the Jensen inequality), $\mathbb{E}[log(q_\lambda(z|x))] - \mathbb{E}[log(p(x|z)]$ is a lower bound of $KL(p_\lambda(z|x)||p(z|x))$. Then, minimising $KL(p_\lambda(z|x)||p(z|x))$ can be approximated by the maximisation of the so-called Evidence Lower BOund (ELBO), in eq. A.18, which is tractable.

$$ELBO(\lambda) = \mathbb{E}[log(q_\lambda(z|x))] - \mathbb{E}[log(p(x|z)] \tag{A.18}$$

In practice, the approximate posterior $q_\theta(z|x, \lambda)$ is parameterised with an inference network (or encoder) with parameters θ that takes as input data x_i, and outputs λ_i (parameter of the distribution from which we then draw z_i). The likelihood $p(x|z)$ is also parameterised with a generative network (or decoder) with parameters ϕ that take the latent representation z as inputs, and outputs parameters to the data distribution $p_\phi(x|z)$. The parameters θ and ϕ are typically the weights and biases of neuron networks. The weights are adjusted to maximise the ELBO using stochastic gradient descent. Note that back-propagation cannot flow through a random node, i.e. through the sampling operation $z \sim \mathcal{N}(\mu, \Sigma)$. The so-called "reparameterisation trick" solves this issue. It introduces a new parameter ϵ which re-parameterises z in a way that allows back-propagation: $z = \mu + L\epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, $\Sigma = L^T L$. From this point of view, z is not sampled from

$q_\phi(z|x)$, but z is a function that takes parameter $(\epsilon, (\mu, L))$. (μ, L) come from the encoder and therefore, the encoder's weights can be adjusted by backpropagation via the partial derivatives w.r.t. (μ, L). ϵ is now the "source of randomness", and there is no need for taking its derivatives to train the encoder.

From a neuron network based auto-encoder point of view, we can recover the same loss function easily. The VAE network is trained by minimising the loss function defined as the reconstruction error arising from encoding z by x, and decoding x from z plus a loss that enforces the latent representation to follow a unit Gaussian distribution. More precisely, $\mathcal{L} = \sum_{i=1}^{N} l_i$ where

$$l_i(\lambda, \Phi) = -\mathbb{E}_{z\sim p_\lambda(z|x_i)}[log(q_\Phi(x_i|z_i))] + KL(p_\lambda(z_i|x_i)||p(z)) \tag{A.19}$$

Eq. A.18 and A.19 are equivalent (see [404]). Eq. A.19 provides helpful intuition of the representative power of the VAE. The first term $-\mathbb{E}_{z\sim p_\lambda(z|x_i)}[log(q_\Phi(x_i|z_i))]$ is the reconstruction loss after the encoding and decoding process. With no limits the values vectors μ and σ can take, the encoder can learn representations which are still spread over the latent space. However, we want to push similar samples to be encoded similarly in order to enable smooth interpolation. The second term forces $p_\theta(z|x_i)$ to be similar to unite Gaussian $\mathcal{N}(0, I)$ by minimising the KL divergence between $p_\theta(z|x_i)$ and $p(z) \sim \mathcal{N}(0, I)$. Therefore, this loss encourages the encoder to distribute all representations around the centre of the latent space. Optimising the two together results in an appropriate representation for the decoder, yet densely packed near the origin of the latent space.

Note that a generalisation of the VAE, called β-VAE [405], has been proposed. It introduces a parameter β on top of the second term $KL(p_\lambda(z_i|x_i)||p(z))$. Then $\beta > 1$ puts an extra-weight on the Gaussian regularisation, and enforces the features in the latent representation to be independent (i.e. a disentangled representation). In other words, by decreasing the size of the available latent space, it enforces the latent components to correspond to features that hold different information.

Another popular generative model is the Adversarial Autoencoder (AAE). It is a classical deterministic auto-encoder, but it is regularised to force the distribution of the latent representation $q(z)$ to match a tractable prior $p(z)$. More precisely, AAE is an auto-encoder trained jointly with a separate network (called "discriminator" network) which predicts whether a given sample came from the latent space of the encoder $q(z)$, or from an a priori distribution $p(z)$. In other words, an AAE optimises alternatively the loss of the auto-encoder and the adversarial loss. It optimises both the discriminator weights and the generator weights, to fool the discriminator into mixing randomly sampled $z \sim p(z)$ and latent $z \sim q(z)$ produced by the generator.

Commonly, the discriminator network compares and tries to discriminate the latent vector $z \sim p(z)$ with a vector sampled from $q = \mathcal{N}(0, \mathbb{I})$. Thus, AAE applies the adversarial scheme to the basic idea of VAE.

Generative Adversarial Networks (GANs) [406] are another type of adversarial deterministic auto-encoders. They are also very popular neuron network-based generative models. Similarly to AAE, a GAN has two networks trained in competition: a generator G, generating instances from pure noise pretending they are

genuine, and a discriminator D trained to discriminate authentic samples from forgeries. Both models train in alternation: the generator generates synthetic data from sampling a noise space and competes against the discriminator to mislead it. This scheme can be formalised as a game-theoretic optimisation problem:

$$\min_G \max_D \mathbb{E}_{x\sim p_{data}}[logD(x)] + \mathbb{E}_{z\sim p_z}[log(1 - D(G(z)))].$$

A schematic view of GAN is in Figure A.19.

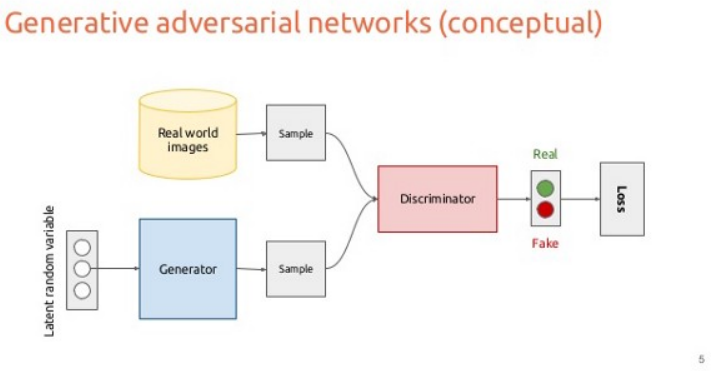


Figure A.19. Schematic view of the Generative Adversarial Network. (picture from [406])

Intuitively, a GAN is expected to sample from the original data distribution if the generator has learned to fool a well-trained discriminator, since any imperfections could have been amplified and used by the well-trained discriminator.

Though a simple model, it is not straightforward to make a GAN converge as it suffers from several issues. The most typical one is the discriminator overwhelming the generator during training. This issue is referred as "mode collapse", and several improvements have been proposed to prevent this undesired behaviour. For instance, Salimans et al. [407] proposed to use mini-batch, and Arjovsky et al. [408] proposed "WGAN" which considers as loss function an approximation of the Earth Moving distance between the empirical distribution and the generator distribution.

Despite similar mathematical structure and similar results in terms of likelihood, current state of the art suggests that GANs are better in generating images, which is currently the most common application of both AAE and VAE [409].

Generative deep neuron networks is an active research topic of deep learning in chemoinformatics, because it is expected to enable generating molecules with some desired physical and biological properties (as discussed in appendix E).

Attention Mechanism

Motivations

Attention mechanisms were found extremely powerful techniques to enable neuron networks to focus on a subset of the information they are given. Indeed, in the case of large and noisy data, only a part of the input data may be relevant for the learning task. In such a case, an attentional interface can highlight elements with the most relevant information, in order to improve the signal to noise ratio. Attention can be used on top of CNN or of RNN layers.

Attention mechanisms were first applied on conversational modelling [410], translation [411] and voice recognition [412]. In the particular case of automatic translation, attention mechanisms are of major importance since the order of words differs for different languages, which means that direct translation of a sequence of words into the corresponding sequence of words is expected to fail.

Note that standard neuron layers can, in theory, learn to focus on a part of its input, by adjusting its weights accordingly. However, in practice, all weights tend to get values in the same range, mostly because the weights are usually regularised by a L^2 penalty (see Section A.8.4). Moreover, a deep neuron network is known to be able to approximate any function [398, 399], but its optimisation is not convex and adjusting the neuron network architecture to smooth the optimisation landscape is meant to facilitate learning.

In addition to performance benefits, an attention mechanism can guide the interpretation of network decision making by analysing the attention weights over the input set.

General definition

Attention mechanism is defined as a function $f : Q \times X^N \rightarrow [0,1]^N$ that maps each of the objects in the input set X^N to a relevance score, or attention weights, based on a query q whose set Q is of the same dimension as X. Usually, these weights α_i result from the softmax function applied on a function g assessing the similarity between the query vector q and the input x_i: $\alpha_i = \frac{exp(g(q,x_i))}{\sum_j exp(g(q,x_j))}$. Typically, g is the dot product $g(q,x_i) = q^T.x_i$, or a more general dot product like $g(q,x_i) = q^T.W_{att}.x_i$, where W_{att} is a learnable weight matrix. Note that the query and other parameters, like W_{att}, are typically learned via stochastic gradient descent together with the weights of the network. In the case of LSTM, in which each input x_i is associated with a hidden state s_i and output state h_i, we can derive similar attention mechanisms like: $g(q,s_i,h_i) = q^T.tanh(W_{att}.[s_i,h_i])$.

Then, the attention mechanism outputs a weighted combination of the input $\sum_i(\alpha_i + 1) \odot x_i$, where $\odot$ refers to the element-wise product. The "+1" operation, often removed, helps the network to learn more robust attention maps by avoiding vanishing gradients caused by consecutive layer-by-layer multiplications [413, 414]. Indeed, a neuron assigned consecutively to a null attention weight may become dead since it cannot adjust to the loss.

Similarly, "self-attention mechanism" refers to an attention mechanism for which the query vector is built based on the input itself, or even on the set of all inputs, usually via a sigmoid layer: $q = \sigma(W_{att}.x + b_{att})$. Intuitively, it relies on the input to compute the query used to focus on some of the input.

attLSTM

In particular, an attention mechanism is an order-invariant operation.

A notable application is a module [306] named attLSTM. We refer to this module in section 6.2. It allows learning a representation for an unordered set of inputs. For instance, we can use this algorithm to learn a graph-level representation based on the unordered set of its nodes' representations.

In essence, it is an iterative way to build set-level representation q^* as a learnable sum over the set of input feature vectors $\{x_i\}$.

At each step t, it uses the output q^*_{t-1} (the temporary set-level representation at step $t-1$) as a state for an LSTM to generate a temporary query q_t so that (no input in the LSTM):

$$q_t = LSTM(q^*_{t-1}, 0).$$

Then, the temporary query is used to build an attention mechanism on the learnable transformation of the original feature vectors

$$z_i t = \sigma(W^{(t)}.x_i + b^{(t)}),\ \forall i.$$

So that:

$$r_t = \sum_i \alpha_{it} z_i \text{ with } \alpha_{it} = \frac{exp(e_{it})}{\sum_j exp(e_{jt})} \text{ where } e_{t,i} = dot(q_t, z_i).$$

The update of the output is the concatenation of the temporary query q_t and attention based weighted sum r_t:

$$q^*_t = concat(q_t, r_t).$$

At the first step of the algorithm, there is no former state, so that q^*_0 is a learnable query vector.

A.8.3 Multitask with deep neuron networks

Singletask and multitask learning have already been discussed in appendix A.7. In a nutshell, singletask models make predictions for one specific question (for instance the interaction of ligands with a single protein), whereas multitask models address multiple questions simultaneously (interaction of ligands with multiple proteins). We focus here on the multitask in the context of neuron networks. Since training deep neuron networks require a relatively large amount of data, multitask learning is expected to have a substantial positive impact on the achievable performances.

Multitask deep learning with ANN: hard-sharing

First, we must highlight that ANNs are inherently multitask models. Indeed, a final output layer with "n" neurons can make predictions for "n" different tasks. In such a case, all layers except the last one are shared

by the "n" tasks, which means that all neurons of the output layer use the same final abstract representation. In addition, all task prediction errors are simultaneously used to adjust the weights of the network encoding the input data into the abstract representation, whereas output units are optimised on individual tasks. This may have a regularisation effect and help the network to generalise better.

A few studies [308, 333, 334], using FNN to predict molecular activities in two hundred biological assays, have shown some efficiency of such multitask learning in the context of DTI prediction. The benefits of multitask learning appear to be stronger for specific tasks. Similarly, other studies noticed the benefits of multitask learning is highly task dependent [308]. Ramsundar et al. [333] noticed that a multitask FNN model exhibits better performances for tasks sharing many active compounds with the others. Xu et al.'s work [334] showed, in the case of QSAR approaches, that the more two tasks can benefit from each other, the more the compounds in the training data of the first task are similar to compounds from the test data of the second task (similar in terms of molecular structures and bio-activities). If bio-activities are uncorrelated, a negative transfer between the tasks has been observed.

Note that forcing two task-specific networks to share the same encoding layers is quite a strong assumptions, and it may be relevant only in the case of extremely related tasks (like predicting interactions for two proteins of the same family).

To consider a less stringent sharing of layers, we can restrict the number of shared layers only to some of the initial layers [415] (see Fig. A.20). In the literature, this MT strategy is referred to as "parameter hard-sharing" since some neuron network layers are directly shared between tasks.

The hard-sharing strategy results in non-smooth regularisation. Moreover, it quickly breaks down if tasks are not closely related, or require sharing information at different levels. A smoother sharing is preferable in most cases.

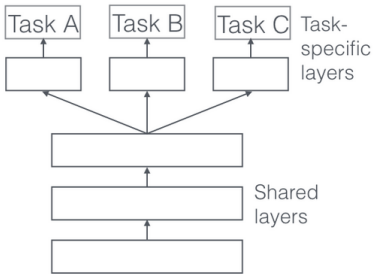


Figure A.20. Sketch of the parameter hard-sharing framework for multitask learning with deep neuron networks. (picture from [353])

Multitask deep learning with ANN: soft-sharing

"Soft-sharing" is a similar approach introducing a softer inter-task knowledge transfer. All tasks are inferred with their individual neuron networks, but task-specific weights are encouraged to be similar layer-wise (see Fig. A.21).

For instance, for each layer l and each pair of tasks (t, t'), we can add to the loss the layer-specific distance between task-specific weights : $||W_t^l - W_{t'}^l||^2$. [416] Note that, it does not force the representations to be similar, but rather encourage the extracted features to be similar. Also, it appears to be more efficient to consider decreasing weights sharing with layer depth l, allowing more and more abstract representations to differ between tasks [353].

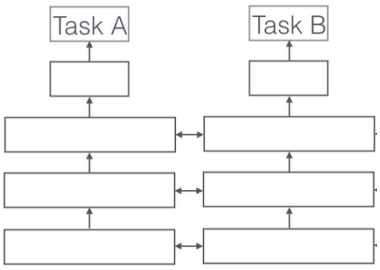


Figure A.21. Sketch of the soft-sharing framework for multitask learning with deep neuron networks. (picture from [353])

Multitask soft-sharing was successfully applied to several object recognition and detection tasks on images by Rozantsev et al. [417]. In this study, authors managed to leverage abundant annotated data from a source task to train a classifier in a sparse target task, by soft-sharing the encoding layers between task-specific networks. They also show that the soft-sharing approach outperformed hard-sharing one.

Multitask deep learning with ANN: partial-sharing

The above approaches force all neurons in each layer, i.e. all representations at every depth, to be at least softly shared between tasks. This strategy can work for strongly related tasks, i.e. tasks which can use the same abstract representation. In the case of loosely related tasks, we can still expect a positive transfer by considering smoother approaches which would let the architecture express both task-shared features and task-specific features. In other words, the network would learn a general representation on different tasks, while providing the ability to learn task tailored features to avoid underfitting.

Some approaches consider neuron modules to transfer layer-wise knowledge between task-specific networks.

Misra et al. [378] defined a cross-stitch network which proposes a layer-wise re-weighting of the output of each task-specific network, to control the flow of information transfer between two tasks. It can be viewed as an extension of soft-sharing to learnable soft-sharing. Indeed, it leverages at each layer the knowledge of the other task by learning a linear combination of the output of the previous layer of both tasks (see Fig. A.22).

Importantly, the information sharing weights are trainable and allow the model to learn what to share.

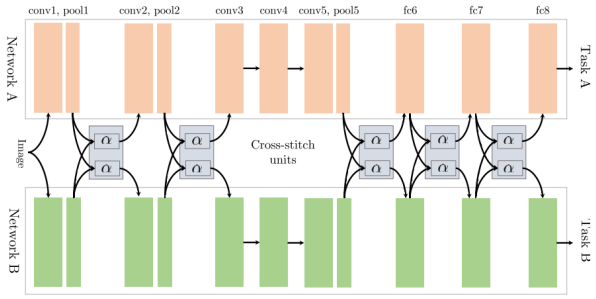


Figure A.22. Cross-stitch network in [378]. Note they only place cross-stitch units after pooling and fully-connected layers.

Ruder et al. [418] extended this work by combining tensor factorisation and the cross-stitch network to learn what layers should be shared in a flexible fashion and how. In other words, they define an architecture that separates at each layer task-specific and shareable representations, in order to learn disentangled representations and improve the learning success and robustness.

Alternatively to transferring information based on direct re-weighting, many alternatives has been published to share partial knowledge between prediction task.

For instance, Xiao et al. [419] defined a neuron network module, inspired by GRU, to share layer-wise information between two tasks.

Also, Liu et al. [420] consider a single shared network which learns a global representation for all tasks. Then, for each task, a task-specific soft attention mask is learnt at each layer of the shared network, to automatically determine the task-specific features among global features. This allows the learning of both task-shared and task-specific features in an end-to-end manner. Furthermore, sharing a pool of global features with automatic task-specific feature combinations allows partial information sharing, with far fewer parameters than the previous multitask architectures which use an explicit separation of tasks.

In this section, we have only considered multitask models via layer-wise knowledge sharing. Meyerson et al. [421] explored the effect of reordering shared layers in a multitask framework.

More precisely, they reported that parallel ordering limits the potential of deep multitask learning, because of the strong constraint it imposes on how each layer is used. Indeed, it may be more difficult for deeper layers to represent features for all tasks, since tasks feature hierarchies may not match, particularly when more and more tasks are added to the model. Moreover, layers used at different depth by different tasks may learn more general features than layers trained at a fixed depth.

Beyond these results and observations, this study suggests to go beyond layer-wise information sharing, by designing neuron modules which may capture recurrent modularities in latent representations at several depths.

A.8.4 Considerations about the training of deep neuron networks

In this section, we introduce practical aspects of deep neuron networks optimisation. Deep neuron networks are known for their high performances, but also for their large number of hyper-parameters and tendency to overfit. In particular, the optimisation of a ANN is non-convex. Therefore, the choice of hyper-parameters has a huge impact on how the parameter space is spanned, and on the optimisation landscape itself.

ANN hyperparameters

We can divide ANN hyper-parameters into three categories: hyper-parameters for the architecture, for the learning process, and for regularisation.

Architecture hyperparameters

Architecture parameters refer mainly to the number of layers and hidden units in the case of FNN and RNN, and the convolution parameters for CNN (see appendix A.8.2). In the case of CNN, we can play with (i) the number of filters (i.e. the number of patterns a single convolutional layer detects simultaneously), (ii) the filter size (i.e. the size of the receptive field on which the filter operates), (iv) the stride value (i.e. defining the sliding step size of the convolutional filters in all directions) and (v) the padding strategy (i.e. the strategy to operate the convolution filters on the edges of the input data).

A bunch of other parameters can also be considered as architecture parameters, such as weight initialisation procedure. Some of these hyper-parameters have default values working for a wide range of applications. In particular, Glorot weight initialisation [393] and gradient optimisation techniques (e.g. Adam [422]) have a significant influence on the overall performance. Note that combining ensembles of 5–10 models initialised with different random seeds usually leads to a substantial increase in performance. Deep learning networks are also robust to architecture-related parameters, which means that a large range of values usually corresponds to the best performance.

Note, however, that the architecture must be deep and wide enough to model the data, while avoiding being too complex to avoid overfitting.

Regularisation hyper-parameters.

In order to reduce the overfitting issue, deep neuron networks consider a collection of methods with their corresponding hyper-parameters. The most important ones being weight decay, dropout [391], batch normalisation [392] and early stopping.

Weight decay refers to the traditional L^2 penalty on the weights.

Dropout is a simple yet highly effective method to control overfitting in the case of artificial neuron networks. Dropout refers to the withdrawal of some neuron units according to a certain probability, at each iteration of the training phase. Therefore, the network learns with missing neurons, which increases its generalisation properties.

Batch normalisation re-parameterises the hidden unit activation, in order to increase convergence speed, but it also renders the output stochastic, creating a regularising effect.

Early stopping refers to the idea of stopping the training when the network starts overfitting, i.e. when the performance on a validation set starts to decrease. In this case, the performance of deep neuron networks must be evaluated with nested-cross validation.

Learning process hyper-parameters

The essential hyper-parameters to tune in order to reach high performances are related to the learning schedule. Learning is the process by which the weights are adjusted. The main algorithm for training a neuron network is stochastic gradient descent with respect to the errors at the output layer. Recall that stochastic gradient descent can be computed efficiently using the back-propagation algorithm, a straightforward application of the chain rule.

Stochastic gradient descent is a gradient-based algorithm updating the weights for each batch of instances. It requires two parameters: learning rate and batch size. The learning rate controls how much the weights are adjusted with respect the loss gradient, while the batch size defines the number of samples that is propagated through the network at each optimisation iteration. Both parameters are coupled. A bigger batch size gives a more averaged direction for the update step, and a higher learning rate increases the size of the update step in the optimisation landscape.

The learning rate is a sensitive parameter that needs to be tuned carefully. Indeed, a learning rate that is too high is likely to trigger gradient explosion, or at least lead to an unstable model, but a learning rate that is too small may not consistently update the weights. In practice, we often exponentially decrease the learning rate during the training over the epochs so that the learning rate at epoch t is: $lr_t = lr_0 * e^(- \beta.t)$ (where the decay factor β is a hyper-parameter to be tuned). Similarly, we can reduce the learning rate when the training error reaches a plateau for few consecutive epochs. Reducing the learning rate during training enables to start training by spanning the parameter space with a large optimisation walk to find a "global optimal subspace" at first, and to keep on optimising the weights with a smaller optimisation walk in a second phase in order to find a local optimum inside the global optimal subspace.

ANN hyper-parameters selection

Deep neuron network optimisation is a strongly non-convex optimisation problem. Therefore, tuning the hyper-parameters of a deep net is not trivial.

In most cases, the ANN hyper-parameter space is explored manually based on the user's experience. Random search, Bayesian optimisation or reinforcement learning are considered to explore the hyper-parameter space automatically.

Some [423] investigated how to adapt learning hyper-parameters (optimiser, batch size, learning rate, architecture parameters) for producing loss functions that are easier to train (less hilly), in order to produce solutions with better generalisation properties.

Random search

In practice, hyper-parameters are usually semi-automatically optimised via random search. Indeed, random search is preferable to grid search as it evaluates each parameter more diversely for the same amount of computation. We then better assess the effect of any latent parameter on the optimisation landscape. A toy example is shown in Figure A.23.

