# Input noise injection for supervised machine learning, with applications on genomic and image data

Beyrem Khalfaoui

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à MINES ParisTech

# Input noise injection for supervised machine learning with applications on genomic and image data

Injection de bruit pour l'apprentissage automatique supervisé et application sur des données d'images et de génomique


Soutenue par
**Beyrem KHALFAOUI**


Le 26-09-2019

Ecole doctorale n° 432
**Ingénierie des Systèmes, Matériaux, Mécanique, Énergétique**

Spécialité
**BIO-INFORMATIQUE**

Composition du jury :

| | |
|---|---|
| Véronique, STOVEN<br>Pr., Mines Paristech (CBIO) | *Président* |
| Gaël, VAROQUAUX<br>DR, Parietal / Inria | *Rapporteur* |
| Julie, JOSSE<br>Pr., Ecole Polytechnique (CMAP) | *Rapporteur* |
| Jean-Philppe, VERT<br>DR, Mines Paristech / Google Brain | *Examinateur* |
| Julien, CHIQUET<br>CDR, INRAE | *Examinateur* |
| Jean-Philippe, VERT<br>DR, Mines Paristech / Google Brain | *Directeur de thèse* |

MINES ParisTech | PSL

# Abstract

Overfitting is a general and important issue in machine learning that has been addressed in several ways through the progress of the field. We first illustrate the importance of such an issue in a collaborative challenge that provided genotype and clinical data to assess response of Rheumatoid Arthritis patients to anti-TNF treatments. We then re-formalise *Input Noise Injection* (INI) as a set of increasingly popular regularisation methods. We provide a brief taxonomy of its use in supervised learning, its intuitive and theoretical benefits in preventing overfitting and how it can be incorporated in the learning problem. Focusing in this context on the *dropout* trick, we review the related line of work around its understanding and adaptations then provide a novel approximation that can be leveraged for general non-linear models, to get more insight on how *dropout* works. We then present the *DropLasso* method, as both a generalisation of *dropout* by incorporating a sparsity penalty, and apply it in the case of single cell RNA-seq data where we show that it can improve accuracy of the known lasso method while performing biologically meaningful feature selection. Finally we build another generalisation of noise injection where the noise variable follows a structure that can be either fixed, adapted or learnt during training. We present *Adaptive Structured Noise Injection* (ASNI) as a regularisation method for shallow and deep networks, where the noise structure applied on the input of a hidden layer follows the covariance of its activations. We provide a fast algorithm for this particular adaptive scheme, study the regularisation properties of our particular variant on linear and multilayer networks using a quadratic approximation, and show improved results in generalisation performance and in representations disentanglement in real dataset experiments.

# Résumé

Le sur-apprentissage est un problème général qui affecte les algorithmes d'apprentissage statistique de différentes manières et qui a été approché de différentes façons dans la littérature. Nous illustrons dans un premier temps un cas réel de ce problème dans le cadre d'un travail collaboratif visant à prédire la réponse de patients atteints d'arthrose rhumatoïde à des traitement anti-inflammatoires. Nous nous intéressons ensuite à la méthode d'injection de bruit dans les données dans sa généralité en tant que méthode de régularisation. Nous donnons une vue d'ensemble de cette méthode, ses applications, intuitions, algorithmes et quelques éléments théoriques dans le contexte de l'apprentissage supervisé. Nous nous concentrons ensuite sur la méthode du *dropout* introduite dans le contexte d'apprentissage profond et construisons une nouvelle approximation permettant une nouvelle interprétation de cette méthode dans un cadre général. Nous complémentons cette étude par des expériences sur des simulations et des données réelles