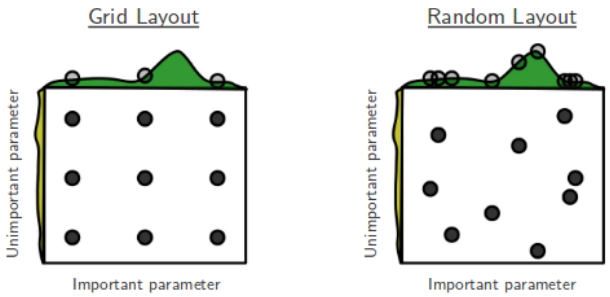


Figure A.23. Random Search better explores the effect of important parameters on the landscape than grid search. (picture from [424])

Bayesian optimisation

However, more sophisticated techniques than random search can be proposed to tune the hyper-parameters of a deep neuron network. Bayesian optimisation [191, 425, 426, 427] is widely used for deep network hyper-parameter optimisation. Note that even more sophisticated approaches combining reinforcement learning with deep learning was used to tune a neuron network [428].

Bayesian Optimisation is a very general framework for finding the global optimum of a non-convex, expensive-to-evaluate objective. It is a gradient-free strategy for the global optimisation of any black-box function. More precisely, given a function f, Bayesian Optimisation constructs a probabilistic surrogate based on previously evaluations, to propose new inputs on which to estimate f based on a predefined acquisition function and the probabilistic surrogate, in the intention of finding its optimum.

Different models can be used as surrogates to model the objective function landscape. The most common are Gaussian Processes, Random Forest and Bayesian neuron networks (standard neuron network except that the weights are now probability distributions from which values are sampled when requested). Gaussian Processes provide a flexible way of finding analytic approximations, but are costly for high dimensional problems. RFs are more scalable but display better prediction performance for categorical input data and classification tasks. Recently, BNNs have gained attention since they retain the flexibility of GPs at a computational cost comparable to RFs.

Likewise, a collection of acquisition functions were studied. One of the earliest and most widely applied is the so-called "expected improvement" function and variants, which quantify the improvement expected by the evaluation of a new point in the parameter space. Each acquisition function proposes hyper-parameters to control the exploration-exploitation trade-off. Indeed, the acquisition function should favour exploration of the entire parameter space when the general location of the optimum has not been determined, and favour

exploitation otherwise.

A.8.5 Interpretability and understanding of deep neuron networks

Deep learning methods are often criticised for being impermeable black-box algorithms, which means that it is difficult to track how they solve a task, compared to other ML algorithm. This is a multifaceted question that requires a careful consideration [429, 430].

Is model interpretability an issue?

First, we expect machine learning methods to learn rules that are, plausibly, too difficult for us because we do not know their underlying rules. This is the case in the fields of material design, bioinformatics and more specifically chemoinformatics. A machine-learning model that could "implicitly discover" such unknown laws are to be always a computational black box.

Secondly, even in the case of known rules, ML models do not, in general, represent knowledge which directly maps to forms that scientists are familiar with. For instance, an artificial neuron network model could learn a "simple" law like the ideal gas law $pV = nRT$, given a good architecture and enough data. But the translation of connection weights into the formula is strongly non-trivial. This introduces the third point.

Indeed, in the case of neuron networks, a certain degree of opacity is inevitable. As suggested by the intuition of the universal approximation theorem (see appendix A.8), training can be viewed as a way of storing the training data in the synapses of the network. It is not an address-based storage like in a hard-disk, but more a spread form of storage since each training example makes a small contribution to each synaptic weight. Therefore, it is fundamentally different from both the equation-based form of storage and digital storage.

Fourth, understanding and controlling is a fundamental desire of human nature. However, we must admit that most of the objects we use and trust every day are black boxes. Most of us do not know how our cars, computers and so on, work in detail. We rely on the fact that there are some "scientists" somewhere who "know" how these objects function, how to repair them ... In the case of health-care, we trust our doctors although we know they have a fragmented, and thus a reduced, understanding of diseases and cures. In any case, every one of us has no clue of how his brain works, but we still trust it blindly (when we do not think about it).

Finally, and more importantly in the case of drug discovery, we must point out that the FDA administration already authorised some therapies based on machine learning. Indeed, the regulatory agencies are prone to authorise black box machine-learning algorithms, as soon as they were proved to be as (or even more) reliable as a panel of the best human experts. In other words, one must prove that when the algorithm takes the wrong decision, a human expert would have also been wrong.

That being said, it is of significant interest to look for tools and tricks to get some insight on the rules learnt by modern machine learning techniques, in order to expand our understanding, especially in the field of chemistry and biology. It could also help practitioners to debug or even design models more effectively, as well as improve trust and transparency, to unleash the adoption of machine-learning methods in decision-making domains.

To this end, we are to (not exhaustively) introduce tools and tricks studied to understand the functioning of deep learning algorithms [431].

Overcoming black-boxness in deep learning

First, it is possible to open the black box of an artificial neuron network, by studying the behaviour of each neuron under all kinds of inputs.

In chemoinformatics for instance, Preuer et al. [432] related individual neuron activations with presence or absence of well-known toxicophores (chemical groups found in toxic molecules) calculated for the same molecules by looking at the correlation (via a Mann–Whitney U-Test and Bonferoni correction) between these toxicophores and the activation of individual neurons after their training.

This kind of approach is prohibitively time consuming, and can only be performed in favourable circumstances, typically in the case of relatively small networks and when expert knowledge can recover the learnt features.

To gain insight on which original features drive the final prediction, a simple yet usually efficient technique is "masking". By comparing the original performance to the one obtained when masking (i.e. set to zero) parts of the original features, we gain a quantitative insight on the masked features relevance.

In the case of neuron networks operating on graphs, Ying et al. [433] proposed a method to identify important sub-graph structures and a small subset of features that play a crucial role in the prediction. Similar to the idea of the "masking" strategy, they considered an optimisation task that maximises the mutual information between the prediction of the full model and the prediction of the same model operating on a simplified sub-graph with removed edge and node features. In other words, the objective is similar to denoising the computation graph to keep edges and features that have the highest mutual information with the prediction.

Furthermore, understanding the internal operations of deep neuron networks fundamentally ties to understanding the data it operates on. Indeed, in the case of classification for instance, the classes are quite tangled at the input layer. However, the next hidden layers have learned to transform the data into new representations in which the classes are more and more separated, because the model has been trained to distinguish the classes.

Thus, understanding the internal functioning of a deep neuron network can be performed relatively well by visualising the internal representations using dimensionality reduction techniques (see appendix A.6 for a brief introduction to dimensionality reduction). Visualising high-dimensional representations using dimensionality reduction is a broadly applicable technique for inspecting models in deep learning.

In the field of chemoinformatics, Hirohara et al. [352] reported that the representation learnt via representation learning on SMILES in a end-to-end manner provided a richer chemical space than standard ECFPs. Indeed, they projected the same set of compounds represented by either the "learnt fingerprints" or standard ECFPs on a 2D space with multidimensional scaling. On this projection, they observed that the former clearly discriminated active and inactive compounds in 2D representation, whereas ECFP failed to discriminate the two groups in this 2D chemical space.

Finally, another interesting understanding of the deep neuron networks is the reliability of individual predictions.

One way to address this issue is using "snapshot ensemble" [434]. This approach derives a set of predictors (i.e. snapshots) with comparable predictive power on average, that however generate slightly different predictions for a given instance. Then, the variability across the predictions are harnessed to compute confidence intervals and assess the reliability of the predictions.

Today, interpretation and fundamental understanding of deep neuron networks has become a field in itself, and many sophisticated methods have been proposed to tackle this question [435, 436, 437, 438, 439, 440, 441].

Appendix B

Multi-kernel learning for drug virtual screening

Mixing heterogeneous data is a promising approach to enhance performances. Indeed, different data sources are likely to contain different, and thus partly independent but complementary, information about the task at hand. Moreover, data sources are expected to be all noisy (missing data, experimental noise etc.), but with different noise patterns. Therefore, combining heterogeneous data may provide regularisation and implicit information augmentation.

Indeed, a variety of biological and molecular data sources are available. In particular, chemical structure, bioassay signatures, drug-induced phenotypes and pharmacophore patterns are widely available for drug-like compounds. Protein geometrical and topological structures, associated GO terms, pathway profile, gene expression, gene methylation profiles and so on, are available for protein targets.

Kernel methods give an efficient framework to integrate various data sources and perform predictions simultaneously with multiple kernels. Indeed, by defining a kernel for each data type, combining such data type-specific kernels allows to integrate various data types.

For instance, we can expect a kernel based on "drug-induced phenotypes" to complement the missing bio-activity knowledge in DTI database. Besides, it could be relevant to consider molecular kernels building a similarity value by comparing molecular graph linear fragment of size 5 and 8, as different length-scale of sub-graph may be involved in the interactions with different proteins.

Most of the published studies on DTI prediction have never been beyond the simple static sum of kernels [115, 114, 130, 132, 109, 107, 112, 442, 113, 134]. These works have reported that a kernel method working on a static linear combination of different kernels outperforms performances obtained with individual kernels.

Learning the weights of such a linear combination of kernels is expected to enhance the information brought by data type combination. Therefore, more sophisticated methods, referred to as Multi-Kernel Learning (MKL) methods, have been successfully applied to various bioinformatics tasks [362, 54]. In the

general case, these methods usually formulate a single optimisation procedure that simultaneously finds the SVM classification solution as well as the weights on the individual data types [443]. Interested readers can refer to the following articles [443, 444, 445, 446].

However, such MKL methodology did not show evidence of its efficiency in many contexts. In particular, Noble et al. [362] have shown, on protein Gene Ontology (GO) term prediction from protein sequences, that a simple unweighted sum of kernels can provide remarkably robust classification performance. A weighted kernel combination method was found relevant only in the case of a large collection of kernels in which some were "clearly" less relevant than the others for the task at hand. In particular, the optimisation of the kernel coefficients is more efficient when there is a noisy kernel. They also report that the weights associated with each kernel cannot be used to interpret the importance of each data source, since a wide range of kernel weights can result in the best performances.

In the case of drug target interaction prediction, different ways of performing multi-kernel learning have been investigated. We review here the two most popular, "similarity network fusion" and "kernel alignment".

Similarity network fusion

Similarity Network Fusion (SNF) [447] non-linearly combines similarity measures into a single fused similarity, while selecting the most relevant non-redundant similarities. More precisely, it iteratively updates each similarity matrices with information from the others, using K-nearest neighbours, such that they are more similar to the others. The SNF method can capture common and complementary information across different similarity measures. In particular, SNF fuses kernels such that weak similarities disappear in the face of strong similarities, helping to reduce the noise. SNF has been successfully applied in the context of virtual screening [164, 165].

In particular, Olayan et al. [165] recently proposed an approach using similarity network fusion to build a DTI heterogeneous graph, which seemed to significantly outperform previous methods. Based on the DTI heterogeneous graph, they built a set of (only) 12 features to predict the interaction score of drug-target pairs with a random forest classifier.

The DTI heterogeneous graph is a weighted graph made of two types of nodes representing molecules and proteins. The edge between two drug nodes or two protein nodes represents their similarity. A drug and a protein are linked by an edge (weighted by one) if they are known to interact. Over this DTI graph, they defines paths from a drug D to a protein target T as walks with more than one edge and without loops. They limited the length of paths to three, such that there are six possible paths from D to T: (D-D-T), (D-T-T), (D-D-D-T), (D-D-T-T), (D-T–D-T) and (D-T-T-T). For computational reasons, they also restricted all paths between D and T to pass only through the 5-nearest neighbours of D and only through the 5-nearest neighbours of T.

Then, they defined six features based on the six types of paths linking D to T, and six features based on the six types of paths linking T to D (therefore, a total of 12 features for each each D-T pair). For each

path p of a specific type of path, they associated a value obtained by multiplying all weights of the edges of the path. Then, the single feature corresponding to a specific type of path is the sum (or max) of the set of values calculated as before.

In a nutshell, this methodology only relies on the product of similarity measures to describe molecules and proteins with little algorithmic refinement, similarly to early approaches. Thus, its predictive power is mostly due to the consideration of multiple kernels.

Kernel alignment

Another widely considered MKL method for DTI prediction [166, 167] is a linear combination of kernels whose weights are computed via "kernel alignment" with a kernel comparing instances via their output values. Kernel alignment between two kernels returns of a real value assessing how much the two kernels are similar (i.e. assessing if instances similar for one kernel are still similar for the other one). Thus, the idea is to learn a linear combination of kernels which is maximally aligned to the response kernel, which can be viewed as the "ideal ground truth" kernel, and then use the learned mixture kernel as the input kernel for training a prediction model.

In the context of DTI prediction, two significant studies discussed the use of kernel alignment [166, 167].

Nascimento et al. [166] defined an MKL named KronRLS-MKL based on kernel alignment. They computed two sets of kernel weights, separately for molecule and protein kernels. Also, they considered various protein kernels (based on PPI, sequence, interaction profile, GO) and molecule kernels (structural and interaction profile).

They fairly compared performances in different settings considering a widely used benchmark dataset (Yamanishi dataset) with others MKL approaches: an unweighted sum of kernels, another MKL method based on kernel alignment, and an MKL approach relying on Bayesian matrix factorisation taking as input a linear combination of kernels [448]. Their method exhibited state-of-the-art performances. However, the optimised kernel weights within the kernel combination approximately correspond to the mean of the kernels, showing that such a sophisticated method is almost equivalent the naive approach of kernel summation.

Later, Cichonska et al. [167] extended MKL based on kernel alignment to pairwise kernels. In particular, they derived a formulation of kernel alignment for kernels resulting from the Kronecker product of two kernels, without requiring the explicit computation of multiple huge pairwise kernels resulting from the Kronecker product. Additionally, they proposed a formulation of a regularised least-squares model, with multiple pairwise kernels resulting from the weighted sum of Kronecker products, again, without explicit construction of any massive pairwise matrices. They report slightly better performances than KronRLS-MKL. Thus, they provide a state-of-the-art MKL approach that is computationally efficient, regarding processing and memory requirements, and which fully exploits the information contained in the space of pairs.

Meta-predictors

Another approach to combine heterogeneous data type comes from the meta-predictor framework introduced in section 3. The idea is to train different classifiers on different feature spaces, and combine the predictions. However, instead of averaging and considering the maximum of the prediction scores, a classifier trained on the output scores learns the best association between the prediction made on different feature spaces. Such an approach has been successfully applied to chemical properties prediction [449, 450].

Partial conclusions

Finally, we think that they are two main issues for applying MKL to DTI predictions.

First, we can imagine that some protein-compound interactions are more easily recovered thanks to the structure-based similarity, and that others are better retrieved thanks to pharmacological-based similarities. In a kernel setting, this means that some samples may be best classified with one kernel and others using another kernel. To tackle this issue, Noble et al. [451] built a method based on latent variable modelling, in which the learned kernel weights in the linear combination of kernels depends on the samples.

The second main issue results from the sparsity of some of the available data types. For instance, side effects data is available only for FDA approved drugs, and drug-induced phenotypes are only available for a subset of them. Facing the same issue, Noble et al. [362] considered three alternatives to tackle missing data in some feature spaces. The first strategy replaces the row and column of kernels corresponding to missing entries with all zeros, which is meant to ignore the kernel with missing data for the corresponding samples. The second makes each missing example similar only to itself by the same kind of process. The third one makes each missing example similar to every other missing examples, but different from all non-missing examples. All missing examples are then placed in a single orthogonal dimension subspace infinitely distant from the other data. However, no strategy can rescue a significant amount of missing data.

Appendix C

An historical perspective on graph representation learning

Representation learning on molecular graph is an extremely active field whose applications are very diverse: social network analysis, gene-gene or protein-protein or drug-drug interaction graph analysis and, more generally, analysis of any type of data with a graph structure.

In the following, we propose a historical perspective on graph representation learning.

C.1 Graph shallow embeddings

As presented in Hamiton et al.'s review [292], shallow embedding methods were first proposed to perform graph representation learning. More precisely, matrix factorisation [452, 453, 454, 455] and random walk [314, 315] based methods were considered.

In the case of shallow methods, an embedding matrix $Z \in \mathbb{R}^{d \times |V|}$ containing the d-dimensional representation vectors for all nodes is optimised directly. More precisely, the optimisation loss is given by:

$$\mathcal{L} = \sum_{(v_i, v_j) \in V^2} l(DEC(z_i, z_j), s_G(v_i, v_j)) \tag{C.1}$$

where z_i is the embedding vector of node v_i, l is a distance measure, DEC and s_G are similarity measures on node embeddings and node topology respectively.

Typically, in the case of matrix factorisation, $s_G(v_i, v_j) = A_{i,j}, \ldots, A^k_{i,j}$ (A being the adjacency matrix of the graph) and $DEC(z_i, z_j) = ||z_i - z_j||^2$ or $DEC(z_i, z_j) = z_j^T z_i$. Thus in the case of matrix factorisation, the "topological similarity measure" s_G between two nodes reflects their connectivity given by the power of the adjacency matrix A^k, $k \in \mathbf{N}^*$ [452] (others also consider more general topological similarity measures [453]).

Alternatively, the "topological similarity measure" $s_G(v_i, v_j)$ can be encoded via the probability of reaching node v_j by a random walk of fixed length starting from v_i. Thus, nodes have similar embeddings if they tend to co-occur on short random walks which is a more flexible measure of node topological similarity than

the adjacency matrix. With this topological similarity measure, the embeddings z is intended to encode the statistics of random walks. In particular, we can define

$$DEC(v_i, v_j) = \frac{e^{z_i^T z_j}}{\sum_{v_k \in V} e^{z_i^T z_k}} \approx p_{G,T}(v_j|v_i)$$

where $p_{G,T}(v_j|v_i)$ is the probability of visiting v_j of a length-T random walk starting at v_i in graph G.

These shallow embedding approaches have the significant drawback of training different embedding vectors for each node independently. Indeed, no parameters are shared between nodes in the encoder. Moreover, these are not transductive approaches, i.e. they can only generate encodings for nodes that are present during the training phase.

To address these issues, a neuron network auto-encoder was proposed [291] as shown in Fig C.1. In this approach, each node v_i is associated with a neighbourhood vector $s_i \in \mathbb{R}^{|V|}$, which corresponds to i^{th} row in the matrix $S_{i,j} = s_G(v_i, v_j)$ (s_G typically defined as before). Such a general auto-encoder is trained by minimising the reconstruction loss: $\mathcal{L} = \sum_{v_i \in V} ||DEC(z_i) - s_i||_2^2$.

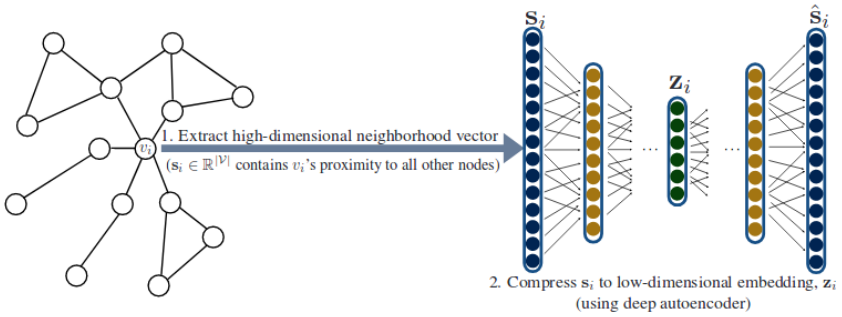


Figure C.1. $s_i \in \mathbb{R}^{|V|}$ summarises v_i's similarities with all other nodes in the graph. The s_i vector is then fed through a deep auto-encoder to reduce its dimensionality, producing the low-dimensional z_i embedding. (picture from [292])

This approach has the benefit of sharing encoding parameters between nodes. However, as the size of the auto-encoder is fixed for training, it cannot cope with evolving graphs, nor can generalise across graphs with a different number of nodes.

Addressing this issue resulted in the design of encoders that rely on a node's local neighbourhood, and so not on the entire graph. Such encoders have been split artificially into two frameworks, the "message passing" and "neighbourhood aggregation" frameworks, although both are strongly related. The "neighbourhood aggregation" framework can be further sub-divided into two approaches, the "spectral" and "spatial" approaches. The intuition behind these frameworks is to generate an embedding for a node by aggregating information from its local neighbourhood.

In the next section, we first introduce the "message passing" framework [317, 316, 312, 303] (also referred to as Graph neuron Network or GNN).

Since the spatial approach of the "neighbourhood aggregation" framework is the most widespread framework nowadays, it is introduced in the main text in Section 6.2.

C.2 Graph neuron networks

The seminal paper considering message passing for graph representation learning based on neuron network is the work of Gori et al. [317]. They presented the first Graph neuron Network (GNN) as a extension of recurrent neuron networks for graph-structured data.

Indeed, GNNs consist of an iterative process, which propagates the node states to each neighbouring node until equilibrium. Thus node representations are updated depending on the graph topology of the data. More precisely, in this seminal paper [317], at each step t, they propose the following update rule for each node i ($\mathbf{h}_i^{(t)}$ is the latent representation of node i at step t and $\mathbf{x}_i$ is its original description, $\mathcal{N}(i)$ is the 1-hop neighbourhood of node i):

$$\mathbf{h}_i^{(t+1)} = \sum_{j \in \mathcal{N}(i)} H_W(\mathbf{h}_j^{(t)}, \mathbf{x}_i, \mathbf{x}_j) \tag{C.2}$$

The function H_W is an arbitrary function modelled by a neuron network with weights W. Eq. C.2 is repeatedly applied until the embeddings converge, which is ensured when H_W is a contraction map [317]. The final encoding of each node is computed following $\mathbf{z}_i = g(\mathbf{h}_i^{(T)})$, where T is the final message passing step and g is an arbitrary differentiable function modelled by a neuron network.

This seminal work on GNN was extended by Sarselli et al. [316]. They proposed the same kind of update rules with the same optimisation strategy but discuss various parameterisations of H_W and g based on multi-layer perceptrons (MLPs).

Later on, this work was extended by Li et al. [312] who proposed, in the case of directed graphs, the use of a Gated Recurrent Unit -GRU- (see appendix A.8.2) per edge type to pass information from each node to its children nodes. In addition to performance improvement, the use of GRU removes the need to run Eq. C.2 until convergence which was the main drawback of this approach. In particular, the message passing rule of Li et al.'s work [312] is defined in the following:

$$\mathbf{h}_i^{(t)} = GRU(\mathbf{h}_i^{(t-1)}, \sum_{j \in \mathcal{N}(i)} \mathbf{W}.\mathbf{h}_j^{(t-1)})$$

Finally, Gilmer et al. [303], discuss a more abstract message passing rule, considering models of the form in eq. C.3, in which q is a differentiable function that computes the incoming "messages" from neighbours and U is a differentiable "update" function. We see that we recover the general framework introduced in section 6.2.

$$\mathbf{h}_i^{(t)} = U(\mathbf{h}_i^{(t-1)}, \sum_{j \in \mathcal{N}(i)} q(\mathbf{h}_i^{(t-1)}, \mathbf{h}_j^{(t-1)})) \tag{C.3}$$

In the past few years, we have seen an explosion of papers considering graph neuron networks for many applications or proposing an analysis and further improvements. Note however that most of them refer to

Graph Convolutional Neuron Networks (GCN) rather than Graph Neuron Networks (GNN). Indeed, the GNN framework, presented before [317, 316, 312, 303], considers graphs as specific scaffolds for a "message passing" algorithm (based on an RNN-like module).

Although separated in the literature, GCN are conceptually strongly related to GNN as they rely on node representation updates based on neighbourhood representation. Indeed, in the case of spatial graph convolution networks, we can generalise the concept of stride and receptive field for images as the following: (i) the stride of the convolution is the shortest distance between the nodes whose attributes are updated via neighbour aggregation (it is usually set to 1 in the case of small graphs like molecules); (ii) the receptive field of a node i is the set of nodes $\{i\} \cup \{j\}_{j \in \mathcal{N}(i)}$ where the neighbourhood $\mathcal{N}(i)$ of i can be defined as the n-hop neighbourhood of i (e.g. all nodes whose maximum distance from i is n) or by a random walk of fixed length starting from i. Thus message passing-based GNN are GCN for which the stride is set to 1 and the size of the receptive field is set as the 1-hop neighbourhood of nodes.

In the following appendix section, we conclude this historical perspective by discussing the specific works on molecular graph embedding.

C.3 Pioneering work on molecular graph representation learning

In parallel with early developments of graph convolutional networks (GCN), few seminal works focused specifically on the representation of molecular graphs, although we noticed it was often a pretext to work with small graphs.

Most of these approaches were tested on benchmark dataset obtained from the MoleculeNet collection (introduced in section 2.3).

None of the following approaches stands out from the others and from more standard fingerprint-based approaches (multi-task DNN, random forest, SVM and logistic regression methods), with regards to predictive performance. In other words, **t**he state-of-the-art performances reached by the following neuron networks-based approaches do not yet exhibit competitive advantages with respect to fingerprint-based approaches. Further discussions on the efficiency of these approaches for biological activity prediction can be found in Section 7.

However, in a chronological perspective, we aim now at introducing and discussing conceptually the seminal works on molecular graph representation learning.

C.3.1 Graph neuron network and graph convolutional network seminal studies

Duvenaud et al. work [287]

The pioneering work on neuron networks for molecular graphs is the neuron fingerprint algorithm by Duvenaud et al. [287] published at NIPS 2015. It is the first recent paper to our knowledge that introduced data-driven feature encoding for molecules.

They performed convolutional operations on graphs to build differentiable graph features (more precisely fingerprint-like features) called "neuron fingerprints". More interestingly, they exhibited the relationship between their differentiable neuron fingerprint and classic ECFPs [57]. The algorithm for building the so-called neural fingerprint can be found in alg. 3. Also, conceptual similarities with the automatic data-blinded extraction of ECFPs are discussed in alg. 3. In essence, this algorithm smoothly and differentially extracts fingerprints directly based on the molecular graphs and their labels, justifying the data-driven nomenclature and hoping for more specificity and flexibility.

Conceptually, alg. 3 is a particular case of GNN whose node updating rule is simply summing the neighbour embeddings. Note also the graph-level embedding is not obtained though consecutive pooling operations like in the case of convolutions of images but is computed after every node update by summation of the nodes' representation.

Neural fingerprints have three benefits compared to the data-blinded ECFPs (state-of-the-art fingerprints): interpretability, parsimony and predictive performance on some chemical property prediction tasks (few points however). Nevertheless, they noticed that the learnt fingerprints are strongly correlated to ECFPs while being more computationally costly.

Algorithm 3 Neural fingerprint algorithm in [287]

Require: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\mathbf{x}_v, \forall v \in \mathcal{V}$; depth L; learnable weight matrices $(\mathbf{W}_0^{(l)}, \mathbf{W}_1^{(l)})\ \forall l \in 1, \dots, L$

initialisation of graph-level representation: $\mathbf{m} = 0$.

initialisation of node-level representation: $\mathbf{h}_v^{(0)} = \mathbf{x}_v\ \forall v \in \mathcal{V}$

for layer: $l = 1 \dots L$ **do**

  for atom: $i \in \mathcal{V}$ **do**

    $\mathbf{v} \leftarrow \mathbf{h}_i^{(l-1)} + \sum_{j \in \mathcal{N}(i)} \mathbf{h}_i^{(l-1)}$ (sum of atom and neighbours feature vectors instead of concatenation in ECFP)

    $\mathbf{h}_i^{(l)} \leftarrow \sigma(\mathbf{v}^T.\mathbf{W}_0^{(l)})$ (instead of hard encoding of graph substructure by a hash function as in ECFP, the encoding based on the activation via the sigmoid function is smooth and differentiable)

    $r \leftarrow softmax({\mathbf{h}_i^{(l)}}^T.\mathbf{W}_1^{(l)})$ (update of the molecule-level encoding)

    $\mathbf{m}$ += r

  end for

end for

Niepert et al. work [456]

Another early work on molecular representation learning is the study of Niepert et al. [456], which has the benefit to be pedagogical. Conceptually, this study generalises the work of Duvenaud et al. [287] by introducing a stride bigger than one and a receptive field bigger than 1-hop-neighbourhood. Indeed, the local receptive field around a node of interest can be seen as a subgraph composed of the current node and

a relevant subset of context nodes in the neighbourhood.

Regarding the algorithm, their approach consists of 3 steps. First, they determine the nodes of the graph for which an abstract representation is learnt via a random walk procedure (in the case the graph is too big, we can avoid considering every node to stay scalable). In a second step, a neighbourhood is created for these nodes by extracting the k closest nodes via a breath-first search and normalising these. In a third step, the obtained graph local sequences are fed into a conventional 1D convolutional neuron network for prediction.

The main drawback which makes this approach unsatisfying is the normalisation step of the neighbourhood which requires a node ordering procedure. Indeed, the normalisation step requires the definition of a node ordering for the nodes in the neighbourhood such that similarly ranked nodes have similar structural roles. Such a node ordering procedure is a generalisation of the classical "graph canonisation problem" which aims at defining a unique mapping from the graph representation to vector -i.e. ordered- representation such that nodes with similar structural roles are positioned similarly in the vector representation. It is an NP-hard problem, so they use efficient approximation techniques suitable for restrained sets of graphs. The requirement of a node ordering as a pre-processing step prevents the approach from being learnable end-to-end.

Kearnes et al. work [300]

The work of Kearnes et al. [300] is also seminal in the field of molecular graph representation learning.

Compared to previous approaches, one of the main interests of this paper is to carefully discuss the original attributes of nodes and edges. They also point out the importance that the molecular representation must be independent of any atom ordering (as well as neighbourhood aggregation).

More importantly, they present a method for graph-level representation learning that captures the distribution among the graph nodes of each learned feature. More precisely, their method models each feature distribution by an approximated frequency histogram of n bins (n being a parameter). Thus the graph level representation of this approach is of size "n×dimension of node representation".

The node embeddings are updated via a message-passing-like algorithm by neuron modules which, in essence, perform neighbourhood aggregation (the model is displayed in Fig C.2). Also, edge feature vectors are considered and updated.

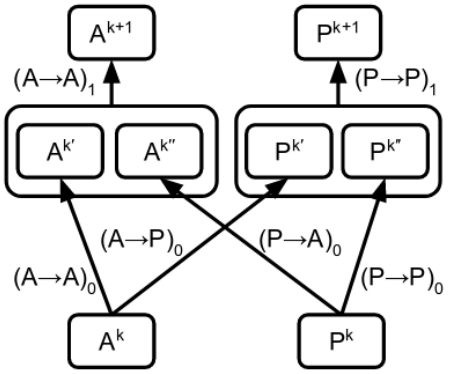


Figure C.2. Atom and bond representation update in [300]. This module takes matrices A_k and P_k (containing atom and bond features, respectively) and combines $A \to A$, $P \to P$, $P \to A$ and $A \to P$ operations to yield a new set of atom and bond features (A_{k+1} and P_{k+1}, respectively). $A \to A$ and $P \to P$ modules are learnable ReLU layers ($output = ReLU(W.input + b)$). $P \to A$ and $A \to P$ modules are the sum of the transformation of each input by a ReLU layer.

Coley et al. work [298]

Coley et al. [298] also produced a significant work in the field of molecular graph representation. Compared to previously presented approaches, their main contribution is showing the efficiency of using relevant atomic and bond featurisation as primary attributes.

Their approach is conceptually a GNN. Their encoder is, in essence, based on neighbourhood aggregation but has the particularity to build and aggregate a graph level representation after each node updates, like in the case of the neural fingerprints of algorithm 3. The algorithm they considered in this study is reported in alg. 4. Unlike to the neuron fingerprints, atom embeddings are not updated just by summing those of the neighbours but via a learnable linear combination of the neighbours passing through a nonlinear activation.

This study also holds an exciting discussion on the visualisation of molecular embeddings. They use "masking" (see appendix A.8.5) to identify what initial part of the molecules is essential for the relevancy of the prediction.

They also discussed transfer learning with alg. 4. Indeed, the authors first pre-trained the encoder network weights on a chemoinformatics prediction task and then re-trained the overall network on a new loosely-related chemoinformatics task. However, they observed relatively little improvement. Then they also tried hard-sharing multitask learning (see appendix A.8.3). It resulted in worse performance than in the singletask setting. They conclude that future work should focus on making MT beneficial by working on the model architecture (improving flexibility after the convolution layer for instance), on loss weighting (to account for the sample size), or on a more suitable learning schedule. We discuss such aspects in appendix A.8.3.

Algorithm 4 Graph convolutional network in [298]

Require: Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\mathbf{x}_v, \forall v \in \mathcal{V}$ and $\mathbf{x}_{vv'}, \forall vv' \in \mathcal{E}$; depth L; learnable weight matrices $(\mathbf{W}_{emb}^{(l)}, \mathbf{W}_{up}^{(l)}, \mathbf{b}_{emb}^{(l)}, \mathbf{b}_{up}^{(l)})\ \forall l \in 1, \dots, L$

initialisation of graph-level representation: $\mathbf{m} = 0$.

initialisation of node-level representation: $\mathbf{h}_v^{(0)} = \mathbf{x}_v\ \forall v \in \mathcal{V}$

for layer: $l = 0 \dots L$ **do**

  for atom: $v \in \mathcal{V}$ **do**

    $\mathbf{f}_v^{(l)} = softmax(\mathbf{W}_{emb}^{(l)}.\mathbf{h}_v^{(l)} + \mathbf{b}_{emb}^{(l)})$

    $\mathbf{h}_v^{(l+1)} = \sum_{v' \in \mathcal{N}(v)} (\mathbf{W}_{up}^{(l)}.(\mathbf{h}_{v'}^{(l)}, \mathbf{x}_{vv'}) + \mathbf{b}_{up}^{(l)})$. (Bond attributes $\mathbf{x}_{vv'}$ are not updated and thus persist throughout this convolution process)

  end for

  $\mathbf{m}$ += $\sum_{v \in \mathcal{V}} \mathbf{f}_v^{(l)}$

end for

Other works explored frameworks different from the one of convolutional networks to build a learnable graph representation.

C.3.2 Approximate inference on undirected graphs

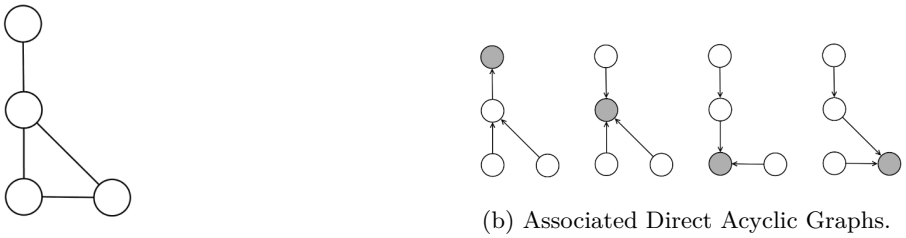
The work of Dai et al. [296] considered the framework of approximate inference on undirected graphs. Indeed a molecular graph is by essence an attributed undirected graph. In addition to the nodes of the molecular graph, they added a latent (i.e. unobserved) node to each atom nodes in the undirected graph. Thus atoms i have an attribute vector $\mathbf{x}_i$ and a latent state vector $\mathbf{h}_i$. They described the latent state not only by a vector but by a distribution.

The goal is to compute the posterior distributions $p(\mathbf{h}_i|\mathbf{x}_i)$ (probability of hidden state on atom i with attributes $\mathbf{x}_i$).

Exact computation of the latent state can only be performed in the case that the graph has a tree structure via message passing, but in most cases, embedding distributions are approximated by mean field approximation or the junction tree algorithm. In practice, approximation by mean field means we approximate $p(\mathbf{h}_i|\mathbf{x}_i)$ with a product of independent components that we write $\mu_i = \tilde{f}_{\mathbf{W}}(\mathbf{x}_i, \{\mu_j\}_{j \in \mathcal{N}(i)})$. In this paper, they parameterise $\tilde{f}$ by an ANN layer. Finally, the molecule-level representation is given by the sigmoid transformation of the sum of node embeddings: $\phi(x) = \sigma(\sum_i \mu_i) \in \mathbb{R}^d$.

Thus we observe the latent embedding framework with mean field approximation is conceptually strictly equivalent to the spatial message passing-like algorithm. Indeed, each nodes vector features are updated based on the aggregation of their neighbourhoods.

Note they also propose an approximation based on the "loopy belief propagation" algorithm which is another widely used variational inference technique.



(a) An Undirected Graph

(b) Associated Direct Acyclic Graphs.

Figure C.3. Decomposition of a molecular graph into DAGs. (picture from [297])

C.3.3 RNN on directed acyclic graphs

Baldi et al. [297] explored the framework of message passing on directed acyclic graphs (DAGs). Again, molecules naturally form undirected graphs (UG). Thus this method appears a priori unsatisfying but we introduce it here as exploratory perspective.

Let us set that molecules have $|V|$ atoms. In this framework, they build $|V|$ DAGs per molecule by choosing each atom successively as a terminal node and directing all edges to the selected terminal node while leaving the derived graph acyclic (see Fig. C.3 for example). Then, an RNN is applied on each DAG, i.e. a learnable neuron module passes the information from leaf nodes to the root node.

Finally, the feature vectors obtained at the N terminal nodes of the N DAGs are collected into one molecular feature vector by summation. This molecular representation is injected into a final neuron layer for prediction (molecular solubility in this paper).

This method has the inconvenience to build $|V|$ DAGs (and run an RNN on each) from an undirected graph $G = (V, E)$. Hence, it requires some heuristics and is non-natural. The need to consider an RNN approach on DAG built from a UG is not justified, compared to considering a CNN approach directly on the UG.

Later, Baldi et al. [457] showed that graph convolution based approach performs generally better than the DAG-RNN method even though both types of approaches reach comparable results on almost all data sets. Theoretically, anything that can be computed using one of the approach can also be obtained using the other, as a consequence of the universal approximation theorem of neuron networks [398, 399].

C.3.4 Convolutional neuron networks on Lewis formula images

To conclude, we would like to introduce a neuron network approach operating on other molecular representation than the molecular graph or 1D representation encoding the graph (like SMILES).

In particular, Goh et al. [458] worked directly on 80*80 pixel pictures in greyscale representation of molecular graphs, as shown in chemical databases. Then, standard CNNs on images can be applied to build a prediction model.

Again, this approach is adapted from another field, and so it is not designed intentionally for encoding graphs. Although not satisfying a priori, the benefit of this approach is its minimal amount of feature engineering and required chemical knowledge. However, it brings others issues from applying CNN on

images, such as "data augmentation" for example. Authors reported that this approach exhibits similar performance, on some chemoinformatics prediction tasks, to an approach based on conventional ECFP4 fingerprints using multitask DNNs [458], in other words, state-of-the-art performances.

Appendix D

Analysis of graph representation learning

Some recent works proposed a fundamental analysis of graph neuron networks.

These works can help the design of more efficient neuron architectures, as the field of GNN is mostly driven by intuition and experimental trial-and-error. In addition, the number of "different" graph neuron network can be arbitrarily extended by slight architecture modifications, as presented in Section 6.2.

Furthermore, there is a fundamental and practical question we would like to address: is the choice of the architecture predominantly crucial or, is it thanks to the learning of model parameters that GNN can reach high accuracy?

Intuitive discussion

First, the iterative update of node representations renders the features of neighbouring nodes similar, thus easing the prediction task, since the diversity of node representation is reduced. However, it also brings the issue of over-smoothing. If a GNN is deep (i.e. composed of many node updates), the node embeddings may be over-smoothed, and nodes may become indistinguishable.

These gives an intuition on why the early work of Kipf et al. [322] found that the best performances are reached by a network with no more than two layers. Note also that, intuitively, message passing-based GNN assumes that nearby vertices on a graph tend to share the same label.

Analogy with the 1-WL algorithm

Some other fundamental works [350, 351, 322] showed the analogy between graph neuron networks and the Weisfeiler-Leman (WL) graph isomorphism test. They show that a GNN using the standard update rule in alg. 1 can at best reach the discriminative power of the WL test.

Recall that state-of-the-art kernels based on the WL algorithm have also been proposed (see section 2.2).

Let us briefly recall the 1-dimensional Weisfeiler-Leman graph isomorphism heuristic [106] presented in Section 2.2. The 1-WL algorithm generates node representation through an iterative relabelling scheme.

Indeed the 1-WL algorithm summarises the substructures present in a graph through iterative update of nodes labels (or colouring) by aggregating over the labels of nodes in the direct neighbourhood.

Hence, explicitly, a graph convolutional network is a differentiable and adaptive formulation of the 1-WL algorithm [322]. Indeed, GNNs are trained in an end-to-end fashion together with the parameters of the prediction sub-network, expecting greater adaptability and generalisation compared to the kernel counterpart.

On top of this observation, Xu et al. [350] emphasise that classical GNN is as expressive as the 1-WL algorithm, when an injective and expressive function models the aggregation function.

Indeed, GNN are built by neighbourhood aggregation. Thus, a discriminative GNN should, by definition, never map two different neighbourhoods to the same representation. Hence, its aggregation function must be injective. In particular, representative power refers here [350] to the discriminative power, i.e. the ability to distinguish all pairs of graph topology in the input set. We think this notion of representative power is limited, because it should also refer to the ability to encode "smoothly" the similarity between non-isomorphic attributed graphs. However, we agree that this aspect is much harder to assess theoretically.

They recall that multi-layer perceptrons (MLPs) can model and learn universal injective functions, based on the universal approximation theorem [398, 399].

They also highlight that mean- and max-pooling operations are order invariant multi-set functions, but are not injective. Indeed, the mean captures the distribution of elements in the multi-set, but not the exact multi-set. This would perform well if the mean structural information is more important than the exact structure of the graph, for the considered prediction task. The mean is also powerful when node features are diverse and rarely repeat. In practice, this is why it is competitive with regards to other aggregation functions. Max-pooling does not capture the exact structure neither, but may be suitable for tasks requiring the identification of key elements, rather than keeping the exact structure. This may be the case for molecular bioactivity, which mainly relies on some pharmacophore or functional groups.

In the same perspective, Moris et al. [351] emphasise that GCNs can, at best, be as discriminative as the 1-WL, and therefore cannot be more powerful at distinguishing non-isomorphic sub-graph. Regarding this aspect, the power of 1-WL has been characterised [105], and some of its limitations are known. For instance, they are not capable of capturing simple graph-theoretic properties like triangle counts (which is essential in the case of social network analysis for instance).

Therefore, the authors proposed a GCN based on the k-WL algorithm, which is more expressive than the 1-WL algorithm. The k-WL is a generalisation of the 1-WL which updates tuples (of size k) of nodes instead of individual nodes. Hence, this variant performs message passing between subgraphs, rather than between individual nodes. This higher-order form of message passing can capture key coarse-grained structures that are not identifiable at the node level. They named k-GCN the GCN based on the k-WL algorithm.

They tested k-GCN on three benchmark datasets and k-GCN barely shows 1 point of performance improvement with regard to those obtained by kernel-based methods. Their analysis was interesting, but the datasets they considered did not reveal its relevancy.

Analogy with kernels

An important work on GCN has been done by Lei et al. [459]. They show that the feature map generated by variants of standard GCNs lie in the same Hilbert space of popular graph kernels.

In this paper, they derived neuron network-based update rules for GCN that theoretically ensure that the final output state belongs to the RKHS of some kernels (the NN modules are actually derived based on the kernels). These update rules are not those considered in standard GCNs, but are remarkably similar and can even appear more sophisticated than those of standard GCNs.

However, they do not compare performance of their GCNs to those of other prevalent GCNs (like graphSAGE [291] and GAT [295]) on benchmark datasets.

Partial conclusion

We have analysed the characteristics and the representation power of GCN from different perspectives. In essence, a standard GCN is a differentiable and parametric generalisation of the 1-WL algorithm on graphs. Besides, the WL algorithm was also used to build state-of-the-art graph kernels. Although a better flexibility is expected by standard GCN, the representation power, in terms of distinguishing different graph topology, is not expected to be enhanced compared to state-of-the-art graph kernel-based methods.

This fundamental discussion on GCN is modest. It can guide our intuition but we cannot conclude on the potential of GCN to overtake kernel-based methods. We cannot neither conclude on the practical question that we asked at the beginning of this appendix. This is partly why we investigated intensively the recent developments on GCNs for DTI prediction in chapter 8.

Appendix E

Inverse Drug Design via automatic molecule generation

As mentioned in Section 1.2, computational approaches are expected to assist and guide the drug discovery process.

Inverse drug design (or equivalently *de novo drug design*) offers a relatively new way of probing the chemical space before wet-lab experiments. It has gathered an extraordinary research effort the past two years [460, 461, 462, 463, 464, 465, 466, 467, 468, 469], and continues to be appealing [470, 471]. To our knowledge, the only review available on inverse drug design is the one of Sanchez et al. [30], published a year ago and "already outdated".

Inverse drug design inverts the classic objective of standard ML methods for drug design. While ML methods are usually used to learn a function mapping molecular compounds to the property of choice, inverse drug design starts from the desired property and searches for molecular structures fulfilling this requirement.

Note that inverse drug design can be performed with standard approaches that map molecular structures to chemical and biological properties. Indeed, predicting molecular properties of a large initial library of molecules (randomly chosen or built based on expert intuitions) can be seen as a greedy approach for inverse drug design.

An early approach for inverse drug design uses evolutionary strategies to explore and map the chemical space [472, 473]. They consider structural fingerprints as "genotypes" that can be perturbed ("mutated") to improve an objective function value ("fitness" of the "genotypes"). The exploration of the space of parameters involves heuristics and procedures inspired by natural evolution.

Today, inverse design approaches usually rely on deep generative models which are a particular class of deep learning methods modelling the underlying probability distribution of both structure and property, in a non-linear fashion. In the following sections, we introduce the neuron architectures of chemical deep generative models, discuss the procedure to bias the generative process toward desired chemical properties,

and finally, propose a comparison of the methods introduced in this appendix.

This survey does not aim at explaining all the tools to perform inverse drug design but is restrained to a broad conceptual introduction. The interested reader may train in deep learning and reinforcement learning to be able to investigate the following approaches.

E.1 Generative models

First, recall that a generative model attempts to determine a joint probability distribution of observing both the molecular representation x and the physical property y ($p(x, y)$). We immediately see that $p(x, y)$ can be used for inverse design by conditioning on the property y ($p(x|y)$).

Generative models based on deep learning can non-linearly link the molecule description x and the physical property y, while performing representation learning.

Deep generative models can be built based on three neuron network architectures: auto-encoders (AE), variational auto-encoders (VAE), and generative adversarial networks (GAN). These methods are introduced in appendix A.8.2.

Recurrent neuron networks (RNN), introduced in appendix A.8.2, are also often used to build generative models for SMILES strings, but also per molecular graphs.

E.2 Molecule generation approaches

In this section, we introduce some of published works on automatic molecule generation that we organise depending on the numerical description of chemicals they consider: a vector of features (fingerprints), a SMILES string or an undirected graph.

E.2.1 Molecular fingerprint generation

Few works [474, 475] proposed to generate compounds in the form of structural fingerprint vectors. In particular, Karudin et al. [475] implemented VAE taking as input the Molecular ACcess System (MACS) fingerprints. The same authors proposed later to train a standard auto-encoder jointly with a discriminator network, such that the encoder both minimises the reconstruction error and maximises the discrimination error (to generate reasonable fingerprints, i.e. fingerprints which correspond to realistic molecules).

As there is no direct way to recover a molecule from a continuous structural fingerprint vector, compounds are often retrieved by searching for the closest molecule, in terms of fingerprints, in a large database. For this reason, encoding and decoding SMILES representation of molecules seems more suitable without entailing additional difficulties.

E.2.2 SMILES generation

Most of the studies about inverse molecular design [460, 461, 476, 477, 478, 479, 480, 481, 482] worked with the SMILES representation of molecules since it is particularly convenient.

Indeed, encoding SMILES strings via RNN and CNN is as straightforward as it is efficient (see Section 6.1). Also, decoding or generating SMILES can be performed easily with RNNs.

Indeed, an RNN can be used to estimate the probability distribution of the next state s_t in a sequence based on the previous state s_{t-1}. More precisely, in the case of a SMILES string, s_t corresponds to the SMILES character at position t. The RNN typically output a probability vector , as a proxy to $p(s_t|s_{t-1})$, whose elements sum to one and whose dimension is the size of the dictionary of possible characters in a SMILES string. Typically, the probability that the character s_t at position t is the k^{th} character in the SMILES dictionary follows $p(s_t = k|s_{t-1}) = \frac{exp(y^k_{t-1})}{\sum_k exp(y^k_{t-1})}$ where y^k_{t-1} is the k^{th} value of the output vector of the $(t-1)^{th}$ RNN cell [460].

The RNN is typically trained to recover the successive characters of millions of SMILES taken from a large database of molecules, such as ChEMBL.

E.2.3 Graph generation

As already mentioned in this manuscript, SMILES representation is dependent on the canonicalisation method used, meaning that similar structures have different SMILES strings.

Thus, recent works have focused on building models to generate undirected graphs directly.

First, Jaakkola et al. [483, 468] presented the junction tree VAE (JT-VAE) which is a two-stage approach based on VAE. In the first stage, the JT-VAE generates a junction tree whose role is to represent the scaffold of subgraph components (building blocks automatically extracted from the training set using tree decomposition) and their relative arrangements in a hierarchical manner. In the second stage, the sub-graphs are assembled into a coherent molecular graph based on the tree structure defined in the first step.

This approach can leverage structure at different scales (atoms, chemical substructure etc.) which enables macro-molecule generation, such as proteins. Also, such a tree-structured generative process might benefit from a reduced complexity but also a good predictive performance.

Simultaneously, several authors proposed similar approaches that decompose the graph generation process into a sequence of node and edge generation processes. This sequential generation is conditioned on the graph structure generated so far [470, 464, 471, 467, 484]. Indeed, after each step of graph expansion (addition of nodes and/or edges), a GCN is typically used to update nodes attribute so that each node attributes contains the information about the current graph structure.

For instance, You et al. [471] formulated an iterative graph generation process as a general decision process $M = (S, A, P, R)$, where (i) $S = \{s_t\}$ stands for the set of states s_t that consists of all possible intermediate graphs, (ii) $A = \{a_t\}$ is the set of actions that trigger modifications on the current graph at each time step t, (iii) P stands for the transition probabilities $p(s_{t+1}|s_t, \ldots, s_0, a_t)$ (typically modelled by

RNN) that specifies the possible outcomes of carrying out an action a_t (i.e. monitor the modification of a graph at each time step) and finally (iv) $R(s_t)$ is a reward function that specifies the reward of each state s_t (in order to optimise the transition probability model). The procedure to generate a graph can thus be described by a sequential trajectory $((s_0, a_0, r_0), \ldots, (s_n, a_n, r_n))$, where s_n is the final generated graph. The reward R can typically guide the generation toward realistic chemical graphs and graphs with desired properties.

Alternatively, Li et al. [467] designed a sequential process which generates one node at a time and edges one by one. In essence, their graph generation process relies on three graph neuron network modules ($f_{add\ node}$, $f_{add\ edge}$ and f_{nodes}) that provide probabilities for each structure building event, which are, (i) add a new node ($f_{add\ node}$), (ii) add a new edge (probabilities provided by $f_{add\ edge}$), and (iii) pick one node to connect to the new node generated after (i) (f_{nodes}).

Finally, some have proposed to generate graph directly all at once [465, 463].

First, Simonovsky et al.[465] used a VAE made of a graph neuron network (encoder) and an MLP (decoder). The MLP decoder is a probabilistic graph decoder whose final layer outputs a probabilistic fully-connected graph $G = (A, E, F)$ of a predefined maximum size directly all at once so that all variables (A, E, F) are assumed independent. A is the predicted adjacency matrix which contains both node and edge probabilities, and E and F are respectively the edge and node probabilistic attribute tensors. The existence of nodes and edges is modelled with Bernoulli variables, whereas node and edge categorical attributes are multinomial variables and node and edge continuous attributes are modelled with Gaussian variables.

A more recent study proposed molGAN [463] (see fig. E.1) which combines a GAN network, composed of a generative network and a discriminator network [485] (cf. appendix A.8.2), with a reward network. The generative network is an MLP that takes as input a D-dimensional vector sampled from a standard normal distribution and outputs two continuous and dense objects: the atom type tensor $F \in \mathbb{R}^{N*T}$ and bond type tensor $E \in \mathbb{R}^{N*N*Y}$ (where N is the number of atoms, T the number of atom's category and Y the number of edge's category). Thus the graph is generated all at once. Note that the adjacency matrix is implicitly defined by the bond type tensor E.

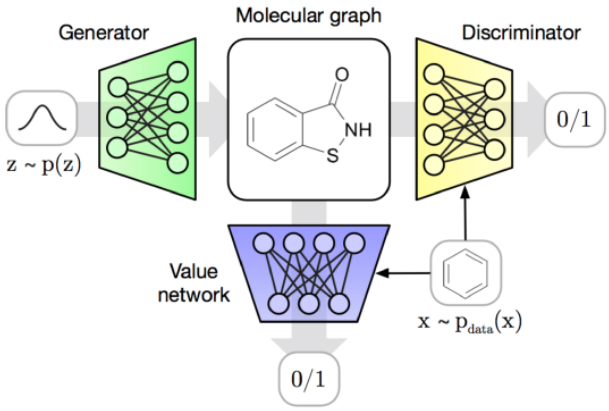


Figure E.1. The molGAN approach [463]

In such an approach, the generator is trained to full the discriminator that is optimised to distinguish generated molecules from real molecules taken in the ChEMBL database. Therefore, the generator is trained to generate realistic molecules. Simultaneously, the generator is trained to maximise the reward outputted by the reward network.

In the next subsection, we discuss in more details several strategies to bias the generation process toward realistic molecules and to perform goal-oriented inverse drug design.

E.3 Biasing the molecule generation process

In this subsection, we introduce several approaches for guiding chemical generative models towards molecules with desired chemical or biological properties, such as drug-likeliness or a high affinity with a target protein.

E.3.1 Guiding molecular generators via overfitting

An early work [486] adopted a simple guiding strategy which consists in retraining a pre-trained auto-encoder a dozen times only with the molecules (initially in the database) known to fulfil to the desired the property. In other words, this procedure overfits a generative model on molecules with a desired feature to enforce generating molecules with the same feature.

This strategy appears to be relevant only for properties concerning a large amounts of available or predicted molecules.

E.3.2 Guiding molecular generators via incorporation of the property in the input

A second direct attempt to guide the molecular generation process with auto-encoders is to concatenate the information of the considered property to each input graph and to each latent representation (or even

to every layer), such that both the encoder and the decoder are encouraged to exploit information in the label [465, 487, 488].

However, when it comes to generating new objects with the desired property, authors have proposed to first sample the latent representation and then pass the latent vector concatenated with the desired property through the decoder.

This process implies the independence of the latent representation and the property of interest, which is inconsistent in most cases.

More powerful guiding strategies, based on Bayesian optimisation or reinforcement learning have also been proposed.

E.3.3 Guiding molecular SMILES generators' latent encoding via Bayesian optimisation

When using variational auto-encoders, a strategy to generate compounds with a desired property is to automatically optimise, typically via Bayesian optimisation, the continuous latent representation so that the latent representation corresponds to a molecule with the property of interest. Indeed, Bayesian optimisation algorithms optimise objective functions over continuous domains. They build a surrogate for the objective function and quantify the uncertainty in that surrogate, using machine learning techniques such as Gaussian processes, to sample the input space and quickly find an optimum (see appendix A.8.4).

One of the first approaches for inverse drug design [460] used this strategy. The authors used a VAE to encode and decode SMILES representation of molecules. The encoder was a CNN, and the decoder was made of fully connected layers followed by successive Gated Recurrent Unit (GRU) layers. They used black box Bayesian optimisation to optimise the latent representation so that it corresponds to a molecule with the desired biological property. In the following, we call this approach VAE-BO.

Later on, Hernadez et al. [461] proposed a slight improvement of this approach: they considered "constrained" Bayesian optimisation to generate more realistic molecules. In addition to optimising the latent molecular representation to maximise the probability to match a biological property, this approach learns what part of the latent space is "forbidden" because it encodes unrealistic molecules.

E.3.4 Guiding molecular generators via reinforcement learning

Alternatively, most of the current works consider reinforcement learning (RL) to guide generative processes.

Typically, RL strategies use a reward network that treats the generator as an agent which must learn how to take actions (add characters to a SMILES string or expand/terminate a molecular graph) within a task (SMILES or graph generation) in order to maximise some reward (chemical properties).

Introduction to reinforcement learning

RL optimisation is meant to guide a generator G (or a "policy network π" in the RL nomenclature), with parameter Θ, that takes as input the previous state s_{t-1} and estimates the probability distribution $p(a_t|s_{t-1})$

of the next action a_t. The next action a_t is then sampled from $p(a_t|s_{t-1})$. We note $A = (a_1, a_2, \ldots, a_T)$ the sequence of actions (called "an episode" in the RL nomenclature), T the number of steps of the generative process, and $r(\cdot)$ a scoring function that rates the desirability of an action or a full episode A. The major issue in the case of chemical generation is that it can assign rewards only once the molecule (SMILES or graph) is completed. Therefore, we cannot easily define a reward $r(a_t|s_{t-1})$ which acts as a measurement of how good it was to take an action at a particular state.

If both actions and states are generated by the agent itself, the learning is referred to as "on-policy", and as "off-policy" otherwise. The strategy defined above belongs to the on-policy framework, whose policy is a neuron network G with parameter Θ. On-policy RL can be performed in "value-based RL" or "policy-based RL".

Value-based RL strategy consists in optimising the policy π_Θ to return an action a_t by maximising the expected return based on the previous state s_t, defined by $J(\Theta) = \mathbb{E}[r(a_t)|s_{t-1}, \Theta]$ (where $r(a_t)$ is the reward associated with action a_t). Hence, this strategy considers action-level rewards, instead of episode-level rewards, to optimise π_Θ.

In the case of SMILES generation, we can typically use Monte Carlo Tree Search (MCTS) [489] which simulates several possible completions for SMILES by constructing a search tree generated by a rollout procedure. Based on the tree, we can score a partial SMILES by weighting the evaluated rewards of several possible completions based on their likeliness. In other words, the space of SMILES strings is represented as a search tree where the i^{th} level corresponds to the i^{th} SMILES character, and each node corresponds to one symbol, so that a path from the root to a terminal node corresponds to a complete SMILES string. The rollout procedure, which evaluates the likeliness of the next SMILES character based on the partial SMILES, is typically an RNN pre-trained on a extensive database of SMILES strings.

Alternatively, policy-based RL directly optimise the stochastic policy π_Θ according to some scoring function $r(\cdot)$ defined at the episode level. Several objective functions can be considered to optimise the policy via gradient-based methods.

For instance, the augmented episodic likelihood objective function optimises the episodic log likelihood of the policy $log[p_{\pi_\Theta}(A)]$ to maximise the episodic reward $r(A)$ and to anchor the policy π_Θ to an a priori policy π_{prior} (pre-trained on large databases to learn both the syntax and distribution of molecular structure). In this case, the objective function is defined by

$$J(\Theta) = -[log[p_{\pi_{prior}}(A)] - log[p_{\pi_\Theta}(A)]]^2 + \alpha r(A)$$

where α is a scalar controlling the trade-off between the two components (i.e. $r(A)$ and $log[p_{\pi_{prior}}(A)] - log[p_{\pi_\Theta}(A)]$).

Another popular policy gradient method is the REINFORCE [490] algorithm which weights the reward of episodes by their likeliness, without referring to a prior policy. Its associated cost function is :

$$J(\Theta) = -r(A) \sum_{t=0}^{T} log[p_{\pi_\Theta}(a_t, s_t)].$$

In practice, for automatic molecule generation, RL is either used to guide a pre-trained generator [471, 467, 463, 476, 477, 478] or to guide a generator to be trained jointly with a GAN [479, 480, 481, 482, 491, 463].

Guiding pre-trained generators via reinforcement learning

First, some authors [471, 467, 463, 476, 477, 478] proposed to explore the chemical space by combining RL and an RNN-based generator (to generate either SMILES [476, 477, 478] or graphs [471, 467, 463]). We use the acronym RNN-RL to refer to this type of approaches.

Such approaches are composed of two phases. In the first one, a generative network (or an "agent" in the RL nomenclature) and a molecular property prediction network are trained separately. They are jointly trained in the second phase. The trained generative model is used to produce novel chemically feasible molecules. The predictive model predicts the properties of each generated molecule and evaluates the agent's behaviour by assigning a numerical reward to individual generations.

Typically, the reward is the predicted probability value (between 0 and 1) to fit the desired biological property, or "-1" for unrealistic molecules.

In particular, in the case of SMILES generation, Yang et al. [476] first proposed to train the agent, made of stacked GRU layers, following a value-based RL strategy (and thus used MCTS, as previously mentioned), to guide the SMILES generation towards realistic and property-oriented compounds.

Later on, Olivecrona et al. [477] considered a similar agent pre-trained on the ChEMBL database. However, they re-trained the agent in the second phase in an on-policy fashion (with the REINFORCE objective function defined above) and standard gradient descent optimiser.

Finally, Popova et al. [478] pointed out that regular RNNs such as LSTM or GRU, are unable to generate valid SMILES efficiently because of their inability to count (valence for all atoms, ring opening and closure, bracket sequences with several bracket types for instance). Therefore, they proposed to use memory-augmented RNN like "Stack-RNN". Such memory-augmented RNN defines an additional differentiable neuron module on top of the standard GRU cell in which continuous vectors are inserted and removed. This is intended to learn to store meaningful long-range inter-dependences. They assessed the efficiency of the stack-RNN in their study. Indeed, the agent with the stacked memory generated molecules such that 95% of them were valid (only 86% for the agent without the stacked memory) and 14% were "similar" to molecules in the original ChEMBL training data set (twice more for the agent without the stacked memory). They used the fingerprint-based Tanimoto metric and a minimum similarity threshold set to 0.85 to assess drug similarity.

Guiding generative adversarial networks via reinforcement learning

To address the challenge of goal-directed generation of valid molecules, others [479, 480, 481, 482, 491] combined reinforcement learning and adversarial training. The adversarial approach is meant to keep the generation of molecules similar to the initial distribution of data (to generate realistic molecules), while

reinforcement learning guides the generation process to maximise some reward associated to the desirable properties. In the following, we use the acronym GAN-RL to refer to this approach.

GAN-RL contains three key elements: a discriminative network (a binary classifier that determines whether a molecule is likely to come from the initial distribution or not), a reinforcement component (quantifies the interest of a generated molecule) and a generator network (generates molecules). The generator is trained by maximising the following objective function: $\mathcal{L}(\Theta) = \lambda\mathcal{L}_{GAN} + (1-\lambda)\mathcal{L}_{RL}$ where $\lambda \in [0,1]$ regulates the trade-off between the two components (i.e. the GAN loss and the RL loss).

Compared to RNN-RL approaches, whose generation process is limited by the pre-training of the generator, GAN provides a convenient generative method to sample new molecules. However, GANs often suffer from mode collapse when the generator ends up exploiting repetitive patterns, or when the discriminator overwhelms the generator, which prevents the generator from improving its capacity. As introduced in appendix A.8.2, several tricks have been developed to partly address this issue.

In the case of SMILES generation, a series of studies [479, 480, 481] have explored the combination of GAN and RL.

Later studies [482, 491] proposed several improvements, such as more powerful generator network to prevent the discriminator model from overcoming the generator during training. More precisely, they used GRU cells equipped with an external memory that are trained in an end-to-end manner, to account for short and long-term dependencies (similar to the stack-RNN of Popova et al. [478]).

Finally, the major study of Kipf et al. [463] explored the combination of GAN and RL to generate undirected molecular graphs all at once. The second main difference with the previously mentioned GAN-RL approaches is that they use a different RL gradient policy than REINFORCE. Indeed, due to the high dimensional action space when generating full graphs, they found that a stochastic policy $\pi_\Theta(s) = p_\Theta(a_t|s_{t-1})$ (a parametric probability distribution with parameter Θ) trained with REINFORCE converges poorly. Therefore, they used a deterministic policy, $\pi_\Theta(s) = a$, modelled by a reward neuron network (a graph neuron network [304]) and used the appropriate Deep Deterministic Policy Gradient (DDPG) [492].

Their approach follows two phases. The reward network is trained independently in the first phase. The generator, an MLP which outputs two continuous tensors (i.e. the atom types and bond types tensors), is also trained independently in the first phase to fool a discriminator (a graph neuron network based on [304]) which tries to segregate molecular graphs taken in the ChEMBL database from those generated from random noise. In the second phase, the discriminator is trained using the WGAN loss [485] while the generator, parameterised by Θ, optimises a linear combination of the WGAN objective and the RL objective: $\mathcal{L}(\Theta) = \lambda\mathcal{L}_{WGAN} + (1-\lambda)\mathcal{L}_{RL}$ ($\lambda \in [0,1]$ regulates the trade-off between the two components).

Again, optimising the WGAN objective aims at guiding the generator toward realistic molecules, whereas the RL objective is meant to maximise some reward to guide the generator toward molecules with the desired properties. Interestingly, they observed that setting $\lambda = 0$ (only the RL objective in the second phase) resulted in the "best" generative models in terms of validity (ratio of the number of valid molecules over all generated molecules), novelty (ratio of the set of valid molecules that are not in the original training dataset

over the total number of valid molecules), and uniqueness (ratio of the number of unique molecules over valid molecules). This means that pre-training the generator in the first phase to fit the distribution of realistic molecules, and then to retrain it with RL only, led to the best model. This crucial observation sharply put into perspective the efficiency of GAN to generate realistic molecules compared to RNN-RL models, such as the one proposed by Popova et al.[478].

In the next subsection, we compare the methods introduced in this appendix.

E.4 Evaluation of molecule generators

Before attempting to compare the approaches we have introduced so far, we discuss several metrics that can be used.

E.4.1 Metrics for de novo drug design models.

Validity, novelty and uniqueness

Several studies [463, 469, 493] have compared methods based on validity, novelty and uniqueness.

Validity is the ratio between the number of valid compounds (syntactically correct) over all generated molecules. It quantifies the ability to generate realistic molecules.

Novelty is defined as the ratio between the set of valid samples that are not in the training dataset over the total number of valid samples. It quantifies the ability to explore unknown parts of the chemical space.

Uniqueness measures the ratio between the number of unique samples and valid samples. It quantifies the degree of variety during sampling. Another similar measure is the internal diversity measure [493] which is defined as the following: $Div(G) = 1 - \frac{1}{|G|^2} \sum_{m_1, m_2 \in G} Tanimoto(m_1, m_2)$ where G is the set of generated molecules and $Tanimoto$ is the fingerprint-based Tanimoto similarity measure between molecules.

Diversity of the generated molecules

To quantify the diversity of generated molecules relatively to the original data set used to train the generative model, some [469] proposed to use the KL divergence between the distribution of generated molecules and the original distribution in the training set. This measure assesses if the set of generated molecules is as diverse as the training set.

Another similar metric is the nearest neighbour similarity (NNS) [493] which is defined as the average of the Tanimoto similarity $Tanimoto(\cdot, \cdot)$ between fingerprints of a molecule m_G from the generated set G and its nearest neighbour molecule m_R in the training dataset R:

$$SNN(G, R) = \frac{1}{|G|} \sum_{m_G \in G} \max_{m_R \in R} Tanimoto(m_G, m_R)$$

.

Frechet ChemNet Distance

Furthermore, Preuer et al [494] proposed the Frechet ChemNet Distance (FCD) which is another measure of how close the distributions of generated compounds is to the distribution of molecules in the training set.

Interestingly, as molecular descriptors, they considered the molecular representation encoded by the ChemNet multitask deep neuron network [354] trained to predict the biological activities of drugs. It incorporates a wide variety of important chemical and biological features into a single vector representation.

The FCD compares the generated distribution G with the training distribution R via the descriptors obtained from the ChemNet network following:

$$FCD(G,R) = ||\mu_G - \mu_R||^2 + Tr(\Sigma_G + \Sigma_R - 2(\Sigma_G\Sigma_R)^{1/2})$$

where μ_G, μ_R are mean vectors and Σ_G, Σ_R are covariance matrices of the activations of the ChemNet's encoding layer on the sets G and R respectively.

They showed that this metric is relevant by considering training datasets biased on purpose (toward low synthetic accessibility, low drug-likeness, ligands of a given protein family, etc...), and by showing that FCD can catch all biases to judge whether created molecules are consistent or not with respect to these biases.

Biological relevance of the molecules

Finally, since the goal of inverse drug design is to automatically generate small compounds with desired properties, we can use a score that reflects how well a molecule fulfils the required properties.

Such properties can be biological properties but also properties to judge whether molecules are reasonable or not such as (i) the quantitative estimate of drug-likeness (QED) [495] which is an empirical measure of drug-likeness, (ii) the water-octanol partition coefficient (logP) which measures how a molecule partitions between water and octanol, and (iii) the Synthetics Accessibility Score (SAS) [496] which estimates whether a molecule is synthetisable based on known reactions. Note that SAS follows the distribution from the commercially available compounds, which is not a well-defined concept [497].

In the last part of this chapter, we report the comparison of the chemical generative models introduced above.

E.4.2 Brief comparison of de novo drug design models.

First, we observed that most current molecular generation applications use the ZINC or ChEMBL databases. The ZINC database records about 35 million compounds, whereas the ChEMBL is cleaner but records "only" a couple of million of compounds (see Section 2.3.1).

Regarding generative models of SMILES-based representation of molecules, Polykovskiy et al. [493] showed that the VAE-based[460] model outperforms AAE[488] and GAN-RL[480] architectures on the Fréchet

ChemNet distance metric. This suggests that the VAE describes the chemical space better than other models. They also observed that all models generate valid molecules, meaning that they successfully learn SMILES syntax.

Another study [476] showed that RNN-RL models [476, 478] are significantly faster than VAE-BO based methods [460] at generating valid SMILES strings. Moreover, the improved RNN-RL method [478] exhibited impressive performance. They showed that 95% of all generated structures were valid and less than 0.1% of generated structures were identical to those in the training data set.

Regarding generative models that output graphs all at once, the study of Kipf et al. [463] compared molGAN to other methods via the validity/novelty/uniqueness metrics. Among related works, they considered a GAN-RL method [480] and a VAE-BO method [460] both working on SMILES strings and another graph-based generative model using VAE [465].

First, molGAN outperforms the GAN-RL method (which is very similar to molGAN) handling SMILES for the three objective scores. This means that it should be easier to optimise a molecular graph predicted as a single sample than to optimise an RNN model that outputs a sequence of characters. Moreover, molGAN achieves both high validity and novelty scores, which is not the case of all the other methods. However, molGAN's uniqueness is lower than that of all other baselines because of the collapse mode of GAN.

To conclude, we recommend the use of the improved RNN-RL method of Popova et al. [478]. The molGAN approach appears promising but it must be corrected for mode collapse. Note also that both studies recommend to fit the generator on a large database of molecules (to learn to generate realistic molecules), and then to retrain the generator with RL only (to bias the generator towards compound with the desired property), instead of optimising both objectives simultaneously.

Bibliography

[1] Joseph A DiMasi, Ronald W Hansen, and Henry G Grabowski. The price of innovation: new estimates of drug development costs. *Journal of health economics*, 22(2):151–185, 2003.

[2] Andrew L Hopkins. Drug discovery: predicting promiscuity. *Nature*, 462(7270):167, 2009.

[3] Chloé-Agathe Azencott. The drug discovery pipeline. `http://cazencott.info/dotclear/public/lectures/s1133_2018/2018-10-29-chemoinformatics-slides.pdf`, 2018.

[4] Jürgen Drews. Drug discovery: a historical perspective. *Science*, 287(5460):1960–1964, 2000.

[5] Konrad H Bleicher, Hans-Joachim Böhm, Klaus Müller, and Alexander I Alanine. A guide to drug discovery: hit and lead generation: beyond high-throughput screening. *Nature reviews Drug discovery*, 2(5):369, 2003.

[6] Regine S Bohacek, Colin McMartin, and Wayne C Guida. The art and practice of structure-based drug design: a molecular modeling perspective. *Medicinal research reviews*, 16(1):3–50, 1996.

[7] Robert D Brown and Yvonne C Martin. The information content of 2d and 3d structural descriptors relevant to ligand-receptor binding. *Journal of Chemical Information and Computer Sciences*, 37(1):1–9, 1997.

[8] Johann Gasteiger and Thomas Engel. *Chemoinformatics: a textbook*. John Wiley & Sons, 2006.

[9] Evan E Bolton, Yanli Wang, Paul A Thiessen, and Stephen H Bryant. Pubchem: integrated platform of small molecules and biological activities. *Annual reports in computational chemistry*, 4:217–241, 2008.

[10] Anna Gaulton, Louisa J Bellis, A Patricia Bento, Jon Chambers, Mark Davies, Anne Hersey, Yvonne Light, Shaun McGlinchey, David Michalovich, Bissan Al-Lazikani, et al. Chembl: a large-scale bioactivity database for drug discovery. *Nucleic acids research*, 40(D1):D1100–D1107, 2012.

[11] Vivian Law, Craig Knox, Yannick Djoumbou, Tim Jewison, An Chi Guo, Yifeng Liu, Adam Maciejewski, David Arndt, Michael Wilson, Vanessa Neveu, Alexandra Tang, Geraldine Gabriel, Carol Ly, Sakina Adamjee, Zerihun Dame, Beomsoo Han, You Zhou, and David S. Wishart. Drugbank 4.0: shedding new light on drug metabolism. *Nucleic acids research*, 42(D1):D1091–D1097, 2013.

[12] UniProt Consortium. Uniprot: a hub for protein information. *Nucleic acids research*, 43(D1):D204–D212, 2014.

[13] Brian K Shoichet, Irwin D Kuntz, and Dale L Bodian. Molecular docking using shape descriptors. *Journal of Computational Chemistry*, 13(3):380–397, 1992.

[14] Gareth Jones, Peter Willett, Robert C Glen, Andrew R Leach, and Robin Taylor. Development and validation of a genetic algorithm for flexible docking. *Journal of molecular biology*, 267(3):727–748, 1997.

[15] Garrett M Morris, David S Goodsell, Robert S Halliday, Ruth Huey, William E Hart, Richard K Belew, and Arthur J Olson. Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function. *Journal of computational chemistry*, 19(14):1639–1662, 1998.

[16] Piotr S Gromski, Alon B Henson, Jarosław M Granda, and Leroy Cronin. How to explore chemical space using algorithms and automation. *Nature Reviews Chemistry*, page 1, 2019.

[17] Chloe-Agathe Azencott. *Statistical Machine Learning and Data Mining for Chemoinformatics and Drug Discovery*. PhD thesis, University of California, Irvine, 2010.

[18] Jean-Philippe Vert and Laurent Jacob. Machine learning for in silico virtual screening and chemical genomics: new strategies. *Combinatorial chemistry & high throughput screening*, 11(8):677–685, 2008.

[19] Justin S Smith, Ben Nebgen, Nicholas Lubbers, Olexandr Isayev, and Adrian E Roitberg. Less is more: Sampling chemical space with active learning. *The Journal of chemical physics*, 148(24):241733, 2018.

[20] S Joshua Swamidass, Joshua A Bittker, Nicole E Bodycombe, Sean P Ryder, and Paul A Clemons. An economic framework to prioritize confirmatory tests after a high-throughput screen. *Journal of biomolecular screening*, 15(6):680–686, 2010.

[21] Rachel A Powers, Federica Morandi, and Brian K Shoichet. Structure-based discovery of a novel, noncovalent inhibitor of ampc β-lactamase. *Structure*, 10(7):1013–1023, 2002.

[22] Thompson N Doman, Susan L McGovern, Bryan J Witherbee, Thomas P Kasten, Ravi Kurumbail, William C Stallings, Daniel T Connolly, and Brian K Shoichet. Molecular docking and high-throughput screening for novel inhibitors of protein tyrosine phosphatase-1b. *Journal of medicinal chemistry*, 45(11):2213–2221, 2002.

[23] Christopher A Lipinski, Franco Lombardo, Beryl W Dominy, and Paul J Feeney. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Advanced drug delivery reviews*, 23(1-3):3–25, 1997.

[24] Andrew L Hopkins. Network pharmacology: the next paradigm in drug discovery. *Nature chemical biology*, 4(11):682, 2008.

[25] Hongyi Zhou, Mu Gao, and Jeffrey Skolnick. Comprehensive prediction of drug-protein interactions and side effects for the human proteome. *Scientific reports*, 5:11090, 2015.

[26] Monica Campillos, Michael Kuhn, Anne-Claude Gavin, Lars Juhl Jensen, and Peer Bork. Drug target identification using side-effect similarity. *Science*, 321(5886):263–266, 2008.

[27] Ted T Ashburn and Karl B Thor. Drug repositioning: identifying and developing new uses for existing drugs. *Nature reviews Drug discovery*, 3(8):673–683, 2004.

[28] Dik-Lung Ma, Daniel Shiu-Hin Chan, and Chung-Hang Leung. Drug repositioning by structure-based virtual screening. *Chemical Society Reviews*, 42(5):2130–2141, 2013.

[29] Khader Shameer, Kipp Johnson, Benjamin S Glicksberg, Rachel Hodos, Ben Readhead, Max S Tomlinson, and Joel T Dudley. Prioritizing small molecule as candidates for drug repositioning using machine learning. *bioRxiv*, page 331975, 2018.

[30] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[31] Jochen Sieg, Florian Flachsenberg, and Matthias Rarey. In need of bias control: Evaluating chemical data for machine learning in structure-based virtual screening. *Journal of chemical information and modeling*, 2019.

[32] Ann E Cleves and Ajay N Jain. Effects of inductive bias on computational evaluations of ligand-based modeling and on drug discovery. *Journal of computer-aided molecular design*, 22(3-4):147–159, 2008.

[33] Bernhard Schölkopf, John C Platt, John Shawe-Taylor, Alex J Smola, and Robert C Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 13(7):1443–1471, 2001.

[34] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *the Journal of machine Learning research*, 2:139–154, 2002.

[35] Yanshan Xiao, Bing Liu, Jie Yin, Longbing Cao, Chengqi Zhang, and Zhifeng Hao. Similarity-based approach for positive and unlabeled learning. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1577, 2011.

[36] Nobuyoshi Nagamine and Yasubumi Sakakibara. Statistical prediction of protein–chemical interactions based on chemical structure and mass spectrometry data. *Bioinformatics*, 23(15):2004–2012, 2007.

[37] Rehan Akbani, Stephen Kwek, and Nathalie Japkowicz. Applying support vector machines to imbalanced datasets. In *Machine learning: ECML 2004*, pages 39–50. Springer, 2004.

[38] Konstantinos Veropoulos, Colin Campbell, Nello Cristianini, et al. Controlling the sensitivity of support vector machines. In *Proceedings of the international joint conference on AI*, pages 55–60, 1999.

[39] Ajay N Jain and Ann E Cleves. Does your model weigh the same as a duck? *Journal of computer-aided molecular design*, 26(1):57–67, 2012.

[40] Tip W Loo, M Claire Bartlett, and David M Clarke. Simultaneous binding of two different drugs in the binding pocket of the human multidrug resistance p-glycoprotein. *Journal of Biological Chemistry*, 278(41):39706–39710, 2003.

[41] Yoshihiro Yamanishi, Edouard Pauwels, Hiroto Saigo, and Véronique Stoven. Identification of chemogenomic features from drug-target interaction networks by sparse canonical correspondence analysis. *Machine Learning in Systems Biology*, page 87, 2011.

[42] Yuhao Wang and Jianyang Zeng. Predicting drug-target interactions using restricted boltzmann machines. *Bioinformatics*, 29(13):i126–i134, 2013.

[43] Stefan Günther, Michael Kuhn, Mathias Dunkel, Monica Campillos, Christian Senger, Evangelia Petsalaki, Jessica Ahmed, Eduardo Garcia Urdiales, Andreas Gewiess, Lars Juhl Jensen, et al. Supertarget and matador: resources for exploring drug-target relationships. *Nucleic acids research*, 36(suppl 1):D919–D922, 2008.

[44] Michael R Browning, Bradley T Calhoun, and S Joshua Swamidass. Managing missing measurements in small-molecule screens. *Journal of computer-aided molecular design*, 27(5):469–478, 2013.

[45] Minoru Asogawa, Tsutomu Osoda, Yukiko Fujiwara, and Yoshiko Yamashita. Efficient drug screening using active learning. *NEC J. Adv. Technol*, 2(2):145–148, 2005.

[46] M İhsan Ecemiş, James Wikel, Christopher Bingham, and Eric Bonabeau. A drug candidate design environment using evolutionary computation. *Evolutionary Computation, IEEE Transactions on*, 12(5):591–603, 2008.

[47] Yukiko Fujiwara, Yoshiko Yamashita, Tsutomu Osoda, Minoru Asogawa, Chiaki Fukushima, Masaaki Asao, Hideshi Shimadzu, Kazuya Nakao, and Ryo Shimizu. Virtual screening system for finding structurally diverse hits by active learning. *Journal of chemical information and modeling*, 48(4):930–940, 2008.

[48] Daniel Reker and Gisbert Schneider. Active-learning strategies in computer-assisted drug discovery. *Drug discovery today*, 20(4):458–465, 2015.

[49] Thomas Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1):49–58, 2003.

[50] Benoit Playe, Chloe-Agathe Azencott, and Veronique Stoven. Efficient multi-task chemogenomics for drug specificity prediction. *bioRxiv*, page 193391, 2017.

[51] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[52] James F Blake. Chemoinformatics–predicting the physicochemical properties of 'drug-like' molecules. *Current Opinion in Biotechnology*, 11(1):104–107, 2000.

[53] Svava Ósk Jónsdóttir, Flemming Steen Jørgensen, and Søren Brunak. Prediction methods and databases within chemoinformatics: emphasis on drugs and drug candidates. *Bioinformatics*, 21(10):2145–2160, 2005.

[54] Gert RG Lanckriet, Tijl De Bie, Nello Cristianini, Michael I Jordan, and William Stafford Noble. A statistical framework for genomic data fusion. *Bioinformatics*, 20(16):2626–2635, 2004.

[55] Paul D Dobson and Andrew J Doig. Predicting enzyme class from protein structure without alignments. *Journal of molecular biology*, 345(1):187–199, 2005.

[56] Hendrik Timmerman, Roberto Todeschini, Viviana Consonni, Raimund Mannhold, and Hugo Kubinyi. Handbook of molecular descriptors, 2002.

[57] David Rogers and Mathew Hahn. Extended-connectivity fingerprints. *Journal of chemical information and modeling*, 50(5):742–754, 2010.

[58] Ahmet Sureyya Rifaioglu, Heval Atas, Maria Jesus Martin, Rengul Cetin-Atalay, Volkan Atalay, and Tunca Doğan. Recent applications of deep learning and machine intelligence on in silico drug discovery: methods, tools and databases. *Briefings in bioinformatics*, 2018.

[59] Roberto Todeschini and Viviana Consonni. *Molecular descriptors for chemoinformatics: volume I: alphabetical listing/volume II: appendices, references*, volume 41. John Wiley & Sons, 2009.

[60] Adrià Cereto-Massagué, María José Ojeda, Cristina Valls, Miquel Mulero, Santiago Garcia-Vallvé, and Gerard Pujadas. Molecular fingerprint similarity search in virtual screening. *Methods*, 71:58–63, 2015.

[61] David J Wood, Jacob de Vlieg, Markus Wagener, and Tina Ritschel. Pharmacophore fingerprint-based approach to binding site subpocket similarity and its application to bioisostere replacement. *Journal of chemical information and modeling*, 52(8):2031–2043, 2012.

[62] David Weininger. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *Journal of chemical information and computer sciences*, 28(1):31–36, 1988.

[63] Stephen R Heller, Alan McNaught, Igor Pletnev, Stephen Stein, and Dmitrii Tchekhovskoi. Inchi, the iupac international chemical identifier. *Journal of cheminformatics*, 7(1):23, 2015.

[64] Jianxin Duan, M Sastry, Steven L Dixon, Jeffrey F Lowrie, and Woody Sherman. Analysis and comparison of 2d fingerprints: insights into database screening performance using eight fingerprint methods. *Journal of cheminformatics*, 3(S1):P1, 2011.

[65] Andreas Bender, Jeremy L Jenkins, Josef Scheiber, Sai Chetan K Sukuru, Meir Glick, and John W Davies. How similar are similarity searching methods? a principal component analysis of molecular descriptor space. *Journal of chemical information and modeling*, 49(1):108–119, 2009.

[66] Domenico Alberga, Daniela Trisciuzzi, Michele Montaruli, Francesco Leonetti, Giuseppe Felice Mangiatordi, and Orazio Nicolotti. A new approach for drug target and bioactivity prediction: The multi-fingerprint similarity search algorithm (mussel). *Journal of chemical information and modeling*, 2018.

[67] Sereina Riniker and Gregory A Landrum. Open-source platform to benchmark fingerprints for ligand-based virtual screening. *Journal of cheminformatics*, 5(1):26, 2013.

[68] Ze-Rong Li, Hong Huang Lin, LY Han, L Jiang, X Chen, and Yu Zong Chen. Profeat: a web server for computing structural and physicochemical features of proteins and peptides from amino acid sequence. *Nucleic Acids Research*, 34(suppl 2):W32–W37, 2006.

[69] Serene AK Ong, Hong Huang Lin, Yu Zong Chen, Ze Rong Li, and Zhiwei Cao. Efficacy of different protein descriptors in predicting protein functional families. *Bmc Bioinformatics*, 8(1):300, 2007.

[70] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. *The Journal of Machine Learning Research*, 2:419–444, 2002.

[71] Christina S Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific symposium on biocomputing*, volume 7, pages 566–575, 2002.

[72] Eleazar Eskin, Jason Weston, William S Noble, and Christina S Leslie. Mismatch string kernels for svm protein classification. In *Advances in neural information processing systems*, pages 1417–1424, 2002.

[73] Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Journal of bioinformatics and computational biology*, 3(03):527–550, 2005.

[74] Meng Wang, Jie Yang, Guo-Ping Liu, Zhi-Jie Xu, and Kuo-Chen Chou. Weighted-support vector machines for predicting membrane protein types based on pseudo-amino acid composition. *Protein Engineering Design and Selection*, 17(6):509–516, 2004.

[75] Shao-Wu Zhang, Quan Pan, Hong-Cai Zhang, Yun-Long Zhang, and Hai-Yu Wang. Classification of protein quaternary structure with support vector machine. *Bioinformatics*, 19(18):2390–2396, 2003.

[76] Temple F Smith and Michael S Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147(1):195–197, 1981.

[77] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[78] William R Pearson. [5] rapid and sensitive sequence comparison with fastp and fasta. *Methods in enzymology*, 183:63–98, 1990.

[79] Li Liao and William Stafford Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *Proceedings of the sixth annual international conference on Computational biology*, pages 225–232. ACM, 2002.

[80] Hiroto Saigo, Jean-Philippe Vert, Nobuhisa Ueda, and Tatsuya Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

[81] Michael J Keiser, Bryan L Roth, Blaine N Armbruster, Paul Ernsberger, John J Irwin, and Brian K Shoichet. Relating protein pharmacology by ligand chemistry. *Nature biotechnology*, 25(2):197–206, 2007.

[82] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl 1):i47–i56, 2005.

[83] Brice Hoffmann, Mikhail Zaslavskiy, Jean-Philippe Vert, and Véronique Stoven. A new protein binding pocket similarity measure based on comparison of clouds of atoms in 3d: application to ligand prediction. *BMC bioinformatics*, 11(1):1, 2010.

[84] Risi Kondor, Nino Shervashidze, and Karsten M Borgwardt. The graphlet spectrum. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 529–536. ACM, 2009.

[85] Darren R Flower. On the properties of bit string-based measures of chemical similarity. *Journal of Chemical Information and Computer Sciences*, 38(3):379–386, 1998.

[86] Masahiro Hattori, Yasushi Okuno, Susumu Goto, and Minoru Kanehisa. Development of a chemical structure comparison method for integrated analysis of chemical and genomic information in the metabolic pathways. *Journal of the American Chemical Society*, 125(39):11853–11865, 2003.

[87] S Joshua Swamidass, Jonathan Chen, Jocelyne Bruand, Peter Phung, Liva Ralaivola, and Pierre Baldi. Kernels for small molecules and the prediction of mutagenicity, toxicity and anti-cancer activity. *Bioinformatics*, 21(suppl 1):i359–i368, 2005.

[88] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *ICML*, volume 3, pages 321–328, 2003.

[89] Thomas Gärtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Learning Theory and Kernel Machines*, pages 129–143. Springer, 2003.

[90] Jan Ramon and Thomas Gärtner. Expressivity versus efficiency of graph kernels. In *First international workshop on mining graphs, trees and sequences*, pages 65–74. Citeseer, 2003.

[91] Pierre Mahé, Nobuhisa Ueda, Tatsuya Akutsu, Jean-Luc Perret, and Jean-Philippe Vert. Graph kernels for molecular structure-activity relationship analysis with support vector machines. *Journal of chemical information and modeling*, 45(4):939–951, 2005.

[92] Tamás Horváth. Cyclic pattern kernels revisited. In *Advances in knowledge discovery and data mining*, pages 791–801. Springer, 2005.

[93] Pierre Mahé and Jean-Philippe Vert. Graph kernels based on tree patterns for molecules. *Machine learning*, 75(1):3–35, 2009.

[94] Pierre Mahé, Liva Ralaivola, Véronique Stoven, and Jean-Philippe Vert. The pharmacophore kernel for virtual screening with support vector machines. *Journal of Chemical Information and Modeling*, 46(5):2003–2014, 2006.

[95] Laurent Jacob, Brice Hoffmann, Véronique Stoven, and Jean-Philippe Vert. Virtual screening of gpcrs: an in silico chemogenomics approach. *BMC bioinformatics*, 9(1):363, 2008.

[96] Liva Ralaivola, Sanjay J Swamidass, Hiroto Saigo, and Pierre Baldi. Graph kernels for chemical informatics. *Neural Networks*, 18(8):1093–1110, 2005.

[97] Chloé-Agathe Azencott, Alexandre Ksikes, S Joshua Swamidass, Jonathan H Chen, Liva Ralaivola, and Pierre Baldi. One-to four-dimensional kernels for virtual screening and the prediction of physical, chemical, and biological properties. *Journal of chemical information and modeling*, 47(3):965–974, 2007.

[98] Karsten M Borgwardt and Hans-Peter Kriegel. Shortest-path kernels on graphs. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[99] Tamás Horváth, Thomas Gärtner, and Stefan Wrobel. Cyclic pattern kernels for predictive graph mining. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 158–167. ACM, 2004.

[100] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *The Journal of Machine Learning Research*, 12:2539–2561, 2011.

[101] Giannis Nikolentzos and Michalis Vazirgiannis. Message passing graph kernels. *arXiv preprint arXiv:1808.02510*, 2018.

[102] Kristian Kersting, Martin Mladenov, Roman Garnett, and Martin Grohe. Power iterated color refinement. In *AAAI*, pages 1904–1910, 2014.

[103] Bin Li, Xingquan Zhu, Lianhua Chi, and Chengqi Zhang. Nested subtree hash kernels for large-scale graph classification over streams. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 399–408. IEEE, 2012.

[104] Christopher Morris, Nils M Kriege, Kristian Kersting, and Petra Mutzel. Faster kernels for graphs with continuous attributes via hashing. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1095–1100. IEEE, 2016.

[105] Vikraman Arvind, Johannes Köbler, Gaurav Rattan, and Oleg Verbitsky. On the power of color refinement. In *International Symposium on Fundamentals of Computation Theory*, pages 339–350. Springer, 2015.

[106] Christopher Morris, Kristian Kersting, and Petra Mutzel. Glocalized weisfeiler-lehman graph kernels: Global-local feature maps of graphs. In *Data Mining (ICDM), 2017 IEEE International Conference on*, pages 327–336. IEEE, 2017.

[107] Yoshiyuki Hizukuri, Ryusuke Sawada, and Yoshihiro Yamanishi. Predicting target proteins for drug candidate compounds based on drug-induced gene expression data in a chemical structure-independent manner. *BMC medical genomics*, 8(1):1, 2015.

[108] Sayaka Mizutani, Edouard Pauwels, Véronique Stoven, Susumu Goto, and Yoshihiro Yamanishi. Relating drug–protein interaction network with drug side effects. *Bioinformatics*, 28(18):i522–i528, 2012.

[109] Masataka Takarabe, Masaaki Kotera, Yosuke Nishimura, Susumu Goto, and Yoshihiro Yamanishi. Drug target prediction using adverse event report systems: a pharmacogenomic approach. *Bioinformatics*, 28(18):i611–i618, 2012.

[110] Edouard Pauwels, Véronique Stoven, and Yoshihiro Yamanishi. Predicting drug side-effect profiles: a chemical fragment-based approach. *BMC bioinformatics*, 12(1):169, 2011.

[111] Justin Lamb, Emily D Crawford, David Peck, Joshua W Modell, Irene C Blat, Matthew J Wrobel, Jim Lerner, Jean-Philippe Brunet, Aravind Subramanian, Kenneth N Ross, et al. The connectivity map: using gene-expression signatures to connect small molecules, genes, and disease. *science*, 313(5795):1929–1935, 2006.

[112] Francesco Napolitano, Yan Zhao, Vânia M Moreira, Roberto Tagliaferri, Juha Kere, Mauro D'Amato, and Dario Greco. Drug repositioning: a machine-learning approach through data integration. *J. Cheminformatics*, 5:30, 2013.

[113] Shiwen Zhao and Shao Li. Network-based relating pharmacological and genomic spaces for drug target identification. *PloS one*, 5(7):e11764, 2010.

[114] Twan van Laarhoven, Sander B Nabuurs, and Elena Marchiori. Gaussian interaction profile kernels for predicting drug–target interaction. *Bioinformatics*, 27(21):3036–3043, 2011.

[115] Twan van Laarhoven and Elena Marchiori. Predicting drug-target interactions for new drug compounds using a weighted nearest neighbor profile. *PloS one*, 8(6):e66952, 2013.

[116] Tiqing Liu, Yuhmei Lin, Xin Wen, Robert N Jorissen, and Michael K Gilson. Bindingdb: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic acids research*, 35(suppl_1):D198–D201, 2006.

[117] David S Wishart, Craig Knox, An Chi Guo, Savita Shrivastava, Murtaza Hassanali, Paul Stothard, Zhan Chang, and Jennifer Woolsey. Drugbank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic acids research*, 34(suppl 1):D668–D672, 2006.

[118] Minoru Kanehisa, Susumu Goto, Yoko Sato, Miho Furumichi, and Mao Tanabe. Kegg for integration and interpretation of large-scale molecular data sets. *Nucleic acids research*, page gkr988, 2011.

[119] G Joshi-Tope, Marc Gillespie, Imre Vastrik, Peter D'Eustachio, Esther Schmidt, Bernard de Bono, Bijay Jassal, GR Gopinath, GR Wu, Lisa Matthews, et al. Reactome: a knowledgebase of biological pathways. *Nucleic acids research*, 33(suppl 1):D428–D432, 2005.

[120] Michael Kuhn, Damian Szklarczyk, Andrea Franceschini, Christian Von Mering, Lars Juhl Jensen, and Peer Bork. Stitch 3: zooming in on protein–chemical interactions. *Nucleic acids research*, 40(D1):D876–D880, 2011.

[121] Kirill Degtyarenko, Paula De Matos, Marcus Ennis, Janna Hastings, Martin Zbinden, Alan McNaught, Rafael Alcántara, Michael Darsow, Mickaël Guedj, and Michael Ashburner. Chebi: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, 36(suppl_1):D344–D350, 2007.

[122] John J Irwin and Brian K Shoichet. Zinc- a free database of commercially available compounds for virtual screening. *Journal of chemical information and modeling*, 45(1):177–182, 2005.

[123] Joel L Sussman, Dawei Lin, Jiansheng Jiang, Nancy O Manning, Jaime Prilusky, Otto Ritter, and Enrique E Abola. Protein data bank (pdb): database of three-dimensional structural information of biological macromolecules. *Acta Crystallographica Section D*, 54(6-1):1078–1084, 1998.

[124] Desaphy Jérémy, Bret Guillaume, Rognan Didier, and Kellenberger Esther. sc-pdb: a 3d-database of ligandable binding sites—10 years on. *Nucleic acids research*, page gku928, 2014.

[125] Alex Bateman, Ewan Birney, Richard Durbin, Sean R Eddy, Robert D Finn, and Erik LL Sonnhammer. Pfam 3.1: 1313 multiple alignments and profile hmms match the majority of proteins. *Nucleic acids research*, 27(1):260–262, 1999.

[126] Alexey G Murzin, Steven E Brenner, Tim Hubbard, and Cyrus Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of molecular biology*, 247(4):536–540, 1995.

[127] Sarah Hunter, Rolf Apweiler, Teresa K Attwood, Amos Bairoch, Alex Bateman, David Binns, Peer Bork, Ujjwal Das, Louise Daugherty, Lauranne Duquenne, et al. Interpro: the integrative protein signature database. *Nucleic acids research*, 37(suppl_1):D211–D215, 2008.

[128] Frances MG Pearl, CF Bennett, James E Bray, Andrew P Harrison, Nigel Martin, A Shepherd, Ian Sillitoe, J Thornton, and Christine A Orengo. The cath database: an extended protein family resource for structural and functional genomics. *Nucleic acids research*, 31(1):452–455, 2003.

[129] Yoshihiro Yamanishi, Michihiro Araki, Alex Gutteridge, Wataru Honda, and Minoru Kanehisa. Prediction of drug–target interaction networks from the integration of chemical and genomic spaces. *Bioinformatics*, 24(13):i232–i240, 2008.

[130] Yoshihiro Yamanishi, Masaaki Kotera, Minoru Kanehisa, and Susumu Goto. Drug-target interaction prediction from chemical, genomic and pharmacological data in an integrated framework. *Bioinformatics*, 26(12):i246–i254, 2010.

[131] Kevin Bleakley and Yoshihiro Yamanishi. Supervised prediction of drug–target interactions using bipartite local models. *Bioinformatics*, 25(18):2397–2403, 2009.

[132] Jian-Ping Mei, Chee-Keong Kwoh, Peng Yang, Xiao-Li Li, and Jie Zheng. Drug–target interaction prediction by learning from local information and neighbors. *Bioinformatics*, 29(2):238–245, 2013.

[133] Mehmet Gönen. Predicting drug–target interactions from chemical and genomic kernels using bayesian matrix factorization. *Bioinformatics*, 28(18):2304–2310, 2012.

[134] Mehmet Gonen and Samuel Kaski. Kernelized bayesian matrix factorization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(10):2047–2060, 2014.

[135] Laurent Jacob and Jean-Philippe Vert. Protein-ligand interaction prediction: an improved chemogenomics approach. *Bioinformatics*, 24(19):2149–2156, 2008.

[136] Yong Liu, Min Wu, Chunyan Miao, Peilin Zhao, and Xiao-Li Li. Neighborhood regularized logistic matrix factorization for drug-target interaction prediction. *PLoS computational biology*, 12(2):e1004760, 2016.

[137] Michael M Mysinger, Michael Carchia, John J Irwin, and Brian K Shoichet. Directory of useful decoys, enhanced (dud-e): better ligands and decoys for better benchmarking. *Journal of medicinal chemistry*, 55(14):6582–6594, 2012.

[138] Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

[139] Greg Landrum. Rdkit documentation. *Release*, 1:1–79, 2013.

[140] Jean-Luc Perret and Pierre Mahé. Chemcpp user guide. *Bioinformatics Center and Center for Computational Biology: Kyoto, Japan and Paris, France*, 2006.

[141] Jean-Philippe Vert. The optimal assignment kernel is not positive definite. *arXiv preprint arXiv:0801.4061*, 2008.

[142] Zhen Chen, Pei Zhao, Fuyi Li, André Leier, Tatiana T Marquez-Lago, Yanan Wang, Geoffrey I Webb, A Ian Smith, Roger J Daly, Kuo-Chen Chou, et al. ifeature: a python package and web server for features extraction and selection from protein and peptide sequences. *Bioinformatics*, 34(14):2499–2502, 2018.

[143] Nan Xiao, Dong-Sheng Cao, Min-Feng Zhu, and Qing-Song Xu. protr/protrweb: R package and web server for generating various numerical representation schemes of protein sequences. *Bioinformatics*, 31(11):1857–1859, 2015.

[144] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[145] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[146] François Chollet et al. Keras. `https://keras.io`, 2015.

[147] Gerard JP van Westen, Jörg K Wegner, Adriaan P IJzerman, Herman WT van Vlijmen, and A Bender. Proteochemometric modeling as a tool to design selective compounds and for extrapolating to novel targets. *MedChemComm*, 2(1):16–30, 2011.

[148] Ming Hao, Stephen H Bryant, and Yanli Wang. A new chemoinformatics approach with improved strategies for effective predictions of potential drugs. *Journal of cheminformatics*, 10(1):50, 2018.

[149] Yoan Martinez-Lopez, Yaile Caballero, Stephen J Barigye, Yovani Marrero-Ponce, Reisel Millan-Cabrera, Julio Madera, Francisco Torrens, and Juan A Castillo-Garit. State of the art review and report of new tool for drug discovery. *Current topics in medicinal chemistry*, 17(26):2957–2976, 2017.

[150] Yoshihiro Yamanishi. Inferring chemogenomic features from drug-target interaction networks. *Molecular Informatics*, 32(11-12):991–999, 2013.

[151] Qingjun Yuan, Junning Gao, Dongliang Wu, Shihua Zhang, Hiroshi Mamitsuka, and Shanfeng Zhu. Druge-rank: improving drug–target interaction prediction of new candidate drugs or targets by ensemble learning to rank. *Bioinformatics*, 32(12):i18–i27, 2016.

[152] Zheng Xia, Ling-Yun Wu, Xiaobo Zhou, and Stephen TC Wong. Semi-supervised drug-protein interaction prediction from heterogeneous biological spaces. *BMC systems biology*, 4(Suppl 2):S6, 2010.

[153] Xiaodong Zheng, Hao Ding, Hiroshi Mamitsuka, and Shanfeng Zhu. Collaborative matrix factorization with multiple similarities for predicting drug-target interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1025–1033. ACM, 2013.

[154] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[155] Christopher C Johnson. Logistic matrix factorization for implicit feedback data. *Advances in Neural Information Processing Systems*, 27, 2014.

[156] Murat Can Cobanoglu, Chang Liu, Feizhuo Hu, Zolta 'an N Oltvai, and Ivet Bahar. Predicting drug–target interactions using probabilistic matrix factorization. *Journal of chemical information and modeling*, 53(12):3399–3409, 2013.

[157] Hua Yu, Jianxin Chen, Xue Xu, Yan Li, Huihui Zhao, Yupeng Fang, Xiuxiu Li, Wei Zhou, Wei Wang, and Yonghua Wang. A systematic prediction of multiple drug-target interactions from chemical, genomic, and pharmacological data. *PloS one*, 7(5):e37608, 2012.

[158] Dong-Sheng Cao, Liu-Xia Zhang, Gui-Shan Tan, Zheng Xiang, Wen-Bin Zeng, Qing-Song Xu, and Alex F Chen. Computational prediction of drug–target interactions using chemical, biological, and network features. *Molecular Informatics*, 33(10):669–681, 2014.

[159] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[160] S Joshua Swamidass, Chloé-Agathe Azencott, Ting-Wan Lin, Hugo Gramajo, Shiou-Chuan Tsai, and Pierre Baldi. Influence relevance voting: an accurate and interpretable virtual high throughput screening method. *Journal of chemical information and modeling*, 49(4):756–766, 2009.

[161] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[162] Tong He, Marten Heidemeyer, Fuqiang Ban, Artem Cherkasov, and Martin Ester. Simboost: a read-across approach for predicting drug–target binding affinities using gradient boosting machines. *Journal of cheminformatics*, 9(1):24, 2017.

[163] Ru Zhang. An ensemble learning approach for improving drug–target interactions prediction. In *Proceedings of the 4th International Conference on Computer Engineering and Networks*, pages 433–442. Springer, 2015.

[164] Ming Hao, Yanli Wang, and Stephen H Bryant. Improved prediction of drug-target interactions using regularized least squares integrating with kernel fusion technique. *Analytica chimica acta*, 909:41–50, 2016.

[165] Rawan S Olayan, Haitham Ashoor, and Vladimir B Bajic. Ddr: efficient computational method to predict drug–target interactions using graph mining and machine learning approaches. *Bioinformatics*, 34(7):1164–1173, 2017.

[166] André CA Nascimento, Ricardo BC Prudêncio, and Ivan G Costa. A multiple kernel learning algorithm for drug-target interaction prediction. *BMC bioinformatics*, 17(1):1, 2016.

[167] Anna Cichonska, Tapio Pahikkala, Sandor Szedmak, Heli Julkunen, Antti Airola, Markus Heinonen, Tero Aittokallio, and Juho Rousu. Learning with multiple pairwise kernels for drug bioactivity prediction. *Bioinformatics*, 34(13):i509–i518, 2018.

[168] Tapio Pahikkala, Antti Airola, Sami Pietilä, Sushil Shakyawar, Agnieszka Szwajda, Jing Tang, and Tero Aittokallio. Toward more realistic drug–target interaction predictions. *Briefings in bioinformatics*, page bbu010, 2014.

[169] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[170] Dumitru Erhan, Pierre-Jean L'Heureux, Shi Yi Yue, and Yoshua Bengio. Collaborative filtering on a family of biological targets. *Journal of chemical information and modeling*, 46(2):626–635, 2006.

[171] Jean-Loup Faulon, Milind Misra, Shawn Martin, Ken Sale, and Rajat Sapra. Genome scale enzyme–metabolite and drug–target interaction predictions using the signature molecular descriptor. *Bioinformatics*, 24(2):225–233, 2008.

[172] Tommi Jaakkola, Mark Diekhans, and David Haussler. A discriminative framework for detecting remote protein homologies. *Journal of computational biology*, 7(1-2):95–114, 2000.

[173] Yasushi Okuno, Akiko Tamon, Hiroaki Yabuuchi, Satoshi Niijima, Yohsuke Minowa, Koichiro Tonomura, Ryo Kunimoto, and Chunlai Feng. Glida: Gpcr—ligand database for chemical genomics drug discovery—database and tools update. *Nucleic acids research*, 36(suppl_1):D907–D912, 2007.

[174] Gerard Manning, David B Whyte, Ricardo Martinez, Tony Hunter, and Sucha Sudarsanam. The protein kinase complement of the human genome. *Science*, 298(5600):1912–1934, 2002.

[175] Minoru Kanehisa, Michihiro Araki, Susumu Goto, Masahiro Hattori, Mika Hirakawa, Masumi Itoh, Toshiaki Katayama, Shuichi Kawashima, Shujiro Okuda, Toshiaki Tokimatsu, and Yoshihiro Yamanishi. Kegg for linking genomes to life and the environment. *Nucleic acids research*, 36(suppl_1):D480–D484, 2007.

[176] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[177] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[178] Vijay Raghavan, Peter Bollmann, and Gwang S Jung. A critical investigation of recall and precision as measures of retrieval system performance. *ACM Transactions on Information Systems (TOIS)*, 7(3):205–229, 1989.

[179] Christian Kramer and Peter Gedeck. Leave-cluster-out cross-validation is appropriate for scoring functions derived from diverse protein data sets. *Journal of chemical information and modeling*, 50(11):1961–1969, 2010.

[180] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[181] Manfred K Warmuth, Jun Liao, Gunnar Rätsch, Michael Mathieson, Santosh Putta, and Christian Lemmen. Active learning with support vector machines in the drug discovery process. *Journal of Chemical Information and Computer Sciences*, 43(2):667–673, 2003.

[182] Man Fung, Anna Thornton, Kathy Mybeck, Jasmanda Hsiao-hui Wu, Ken Hornbuckle, and Edmundo Muniz. Evaluation of the characteristics of safety withdrawal of prescription drugs from worldwide pharmaceutical markets-1960 to 1999. *Drug Information Journal*, 35(1):293–317, 2001.

[183] Zaina P Qureshi, Enrique Seoane-Vazquez, Rosa Rodriguez-Monguio, Kurt B Stevenson, and Sheryl L Szeinbach. Market withdrawal of new molecular entities approved in the united states from 1980 to 2009. *Pharmacoepidemiology and drug safety*, 20(7):772–777, 2011.

[184] Fabrıcio A Moreira. Best practice & research clinical endocrinology & metabolism. *Best Practice & Research Clinical Endocrinology & Metabolism*, 23:133–144, 2009.

[185] JM Cotton, MT Kearney, and AM Shah. Nitric oxide and myocardial function in heart failure: friend or foe? *Heart*, 88(6):564–566, 2002.

[186] Andrey Kazakov, Rabea Hall, Philippe Jagoda, Katrin Bachelier, Patrick Müller-Best, Alexander Semenov, Frank Lammert, Michael Böhm, and Ulrich Laufs. Inhibition of endothelial nitric oxide synthase induces and enhances myocardial fibrosis. *Cardiovascular research*, 100(2):211–221, 2013.

[187] Xiao-Dong Zhang, Deborah K Lieu, and Nipavan Chiamvimonvat. Small-conductance ca 2+-activated k+ channels and cardiac arrhythmias. *Heart Rhythm*, 12(8):1845–1851, 2015.

[188] Toshinori Aoyagi and Takashi Matsui. Phosphoinositide-3 kinase signaling in cardiac hypertrophy and heart failure. *Current pharmaceutical design*, 17(18):1818–1824, 2011.

[189] Bolette Christiansen, Anne-Kristine Meinild, Anders A Jensen, and Hans Braüner-Osborne. Cloning and characterization of a functional human γ-aminobutyric acid (gaba) transporter, human gat-2. *Journal of Biological Chemistry*, 282(27):19331–19341, 2007.

[190] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8599–8603. IEEE, 2013.

[191] Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular Systems Biology*, 12(7):878, 2016.

[192] Seonwoo Min, Byunghan Lee, and Sungroh Yoon. Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5):851–869, 2017.

[193] Philip M Farrell, Beryl J Rosenstein, Terry B White, Frank J Accurso, Carlo Castellani, Garry R Cutting, Peter R Durie, Vicky A LeGrys, John Massie, Richard B Parad, et al. Guidelines for diagnosis of cystic fibrosis in newborns through older adults: Cystic fibrosis foundation consensus report. *The Journal of pediatrics*, 153(2):S4–S14, 2008.

[194] Johanna M Rommens, Michael C Iannuzzi, Bat-sheva Kerem, Mitchell L Drumm, Georg Melmer, Michhael Dean, Richard Rozmahel, Jeffery L Cole, Dara Kennedy, Noriko Hidaka, et al. Identification of the cystic fibrosis gene: chromosome walking and jumping. *Science*, 245(4922):1059–1065, 1989.

[195] Christopher F Higgins. Abc transporters: from microorganisms to man. *Annual review of cell biology*, 8(1):67–113, 1992.

[196] Ann EO Trezíse and Manuel Buchwald. In vivo cell-specific expression of the cystic fibrosis transmembrane conductance regulator. *Nature*, 353(6343):434, 1991.

[197] Paul Linsdell and John W Hanrahan. Glutathione permeability of cftr. *American Journal of Physiology-Cell Physiology*, 275(1):C323–C326, 1998.

[198] Patryk Moskwa, Daniel Lorentzen, Katherine JDA Excoffon, Joseph Zabner, Paul B McCray Jr, William M Nauseef, Corinne Dupuy, and Botond Bánfi. A novel host defense system of airways is defective in cystic fibrosis. *American journal of respiratory and critical care medicine*, 175(2):174–183, 2007.

[199] David A Stoltz, David K Meyerholz, and Michael J Welsh. Origins of cystic fibrosis lung disease. *New England Journal of Medicine*, 372(4):351–362, 2015.

[200] Francesco Galli, Andrea Battistoni, Roberto Gambari, Alfonso Pompella, Alessandra Bragonzi, Francesca Pilolli, Luigi Iuliano, Marta Piroddi, Maria Cristina Dechecchi, and Giulio Cabrini. Oxidative stress and antioxidant therapy in cystic fibrosis. *Biochimica et Biophysica Acta (BBA)-Molecular Basis of Disease*, 1822(5):690–713, 2012.

[201] JS Elborn. April 2016. cystic fibrosis. *Lancet http://dx. doi. org/10.1016/S0140-6736 (16)*, pages 00576–6, 29.

[202] Y Alahdab Ozen and Deniz G Duman. Pancreatic involvement in cystic fibrosis. *Minerva medica*, 107(6):427–436, 2016.

[203] Natasha Kamal, Pallavi Surana, and Christopher Koh. Liver disease in patients with cystic fibrosis. *Current opinion in gastroenterology*, 34(3):146–151, 2018.

[204] Tracey L Bonfield, Michael W Konstan, and Melvin Berger. Altered respiratory epithelial cell cytokine production in cystic fibrosis. *Journal of Allergy and Clinical Immunology*, 104(1):72–78, 1999.

[205] Xian Xia, Jie Wang, Yuan Liu, and Ming Yue. Lower cystic fibrosis transmembrane conductance regulator (cftr) promotes the proliferation and migration of endometrial carcinoma. *Medical science monitor: international medical journal of experimental and clinical research*, 23:966, 2017.

[206] Xiao Zhong, Hong-qi Chen, Xiu-ling Yang, Qing Wang, Wenliang Chen, and Chunfu Li. Cftr activation suppresses glioblastoma cell proliferation, migration and invasion. *Biochemical and biophysical research communications*, 508(4):1279–1285, 2019.

[207] Manish Bodas and Neeraj Vij. Adapting proteostasis and autophagy for controlling the pathogenesis of cystic fibrosis lung disease. *Frontiers in pharmacology*, 10, 2019.

[208] Varrie Ogilvie, Margaret Passmore, Laura Hyndman, Lisa Jones, Barbara Stevenson, Abigail Wilson, Heather Davidson, Robert R Kitchen, Robert D Gray, Pallav Shah, et al. Differential global gene expression in cystic fibrosis nasal and bronchial epithelium. *Genomics*, 98(5):327–336, 2011.

[209] Todd MacKenzie, Alex H Gifford, Kathryn A Sabadosa, Hebe B Quinton, Emily A Knapp, Christopher H Goss, and Bruce C Marshall. Longevity of patients with cystic fibrosis in 2000 to 2010 and beyond: survival analysis of the cystic fibrosis foundation patient registry. *Annals of internal medicine*, 161(4):233–241, 2014.

[210] Xin Meng, Jack Clews, Vasileios Kargas, Xiaomeng Wang, and Robert C Ford. The cystic fibrosis transmembrane conductance regulator (cftr) and its stability. *Cellular and Molecular Life Sciences*, 74(1):23–38, 2017.

[211] Anjaparavanda P Naren, Bryan Cobb, Chunying Li, Koushik Roy, David Nelson, Ghanshyam D Heda, Jie Liao, Kevin L Kirk, Eric J Sorscher, John Hanrahan, et al. A macromolecular complex of β2 adrenergic receptor, cftr, and ezrin/radixin/moesin-binding phosphoprotein 50 is regulated by pka. *Proceedings of the National Academy of Sciences*, 100(1):342–346, 2003.

[212] Paola Vergani, Steve W Lockless, Angus C Nairn, and David C Gadsby. Cftr channel opening by atp-driven tight dimerization of its nucleotide-binding domains. *Nature*, 433(7028):876, 2005.

[213] Fernando Augusto Lima Marson, Carmen Sílvia Bertuzzo, and José Dirceu Ribeiro. Classification of cftr mutation classes. *The Lancet Respiratory Medicine*, 4(8):e37–e38, 2016.

[214] Manon Ruffin, Lucie Roussel, Émilie Maillé, Simon Rousseau, and Emmanuelle Brochiero. Vx-809/vx-770 treatment reduces inflammatory response to pseudomonas aeruginosa in primary differentiated cystic fibrosis bronchial epithelial cells. *American Journal of Physiology-Lung Cellular and Molecular Physiology*, 2017.

[215] Steven M Rowe, Sonya L Heltshe, Tanja Gonska, Scott H Donaldson, Drucy Borowitz, Daniel Gelfond, Scott D Sagel, Umer Khan, Nicole Mayer-Hamblett, Jill M Van Dalfsen, et al. Clinical mechanism of the cystic fibrosis transmembrane conductance regulator potentiator ivacaftor in g551d-mediated cystic fibrosis. *American journal of respiratory and critical care medicine*, 190(2):175–184, 2014.

[216] Hong Yu Ren, Diane E Grove, Oxana De La Rosa, Scott A Houck, Pattarawut Sopha, Fredrick Van Goor, Beth J Hoffman, and Douglas M Cyr. Vx-809 corrects folding defects in cystic fibrosis transmembrane conductance regulator protein through action on membrane-spanning domain 1. *Molecular biology of the cell*, 24(19):3016–3024, 2013.

[217] Fredrick Van Goor, Sabine Hadida, Peter DJ Grootenhuis, Bill Burton, Jeffrey H Stack, Kimberly S Straley, Caroline J Decker, Mark Miller, Jason McCartney, Eric R Olson, et al. Correction of the

f508del-cftr protein processing defect in vitro by the investigational drug vx-809. *Proceedings of the National Academy of Sciences*, 108(46):18843–18848, 2011.

[218] Paul DW Eckford, Canhui Li, Mohabir Ramjeesingh, and Christine E Bear. Cystic fibrosis transmembrane conductance regulator (cftr) potentiator vx-770 (ivacaftor) opens the defective channel gate of mutant cftr in a phosphorylation-dependent but atp-independent manner. *Journal of Biological Chemistry*, 287(44):36639–36649, 2012.

[219] Shuzhong Zhang, Chandra L Shrestha, and Benjamin T Kopp. Cystic fibrosis transmembrane conductance regulator (cftr) modulators have differential effects on cystic fibrosis macrophage function. *Scientific reports*, 8(1):17066, 2018.

[220] Roxanna Barnaby, Katja Koeppen, Amanda Nymon, Thomas H Hampton, Brent Berwin, Alix Ashare, and Bruce Stanton. Lumacaftor (vx-809) restores the ability of cf macrophages to phagocytose and kill pseudomonas aeruginosa. *American Journal of Physiology-Lung Cellular and Molecular Physiology*, 2017.

[221] Elena K Schneider, Rachel M McQuade, Vincenzo C Carbone, Felisa Reyes-Ortega, John W Wilson, Brenda Button, Ayame Saito, Daniel P Poole, Daniel Hoyer, Jian Li, et al. The potentially beneficial central nervous system activity profile of ivacaftor and its metabolites. *ERJ open research*, 4(1):00127–2017, 2018.

[222] Elena K Schneider. Cytochrome p450 3a4 induction: lumacaftor versus ivacaftor potentially resulting in significantly reduced plasma concentration of ivacaftor. *Drug metabolism letters*, 12(1):71–74, 2018.

[223] Vamshi K Manda, Bharathi Avula, Olivia R Dale, Zulfiqar Ali, Ikhlas A Khan, Larry A Walker, and Shabana I Khan. Pxr mediated induction of cyp3a4, cyp1a2, and p-gp by mitragyna speciosa and its alkaloids. *Phytotherapy research*, 31(12):1935–1945, 2017.

[224] Brittany A Bruch, Sachinkumar B Singh, Laura J Ramsey, and Timothy D Starner. Impact of a cystic fibrosis transmembrane conductance regulator (cftr) modulator on high-dose ibuprofen therapy in pediatric cystic fibrosis patients. *Pediatric pulmonology*, 53(8):1035–1039, 2018.

[225] Jennifer S Guimbellot, Edward P Acosta, and Steven M Rowe. Sensitivity of ivacaftor to drug-drug interactions with rifampin, a cytochrome p450 3a4 inducer. *Pediatric pulmonology*, 53(5):E6–E8, 2018.

[226] Galina Shmarina, Alexander Pukhalsky, Nika Petrova, Ekaterina Zakharova, Lucine Avakian, Nikolai Kapranov, and Vladimir Alioshkin. Tnf gene polymorphisms in cystic fibrosis patients: contribution to the disease progression. *Journal of translational medicine*, 11(1):19, 2013.

[227] Jérémy Rocca, Sylvie Manin, Anne Hulin, Abdel Aissat, Wilfried Verbecq-Morlot, Virginie Prulière-Escabasse, Adeline Wohlhuter-Haddad, Ralph Epaud, Pascale Fanen, and Agathe Tarze. New use for an old drug: Cox-independent anti-inflammatory effects of sulindac in models of cystic fibrosis. *British journal of pharmacology*, 173(11):1728–1741, 2016.

[228] Nicola J Ronan, Gisli G Einarsson, Maria Twomey, Denver Mooney, David Mullane, Muireann NiChroinin, Grace O'Callaghan, Fergus Shanahan, Desmond M Murphy, Owen J O'Connor, et al. Cork study in cystic fibrosis: sustained improvements in ultra-low-dose chest ct scores after cftr modulation with ivacaftor. *Chest*, 153(2):395–403, 2018.

[229] Daniela De Stefano, Valeria R Villella, Speranza Esposito, Antonella Tosco, Angela Sepe, Fabiola De Gregorio, Laura Salvadori, Rosa Grassia, Carlo A Leone, Giuseppe De Rosa, et al. Restoration of cftr function in patients with cystic fibrosis carrying the f508del-cftr mutation. *Autophagy*, 10(11):2053–2074, 2014.

[230] Molly M He, Annemarie Stroustrup Smith, Johan D Oslob, William M Flanagan, Andrew C Braisted, Adrian Whitty, Mark T Cancilla, Jun Wang, Alexey A Lugovskoy, Josh C Yoburn, et al. Small-molecule inhibition of tnf-α. *Science*, 310(5750):1022–1025, 2005.

[231] Elizabeth R Peitzman, Nathan A Zaidman, Peter J Maniak, and Scott M O'Grady. Carvedilol binding to β2-adrenergic receptors inhibits cftr-dependent anion secretion in airway epithelial cells. *American Journal of Physiology-Lung Cellular and Molecular Physiology*, 310(1):L50–L58, 2015.

[232] Jeeyeon Kim, Miesha Farahmand, Colleen Dunn, Carlos E Milla, Rina I Horii, Ewart AC Thomas, Richard B Moss, and Jeffrey J Wine. Sweat rate analysis of ivacaftor potentiation of cftr in non-cf adults. *Scientific reports*, 8(1):16233, 2018.

[233] John J Brewington, Jessica Backstrom, Amanda Feldman, Elizabeth L Kramer, Jessica D Moncivaiz, Alicia J Ostmann, Xiaoting Zhu, L Jason Lu, and John P Clancy. Chronic β2ar stimulation limits cftr activation in human airway epithelia. *JCI insight*, 3(4), 2018.

[234] Paul DW Eckford, Canhui Li, Mohabir Ramjeesingh, and Christine E Bear. Cftr potentiator vx-770 (ivacaftor) opens the defective channel gate of mutant cftr in a phosphorylation-dependent but atp-independent manner. *Journal of Biological Chemistry*, pages jbc–M112, 2012.

[235] Carlos M Farinha, Agnieszka Swiatecka-Urban, David L Brautigan, and Peter Jordan. Regulatory crosstalk by protein kinases on cftr trafficking and activity. *Frontiers in chemistry*, 4:1, 2016.

[236] Romina Fiorotto, Mariangela Amenduni, Valeria Mariotti, Luca Fabris, Carlo Spirli, and Mario Strazzabosco. Src kinase inhibition reduces inflammatory and cytoskeletal changes in δf508 human cholangiocytes and improves cystic fibrosis transmembrane conductance regulator correctors efficacy. *Hepatology*, 67(3):972–988, 2018.

[237] Yang Fei, Liqin Sun, Chungang Yuan, Min Jiang, Qinhua Lou, and Yan Xu. Cftr ameliorates high glucose-induced oxidative stress and inflammation by mediating the nf-κb and mapk signaling pathways in endothelial cells. *International journal of molecular medicine*, 41(6):3501–3508, 2018.

[238] Agata M Trzcińska-Daneluti, Leo Nguyen, Chong Jiang, Christopher Fladd, David Uehling, Michael Prakesch, Rima Al-awar, and Daniela Rotin. Use of kinase inhibitors to correct δf508-cftr function. *Molecular & Cellular Proteomics*, 11(9):745–757, 2012.

[239] Celltox green cytotoxicity assay. `https://france.promega.com/products/cell-health-assays/cell-viability-and-cytotoxicity-assays/celltox-green-cytotoxicity-assay/?catNum=G8741`. Accessed: 2019-04-20.

[240] Celltiter-glo® luminescent cell viability assay. `https://france.promega.com/products/cell-health-assays/cell-viability-and-cytotoxicity-assays/celltiter_glo-luminescent-cell-viability-assay/?catNum=G7570`. Accessed: 2019-04-20.

[241] Guillermo Prado-Vázquez, Angelo Gámez-Pozo, Lucía Trilla-Fuertes, Jorge M Arevalillo, Andrea Zapater-Moros, María Ferrer-Gómez, Mariana Díaz-Almirón, Rocío López-Vacas, Hilario Navarro, Paloma Maín, et al. A novel approach to triple-negative breast cancer molecular classification reveals a luminal immune-positive subgroup with good prognoses. *Scientific reports*, 9(1):1538, 2019.

[242] Damian Szklarczyk, Andrea Franceschini, Stefan Wyder, Kristoffer Forslund, Davide Heller, Jaime Huerta-Cepas, Milan Simonovic, Alexander Roth, Alberto Santos, Kalliopi P Tsafou, et al. String v10: protein–protein interaction networks, integrated over the tree of life. *Nucleic acids research*, 43(D1):D447–D452, 2014.

[243] Risi Kondor and Jean-Philippe Vert. Diffusion kernels. *kernel methods in computational biology*, pages 171–192, 2004.

[244] Irina S Babina and Nicholas C Turner. Advances and challenges in targeting fgfr signalling in cancer. *Nature Reviews Cancer*, 17(5):318, 2017.

[245] Zoi Karoulia, Evripidis Gavathiotis, and Poulikos I Poulikakos. New perspectives for targeting raf kinase in human cancer. *Nature Reviews Cancer*, 17(11):676, 2017.

[246] Matthew G Vander Heiden and Ralph J DeBerardinis. Understanding the intersections between metabolism and cancer biology. *Cell*, 168(4):657–669, 2017.

[247] Debmalya Roy, Gao Ying Sheng, Semukunzi Herve, Evandro Carvalho, Arpan Mahanty, Shengtao Yuan, and Li Sun. Interplay between cancer cell cycle and metabolism: Challenges, targets and therapeutic opportunities. *Biomedicine & Pharmacotherapy*, 89:288–296, 2017.

[248] Unknown ref.

[249] Eun-Kyoung Breuer, Daniela Fukushiro-Lopes, Annika Dalheim, Miranda Burnette, Jeremiah Zartman, Simon Kaja, Claire Wells, Loredana Campo, Kimberly J Curtis, Ricardo Romero-Moreno, et al. Potassium channel activity controls breast cancer metastasis by affecting β-catenin signaling. *Cell death & disease*, 10(3):180, 2019.

[250] Keith A Dookeran, Wei Zhang, Leslie Stayner, and Maria Argos. Associations of two-pore domain potassium channels and triple negative breast cancer subtype in the cancer genome atlas: systematic evaluation of gene expression and methylation. *BMC research notes*, 10(1):475, 2017.

[251] Alex Panaccione, Yan Guo, Wendell G Yarbrough, and Sergey V Ivanov. Expression profiling of clinical specimens supports the existence of neural progenitor-like stem cells in basal breast cancers. *Clinical breast cancer*, 17(4):298–306, 2017.

[252] Panshi Zhang, Xiaowei Yang, Qian Yin, Jilin Yi, Wenzhuang Shen, Lu Zhao, Zhi Zhu, and Jinwen Liu. Inhibition of sk4 potassium channels suppresses cell proliferation, migration and the epithelial-mesenchymal transition in triple-negative breast cancer cells. *PloS one*, 11(4):e0154471, 2016.

[253] Jordi Barretina, Giordano Caponigro, Nicolas Stransky, Kavitha Venkatesan, Adam A Margolin, Sungjoon Kim, Christopher J Wilson, Joseph Lehár, Gregory V Kryukov, Dmitriy Sonkin, et al. The cancer cell line encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature*, 483(7391):603, 2012.

[254] Mathew J Garnett, Elena J Edelman, Sonja J Heidorn, Chris D Greenman, Anahita Dastur, King Wai Lau, Patricia Greninger, I Richard Thompson, Xi Luo, Jorge Soares, et al. Systematic identification of genomic markers of drug sensitivity in cancer cells. *Nature*, 483(7391):570, 2012.

[255] Benjamin Haibe-Kains, Nehme El-Hachem, Nicolai Juul Birkbak, Andrew C Jin, Andrew H Beck, Hugo JWL Aerts, and John Quackenbush. Inconsistency in large pharmacogenomic studies. *Nature*, 504(7480):389, 2013.

[256] Cancer Cell Line Encyclopedia Consortium, Genomics of Drug Sensitivity in Cancer Consortium, et al. Pharmacogenomic agreement between two cancer cell line data sets. *Nature*, 528(7580):84, 2015.

[257] Nikita Pozdeyev, Minjae Yoo, Ryan Mackie, Rebecca E Schweppe, Aik Choon Tan, and Bryan R Haugen. Integrating heterogeneous drug sensitivity data from cancer pharmacogenomic studies. *Oncotarget*, 7(32):51619, 2016.

[258] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical review E*, 69(6):066138, 2004.

[259] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd, and Dimitry Gorinevsky. An interior-point method for large-scale l1-regularized least squares. *IEEE journal of selected topics in signal processing*, 1(4):606–617, 2007.

[260] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[261] Nicolai Meinshausen and Peter Bühlmann. Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

[262] Mammaprint website. `http://www.mammaprint.fr/`. Accessed: 2019-04-20.

[263] Nanne Aben, Daniel J Vis, Magali Michaut, and Lodewyk Fa Wessels. Tandem: a two-stage approach to maximize interpretability of drug response models based on multiple molecular data types. *Bioinformatics*, 32(17):i413–i420, 2016.

[264] Ladislav Rampasek, Daniel Hidru, Petr Smirnov, Benjamin Haibe-Kains, and Anna Goldenberg. Dr. vae: Drug response variational autoencoder. *arXiv preprint arXiv:1706.08203*, 2017.

[265] Chun-Hao Chang, Ladislav Rampasek, and Anna Goldenberg. Dropout feature ranking for deep learning models. *arXiv preprint arXiv:1712.08645*, 2017.

[266] Marine Le Morvan and Jean-Philippe Vert. Whinter: A working set algorithm for high-dimensional sparse second order interaction models. *arXiv preprint arXiv:1802.05980*, 2018.

[267] Madhu S Dhar and Howard K Plummer. Protein expression of g-protein inwardly rectifying potassium channels (girk) in breast cancer cells. *BMC physiology*, 6(1):8, 2006.

[268] Bradley K Stringer, Amiel G Cooper, and Scott B Shepard. Overexpression of the g-protein inwardly rectifying potassium channel 1 (girk1) in primary breast carcinomas correlates with axillary lymph node metastasis. *Cancer research*, 61(2):582–588, 2001.

[269] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[270] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[271] David T Jones. Protein secondary structure prediction based on position-specific scoring matrices1. *Journal of molecular biology*, 292(2):195–202, 1999.

[272] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287, 2015.

[273] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[274] Sheng Wang, Shunyan Weng, Jianzhu Ma, and Qingming Tang. Deepcnf-d: predicting protein order/disorder regions by weighted deep convolutional neural fields. *International journal of molecular sciences*, 16(8):17315–17330, 2015.

[275] James Lyons, Abdollah Dehzangi, Rhys Heffernan, Alok Sharma, Kuldip Paliwal, Abdul Sattar, Yaoqi Zhou, and Yuedong Yang. Predicting backbone cα angles and dihedrals from protein sequences by stacked sparse auto-encoder deep neural network. *Journal of computational chemistry*, 35(28):2040–2046, 2014.

[276] Søren Kamaric Riis and Anders Krogh. Improving prediction of protein secondary structure using structured neural networks and multiple sequence alignments. *Journal of Computational Biology*, 3(1):163–183, 1996.

[277] Søren Kaae Sønderby and Ole Winther. Protein secondary structure prediction with long short term memory networks. *arXiv preprint arXiv:1412.7828*, 2014.

[278] Michalis Agathocleous, Georgia Christodoulou, Vasilis Promponas, Chris Christodoulou, Vassilis Vassiliades, and Antonis Antoniou. Protein secondary structure prediction with bidirectional recurrent neural nets: Can weight updating for each residue enhance performance? In *IFIP International Conference on Artificial Intelligence Applications and Innovations*, pages 128–137. Springer, 2010.

[279] Vanessa Isabell Jurtz, Alexander Rosenberg Johansen, Morten Nielsen, Jose Juan Almagro Armenteros, Henrik Nielsen, Casper Kaae Sønderby, Ole Winther, and Søren Kaae Sønderby. An introduction to deep learning on biological sequence data: examples and solutions. *Bioinformatics*, 33(22):3685–3690, 2017.

[280] Yanjun Qi, Merja Oja, Jason Weston, and William Stafford Noble. A unified multitask architecture for predicting local protein properties. *PloS one*, 7(3):e32235, 2012.

[281] Fangping Wan and Jianyang Zeng. Deep learning with feature embedding for compound-protein interaction prediction. *bioRxiv*, page 086033, 2016.

[282] Sabrina Jaeger, Simone Fulle, and Samo Turk. Mol2vec: unsupervised machine learning approach with chemical intuition. *Journal of chemical information and modeling*, 58(1):27–35, 2018.

[283] Garrett B Goh, Nathan O Hodas, Charles Siegel, and Abhinav Vishnu. Smiles2vec: An interpretable general-purpose deep neural network for predicting chemical properties. *arXiv preprint arXiv:1712.02034*, 2017.

[284] Xiaoyu Zhang, Sheng Wang, Feiyun Zhu, Zheng Xu, Yuhong Wang, and Junzhou Huang. Seq3seq fingerprint: Towards end-to-end semi-supervised deep drug discovery. In *Proceedings of the 2018 ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 404–413. ACM, 2018.

[285] Talia B Kimber, Sebastian Engelke, Igor V Tetko, Eric Bruno, and Guillaume Godin. Synergy effect between convolutional neural networks and the multiplicity of smiles for improvement of molecular prediction. *arXiv preprint arXiv:1812.04439*, 2018.

[286] Andrew Dalke. Deepsmiles: An adaptation of smiles for use in. *arXiv preprint*, 2018.

[287] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232, 2015.

[288] Sunyoung Kwon and Sungroh Yoon. Deepcci: End-to-end deep learning for chemical-chemical interaction prediction. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 203–212. ACM, 2017.

[289] Zheng Xu, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Seq2seq fingerprint: An unsupervised deep molecular embedding for drug discovery. In *Proceedings of the 8th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 285–294. ACM, 2017.

[290] Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proc. CVPR*, volume 1, page 3, 2017.

[291] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034, 2017.

[292] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[293] Dinh V Tran, Alessandro Sperduti, et al. On filter size in graph convolutional networks. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1534–1541. IEEE, 2018.

[294] Jie Chen, Tengfei Ma, and Cao Xiao. Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*, 2018.

[295] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[296] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *International Conference on Machine Learning*, pages 2702–2711, 2016.

[297] Alessandro Lusci, Gianluca Pollastri, and Pierre Baldi. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *Journal of chemical information and modeling*, 53(7):1563–1575, 2013.

[298] Connor W Coley, Regina Barzilay, William H Green, Tommi S Jaakkola, and Klavs F Jensen. Convolutional embedding of attributed molecular graphs for physical property prediction. *Journal of chemical information and modeling*, 57(8):1757–1772, 2017.

[299] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

[300] Steven Kearnes, Kevin McCloskey, Marc Berndl, Vijay Pande, and Patrick Riley. Molecular graph convolutions: moving beyond fingerprints. *Journal of computer-aided molecular design*, 30(8):595–608, 2016.

[301] Shrey Gadiya, Deepak Anand, and Amit Sethi. Some new layer architectures for graph cnn. *arXiv preprint arXiv:1811.00052*, 2018.

[302] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[303] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[304] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[305] Chao Shang, Qinqing Liu, Ko-Shin Chen, Jiangwen Sun, Jin Lu, Jinfeng Yi, and Jinbo Bi. Edge attention-based multi-relational graph convolutional networks. *arXiv preprint arXiv:1802.04944*, 2018.

[306] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

[307] Han Altae-Tran, Bharath Ramsundar, Aneesh S Pappu, and Vijay Pande. Low data drug discovery with one-shot learning. *ACS central science*, 3(4):283–293, 2017.

[308] Steven Kearnes, Brian Goldman, and Vijay Pande. Modeling industrial admet data with multitask networks. *arXiv preprint arXiv:1606.08793*, 2016.

[309] John Boaz Lee, Ryan A Rossi, Sungchul Kim, Nesreen K Ahmed, and Eunyee Koh. Attention models in graphs: A survey. *arXiv preprint arXiv:1807.07984*, 2018.

[310] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[311] Uday Shankar Shanthamallu, Jayaraman J Thiagarajan, and Andreas Spanias. Improving robustness of attention models on graphs. *arXiv preprint arXiv:1811.00181*, 2018.

[312] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[313] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674. ACM, 2018.

[314] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[315] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[316] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[317] Marco Gori, Gabriele Monfardini, and Franco Scarselli. A new model for learning in graph domains. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 729–734. IEEE, 2005.

[318] Trang Pham, Truyen Tran, Hoa Dam, and Svetha Venkatesh. Graph classification via deep learning with virtual nodes. *arXiv preprint arXiv:1708.04357*, 2017.

[319] Katsuhiko Ishiguro, Shin-ichi Maeda, and Masanori Koyama. Graph warp module: an auxiliary module for boosting the power of graph neural networks. *arXiv preprint arXiv:1902.01020*, 2019.

[320] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning withdifferentiable pooling. *arXiv preprint arXiv:1806.08804*, 2018.

[321] Angelo Porrello, Davide Abati, Simone Calderara, and Rita Cucchiara. Classifying signals on irregular domains via convolutional cluster pooling. *arXiv preprint arXiv:1902.04850*, 2019.

[322] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[323] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. *arXiv preprint arXiv:1806.03536*, 2018.

[324] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[325] Javier Pérez-Sianes, Horacio Pérez-Sánchez, and Fernando Díaz. Virtual screening: a challenge for deep learning. In *10th International Conference on Practical Applications of Computational Biology & Bioinformatics*, pages 13–22. Springer, 2016.

[326] Thomas Unterthiner, Andreas Mayr, Günter Klambauer, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, and Sepp Hochreiter. Deep learning as an opportunity in virtual screening. In *Proceedings of the deep learning workshop at NIPS*, volume 27, pages 1–9, 2014.

[327] Eelke B Lenselink, Niels Ten Dijke, Brandon Bongers, George Papadatos, Herman WT van Vlijmen, Wojtek Kowalczyk, Adriaan P IJzerman, and Gerard JP van Westen. Beyond the hype: deep neural networks outperform established methods using a chembl bioactivity benchmark set. *Journal of cheminformatics*, 9(1):45, 2017.

[328] Masatoshi Hamanaka, Kei Taneishi, Hiroaki Iwata, Jun Ye, Jianguo Pei, Jinlong Hou, and Yasushi Okuno. Cgbvs-dnn: Prediction of compound-protein interactions based on deep learning. *Molecular Informatics*, 2016.

[329] Caihua Wang, Juan Liu, Fei Luo, Yafang Tan, Zixin Deng, and Qian-Nan Hu. Pairwise input neural network for target-ligand interaction prediction. In *Bioinformatics and Biomedicine (BIBM), 2014 IEEE International Conference on*, pages 67–70. IEEE, 2014.

[330] Adam Gonczarek, Jakub M Tomczak, Szymon Zaręba, Joanna Kaczmar, Piotr Dąbrowski, and Michał J Walczak. Learning deep architectures for interaction prediction in structure-based virtual screening. *arXiv preprint arXiv:1610.07187*, 2016.

[331] George E Dahl, Navdeep Jaitly, and Ruslan Salakhutdinov. Multi-task neural networks for qsar predictions. *arXiv preprint arXiv:1406.1231*, 2014.

[332] Junshui Ma, Robert P Sheridan, Andy Liaw, George E Dahl, and Vladimir Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015.

[333] Bharath Ramsundar, Steven Kearnes, Patrick Riley, Dale Webster, David Konerding, and Vijay Pande. Massively multitask networks for drug discovery. *arXiv preprint arXiv:1502.02072*, 2015.

[334] Yuting Xu, Junshui Ma, Andy Liaw, Robert P Sheridan, and Vladimir Svetnik. Demystifying multitask deep neural networks for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 57(10):2490–2504, 2017.

[335] Alexandru Korotcov, Valery Tkachenko, Daniel P Russo, and Sean Ekins. Comparison of deep learning with multiple machine learning methods and metrics using diverse drug discovery data sets. *Molecular pharmaceutics*, 14(12):4462–4475, 2017.

[336] Martin Vogt, Swarit Jasial, and Jurgen Bajorath. Extracting compound profiling matrices from screening data. *ACS Omega*, 3(4):4706–4712, 2018.

[337] Erik Gawehn, Jan A Hiss, and Gisbert Schneider. Deep learning in drug discovery. *Molecular Informatics*, 35(1):3–14, 2016.

[338] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, and Sepp Hochreiter. Deeptox: toxicity prediction using deep learning. *Frontiers in Environmental Science*, 3:80, 2016.

[339] Abdul Karim, Avinash Mishra, MA Hakim Newton, and Abdul Sattar. Efficient toxicity prediction via simple features using shallow neural networks and decision trees. *ACS Omega*, 4(1):1874–1888, 2019.

[340] Alexander Aliper, Sergey Plis, Artem Artemov, Alvaro Ulloa, Polina Mamoshina, and Alex Zhavoronkov. Deep learning applications for predicting pharmacological properties of drugs and drug repurposing using transcriptomic data. *Molecular pharmaceutics*, 2016.

[341] Tyler B Hughes, Grover P Miller, and S Joshua Swamidass. Modeling epoxidation of drug-like molecules with a deep machine learning network. *ACS central science*, 1(4):168–180, 2015.

[342] Raquel Rodríguez-Pérez, Tomoyuki Miyao, Swarit Jasial, Martin Vogt, and Jurgen Bajorath. Prediction of compound profiling matrices using machine learning. *ACS Omega*, 3(4):4713–4723, 2018.

[343] Patrick Hop, Brandon Allgood, and Jessen Yu. Geometric deep learning autonomously learns chemical features that outperform those engineered by domain experts. *Molecular pharmaceutics*, 2018.

[344] Qingyuan Feng, Evgenia Dueva, Artem Cherkasov, and Martin Ester. Padme: A deep learning-based framework for drug-target interaction prediction. *arXiv preprint arXiv:1807.09741*, 2018.

[345] Clyde Fare, Lukas Turcani, and Edward O Pyzer-Knapp. Powerful, transferable representations for molecules through intelligent task selection in deep multitask networks. *arXiv preprint arXiv:1809.06334*, 2018.

[346] Andreas Mayr, Günter Klambauer, Thomas Unterthiner, Marvin Steijaert, Jörg K Wegner, Hugo Ceulemans, Djork-Arné Clevert, and Sepp Hochreiter. Large-scale comparison of machine learning methods for drug target prediction on chembl. *Chemical Science*, 2018.

[347] Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. Deepdta: deep drug–target binding affinity prediction. *Bioinformatics*, 34(17):i821–i829, 2018.

[348] Kyle Yingkai Gao, Achille Fokoue, Heng Luo, Arun Iyengar, Sanjoy Dey, and Ping Zhang. Interpretable drug target prediction using deep neural representation. In *IJCAI*, pages 3371–3377, 2018.

[349] Masashi Tsubaki, Kentaro Tomii, and Jun Sese. Compound-protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics*, 2018.

[350] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[351] Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. *arXiv preprint arXiv:1810.02244*, 2018.

[352] Maya Hirohara, Yutaka Saito, Yuki Koda, Kengo Sato, and Yasubumi Sakakibara. Convolutional neural network based on smiles representation of compounds for detecting chemical motif. *BMC bioinformatics*, 19(19):526, 2018.

[353] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[354] Arindam Paul, Dipendra Jha, Reda Al-Bahrani, Wei-keng Liao, Alok Choudhary, and Ankit Agrawal. Chemixnet: Mixed dnn architectures for predicting chemical properties using multiple molecular representations. *arXiv preprint arXiv:1811.08283*, 2018.

[355] Arindam Paul, Dipendra Jha, Wei-keng Liao, Alok Choudhary, and Ankit Agrawal. Transfer learning using ensemble neural nets for organic solar cell screening. *arXiv preprint arXiv:1903.03178*, 2019.

[356] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014.

[357] Alessandro Tibo, Manfred Jaeger, and Paolo Frasconi. Learning and interpreting multi-multi-instance learning networks. *arXiv preprint arXiv:1810.11514*, 2018.

[358] Bo Jiang, Ziyan Zhang, Jin Tang, and Bin Luo. Multiple graph adversarial learning. *arXiv preprint arXiv:1901.07439*, 2019.

[359] Jingyuan Zhang, Bokai Cao, Sihong Xie, Chun-Ta Lu, Philip S Yu, and Ann B Ragin. Identifying connectivity patterns for brain diseases via multi-side-view guided deep architectures. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 36–44. SIAM, 2016.

[360] Zhiqiang Zhang, Yi Zhao, Xiangke Liao, Wenqiang Shi, Kenli Li, Quan Zou, and Shaoliang Peng. Deep learning in omics: a survey and guideline. *Briefings in functional genomics*, 2018.

[361] Shiliang Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

[362] Darrin P Lewis, Tony Jebara, and William Stafford Noble. Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure. *Bioinformatics*, 22(22):2753–2760, 2006.

[363] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[364] Chloé-Agathe Azencott. Introduction to machine learning for bioinfoinformatics. `http://www.cazencott.info/dotclear/public/lectures/s1133_2018/s1133-2018-intro-ml.pdf`, 2018.

[365] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

[366] Ashirr Kashyap. What does an x axis represent in a logistic regression sigmoid function? https://www.quora.com/What-does-an-x-axis-represent-in-a-logistic-regression-sigmoid-function, 2018.

[367] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA, 2004.

[368] Jean-Philippe Vert. Machine learning with kernel methods. `http://members.cbio.mines-paristech.fr/~jvert/svn/kernelcourse/course/2015mva/index.html`, 2015.

[369] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[370] Rich Caruana and Virginia R De Sa. Promoting poor features to supervisors: Some inputs work better as outputs. In *Advances in Neural Information Processing Systems*, pages 389–395, 1997.

[371] Hao Cheng, Hao Fang, and Mari Ostendorf. Open-domain name error detection using a multi-task rnn. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 737–746, 2015.

[372] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8(Jan):35–63, 2007.

[373] Victor Bellon, Véronique Stoven, and Chloé-Agathe Azencott. Multitask feature selection with task descriptors. In *Biocomputing 2016: Proceedings of the Pacific Symposium*, pages 261–272. World Scientific, 2016.

[374] Laurent Jacob, Jean-philippe Vert, and Francis R Bach. Clustered multi-task learning: A convex formulation. In *Advances in neural information processing systems*, pages 745–752, 2009.

[375] Yu Zhang, Ying Wei, and Qiang Yang. Learning to multitask. *arXiv preprint arXiv:1805.07541*, 2018.

[376] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Learning what to share between loosely related tasks. *arXiv preprint arXiv:1705.08142*, 2017.

[377] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *Advances in neural information processing systems*, pages 41–48, 2007.

[378] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

[379] Aurelie C Lozano and Grzegorz Swirszcz. Multi-level lasso for sparse multi-task regression. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 595–602. Omnipress, 2012.

[380] Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep K Ravikumar. A dirty model for multi-task learning. In *Advances in neural information processing systems*, pages 964–972, 2010.

[381] Ting Kei Pong, Paul Tseng, Shuiwang Ji, and Jieping Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20(6):3465–3489, 2010.

[382] Lei Han and Yu Zhang. Learning tree structure in multi-task learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 397–406. ACM, 2015.

[383] Seyoung Kim and Eric P Xing. Tree-guided group lasso for multitask regression with structured sparsity. 2009, 2010.

[384] Xi Chen, Seyoung Kim, Qihang Lin, Jaime G Carbonell, and Eric P Xing. Graph-structured multi-task regression and an efficient optimization method for general fused lasso. *arXiv preprint arXiv:1005.3579*, 2010.

[385] Guocong Song and Wei Chai. Collaborative learning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 1837–1846, 2018.

[386] TM Heskes. Empirical bayes for learning to learn. *Radboud University Nijmegen*, 2000.

[387] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of the 22nd international conference on Machine learning*, pages 1012–1019. ACM, 2005.

[388] Hal Daumé III. Bayesian multitask learning with latent hierarchies. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 135–142. AUAI Press, 2009.

[389] Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *ICML*, volume 96, pages 489–497, 1996.

[390] Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6(Apr):615–637, 2005.

[391] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[392] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[393] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[394] Yoshua Bengio Ian Goodfellow and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2016.

[395] Adriano A Mol, Luiz S Martins-Filho, José Demisio S da Silva, and Ronilson Rocha. Efficiency parameters estimation in gemstones cut design using artificial neural networks. *Computational materials science*, 38(4):727–736, 2007.

[396] Luke Dormehl. What is an artificial neural network? here's everything you need to know. https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/.

[397] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[398] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[399] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

[400] Christopher olah's blog on deep learning. http://colah.github.io/. Accessed: 2018-10-25.

[401] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[402] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[403] Traitement et analyse d'images. https://perso.esiee.fr/ perretb/I5FM/TAI/index.html. Accessed: 2018-10-25.

[404] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[405] Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in \ *beta*-vae. *arXiv preprint arXiv:1804.03599*, 2018.

[406] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[407] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[408] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[409] Ian Goodfellow. Tutorial: Generative adversarial networks. In *NIPS*, 2016.

[410] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[411] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[412] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE, 2016.

[413] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[414] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. *arXiv preprint arXiv:1704.06904*, 2017.

[415] R Caruna. Multitask learning: A knowledge-based source of inductive bias. In *Machine Learning: Proceedings of the Tenth International Conference*, pages 41–48, 1993.

[416] Long Duong, Trevor Cohn, Steven Bird, and Paul Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 845–850, 2015.

[417] Artem Rozantsev, Mathieu Salzmann, and Pascal Fua. Beyond sharing weights for deep domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[418] Sebastian Ruder12, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Sluice networks: Learning what to share between loosely related tasks. *stat*, 1050:23, 2017.

[419] Liqiang Xiao, Honglun Zhang, Wenqing Chen, Yongkun Wang, and Yaohui Jin. Learning what to share: Leaky multi-task network for text classification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2055–2065, 2018.

[420] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. *arXiv preprint arXiv:1803.10704*, 2018.

[421] Elliot Meyerson and Risto Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. *arXiv preprint arXiv:1711.00108*, 2017.

[422] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[423] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[424] Frans Olaf Zdyb. What methods do you prefer when performing hyperparameter optimization? `https://www.quora.com/What-methods-do-you-prefer-when-performing-hyperparameter-optimization`.

[425] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

[426] James Bergstra and David D Cox. Hyperparameter optimization and boosting for classifying facial expressions: How good can a" null" model be? *arXiv preprint arXiv:1306.3476*, 2013.

[427] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[428] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

[429] Keith T Butler, Daniel W Davies, Hugh Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559(7715):547, 2018.

[430] Pierre Baldi. Deep learning in biomedical data science. *Annual Review of Biomedical Data Science*, 1:181–205, 2018.

[431] Alex Zhavoronkov. Artificial intelligence for drug discovery, biomarker development, and generation of novel chemistry, 2018.

[432] Kristina Preuer, Günter Klambauer, Friedrich Rippmann, Sepp Hochreiter, and Thomas Unterthiner. Interpretable deep learning in drug discovery. *arXiv preprint arXiv:1903.02788*, 2019.

[433] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnn explainer: A tool for post-hoc explanation of graph neural networks. *arXiv preprint arXiv:1903.03894*, 2019.

[434] Isidro Cortés-Ciriano and Andreas Bender. Deep confidence: A computationally efficient framework for calculating reliable prediction errors for deep neural networks. *Journal of chemical information and modeling*, 2018.

[435] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[436] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[437] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[438] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[439] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR, abs/1610.02391*, 7, 2016.

[440] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *arXiv preprint arXiv:1703.01365*, 2017.

[441] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, and Sven Dähne. Patternnet and patternlrp–improving the interpretability of neural networks. *stat*, 1050:16, 2017.

[442] Ryusuke Sawada, Hiroaki Iwata, Sayaka Mizutani, and Yoshihiro Yamanishi. Target-based drug repositioning using large-scale chemical–protein interactome data. *Journal of Chemical Information and Modeling*, 55(12):2717–2730, 2015.

[443] Gert RG Lanckriet, Nello Cristianini, Peter Bartlett, Laurent El Ghaoui, and Michael I Jordan. Learning the kernel matrix with semidefinite programming. *The Journal of Machine Learning Research*, 5:27–72, 2004.

[444] Alain Rakotomamonjy, Francis R Bach, Stéphane Canu, and Yves Grandvalet. Simplemkl. *Journal of Machine Learning Research*, 9(Nov):2491–2521, 2008.

[445] Mehmet Gönen and Ethem Alpaydın. Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12(Jul):2211–2268, 2011.

[446] Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Learning non-linear combinations of kernels. In *Advances in neural information processing systems*, pages 396–404, 2009.

[447] Bo Wang, Aziz M Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. Similarity network fusion for aggregating data types on a genomic scale. *Nature methods*, 11(3):333–337, 2014.

[448] Mehmet Gönen, Suleiman Khan, and Samuel Kaski. Kernelized bayesian matrix factorization. In *International Conference on Machine Learning*, pages 864–872, 2013.

[449] Christian Kramer, Bernd Beck, and Timothy Clark. Insolubility classification with accurate prediction probabilities using a metaclassifier. *Journal of chemical information and modeling*, 50(3):404–414, 2010.

[450] Feixiong Cheng, Yue Yu, Jie Shen, Lei Yang, Weihua Li, Guixia Liu, Philip W Lee, and Yun Tang. Classification of cytochrome p450 inhibitors and noninhibitors using combined classifiers. *Journal of chemical information and modeling*, 51(5):996–1011, 2011.

[451] Darrin P Lewis, Tony Jebara, and William Stafford Noble. Nonstationary kernel combination. In *Proceedings of the 23rd international conference on Machine learning*, pages 553–560. ACM, 2006.

[452] Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900. ACM, 2015.

[453] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1105–1114. ACM, 2016.

[454] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48. ACM, 2013.

[455] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pages 585–591, 2002.

[456] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[457] Gregor Urban, Niranjan Subrahmanya, and Pierre Baldi. Inner and outer recursive neural networks for chemoinformatics applications. *Journal of chemical information and modeling*, 58(2):207–211, 2018.

[458] Garrett B Goh, Charles Siegel, Abhinav Vishnu, Nathan O Hodas, and Nathan Baker. Chemception: a deep neural network with minimal chemistry knowledge matches the performance of expert-developed qsar/qspr models. *arXiv preprint arXiv:1706.06689*, 2017.

[459] Tao Lei, Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Deriving neural architectures from sequence and graph kernels. *arXiv preprint arXiv:1705.09037*, 2017.

[460] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[461] Ryan-Rhys Griffiths. Constrained bayesian optimization for automatic chemical design. *arXiv preprint arXiv:1709.05501*, 2017.

[462] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. *arXiv preprint arXiv:1703.01925*, 2017.

[463] Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

[464] Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: A deep generative model for graphs. *arXiv preprint arXiv:1802.08773*, 2018.

[465] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. *arXiv preprint arXiv:1802.03480*, 2018.

[466] Daniel D Johnson. Learning graphical state transitions. *openreview.net*, 2016.

[467] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

[468] Wengong Jin, Kevin Yang, Regina Barzilay, and Tommi Jaakkola. Learning multimodal graph-to-graph translation for molecular optimization. *arXiv preprint arXiv:1812.01070*, 2018.

[469] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: Benchmarking models for de novo molecular design. *arXiv preprint arXiv:1811.09621*, 2018.

[470] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L Gaunt. Constrained graph variational autoencoders for molecule design. *arXiv preprint arXiv:1805.09076*, 2018.

[471] Jiaxuan You, Bowen Liu, Rex Ying, Vijay Pande, and Jure Leskovec. Graph convolutional policy network for goal-directed molecular graph generation. *arXiv preprint arXiv:1806.02473*, 2018.

[472] Gisbert Schneider and Uli Fechner. Computer-based de novo design of drug-like molecules. *Nature Reviews Drug Discovery*, 4(8):649, 2005.

[473] Chetan Rupakheti, Aaron Virshup, Weitao Yang, and David N Beratan. Strategy to discover diverse optimal molecules in the small molecule universe. *Journal of chemical information and modeling*, 55(3):529–537, 2015.

[474] Artur Kadurin, Alexander Aliper, Andrey Kazennov, Polina Mamoshina, Quentin Vanhaelen, Kuzma Khrabrov, and Alex Zhavoronkov. The cornucopia of meaningful leads: Applying deep adversarial autoencoders for new molecule development in oncology. *Oncotarget*, 8(7):10883, 2017.

[475] Artur Kadurin, Sergey Nikolenko, Kuzma Khrabrov, Alex Aliper, and Alex Zhavoronkov. drugan: an advanced generative adversarial autoencoder model for de novo generation of new molecules with desired molecular properties in silico. *Molecular pharmaceutics*, 14(9):3098–3104, 2017.

[476] Xiufeng Yang, Jinzhe Zhang, Kazuki Yoshizoe, Kei Terayama, and Koji Tsuda. Chemts: an efficient python library for de novo molecular generation. *Science and technology of advanced materials*, 18(1):972–976, 2017.

[477] Marcus Olivecrona, Thomas Blaschke, Ola Engkvist, and Hongming Chen. Molecular de-novo design through deep reinforcement learning. *Journal of cheminformatics*, 9(1):48, 2017.

[478] Mariya Popova, Olexandr Isayev, and Alexander Tropsha. Deep reinforcement learning for de novo drug design. *Science advances*, 4(7):eaap7885, 2018.

[479] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[480] Gabriel Lima Guimaraes, Benjamin Sanchez-Lengeling, Carlos Outeiral, Pedro Luis Cunha Farias, and Alán Aspuru-Guzik. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843*, 2017.

[481] Benjamin Sanchez-Lengeling, Carlos Outeiral, Gabriel L Guimaraes, and Alán Aspuru-Guzik. Optimizing distributions over molecular space. an objective-reinforced generative adversarial network for inverse-design chemistry (organic). *preprint*, 2017.

[482] Evgeny Putin, Arip Asadulaev, Yan Ivanenkov, Vladimir Aladinskiy, Benjamin Sanchez-Lengeling, Alán Aspuru-Guzik, and Alex Zhavoronkov. Reinforced adversarial neural computer for de novo molecular design. *Journal of chemical information and modeling*, 58(6):1194–1204, 2018.

[483] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. *arXiv preprint arXiv:1802.04364*, 2018.

[484] Rim Assouel, Mohamed Ahmed, Marwin H Segler, Amir Saffari, and Yoshua Bengio. Defactor: Differentiable edge factorization-based probabilistic graph generation. *arXiv preprint arXiv:1811.09766*, 2018.

[485] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[486] Marwin HS Segler, Thierry Kogej, Christian Tyrchan, and Mark P Waller. Generating focused molecule libraries for drug discovery with recurrent neural networks. *ACS central science*, 4(1):120–131, 2017.

[487] Daniil Polykovskiy, Alexander Zhebrak, Dmitry Vetrov, Yan Ivanenkov, Vladimir Aladinskiy, Polina Mamoshina, Marine Bozdaganyan, Alexander Aliper, Alex Zhavoronkov, and Artur Kadurin. Entangled conditional adversarial autoencoder for de novo drug discovery. *Molecular pharmaceutics*, 15(10):4398–4405, 2018.

[488] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[489] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[490] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[491] Evgeny Putin, Arip Asadulaev, Quentin Vanhaelen, Yan Ivanenkov, Anastasia V Aladinskaya, Alex Aliper, and Alex Zhavoronkov. Adversarial threshold neural computer for molecular de novo design. *Molecular pharmaceutics*, 15(10):4386–4397, 2018.

[492] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[493] Daniil Polykovskiy, Alexander Zhebrak, Benjamin Sanchez-Lengeling, Sergey Golovanov, Oktai Tatanov, Stanislav Belyaev, Rauf Kurbanov, Aleksey Artamonov, Vladimir Aladinskiy, Mark Veselov, et al. Molecular sets (moses): A benchmarking platform for molecular generation models. *arXiv preprint arXiv:1811.12823*, 2018.

[494] Kristina Preuer, Philipp Renz, Thomas Unterthiner, Sepp Hochreiter, and Güunter Klambauer. Fréchet chemnet distance: a metric for generative models for molecules in drug discovery. *Journal of chemical information and modeling*, 58(9):1736–1741, 2018.

[495] G Richard Bickerton, Gaia V Paolini, Jérémy Besnard, Sorel Muresan, and Andrew L Hopkins. Quantifying the chemical beauty of drugs. *Nature chemistry*, 4(2):90, 2012.

[496] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):8, 2009.

[497] Michael S Lajiness, Gerald M Maggiora, and Veerabahu Shanmugasundaram. Assessment of the consistency of medicinal chemists in reviewing sets of compounds. *Journal of medicinal chemistry*, 47(20):4891–4896, 2004.

RÉSUMÉ

Le processus de découverte de médicaments a un succès limité malgré tous les progrès réalisés. En effet, on estime actuellement que le développement d'un médicament nécessite environ 1,8 milliard de dollars américains sur environ 13 ans. Nous nous concentrons dans cette thèse sur des approches statistiques qui criblent virtuellement un grand ensemble de composés chimique contre un grand nombre de protéines. Leurs applications sont polyvalentes: elles permettent d'identifier des candidats médicaments pour des cibles thérapeutiques connues, d'anticiper des effets secondaires potentiels, ou de proposer de nouvelles indications thérapeutiques pour des médicaments connues. Cette thèse est conçue selon deux cadres d'approches de criblage virtuel : les approches dans lesquelles les données sont décrites numériquement sur la base des connaissances des experts, et les approches basées sur l'apprentissage automatique de la représentation numérique à partir du graphe moléculaire et de la séquence protéique. Nous discutons ces approches et les appliquons pour guider la découverte de médicaments.

MOTS CLÉS

criblage virtuel de médicament, bio-informatique, apprentissage statistique.

ABSTRACT

The rational drug discovery process has limited success despite all the advances in understanding diseases, and technological breakthroughs. Indeed, the process of drug development is currently estimated to require about 1.8 billion US dollars over about 13 years on average. Computational approaches are promising ways to facilitate the tedious task of drug discovery. We focus in this thesis on statistical approaches which virtually screen a large set of compounds against a large set of proteins, which can help to identify drug candidates for known therapeutic targets, anticipate potential side effects or to suggest new therapeutic indications of known drugs. This thesis is conceived following two lines of approaches to perform drug virtual screening: data-blinded feature-based approaches (in which molecules and proteins are numerically described based on experts' knowledge), and data-driven feature-based approaches (in which compounds and proteins numerical descriptors are learned automatically from the chemical graph and the protein sequence). We discuss these approaches, and also propose applications of virtual screening to guide the drug discovery process.

KEYWORDS

drug virtual screening, bioinformatics, machine learning.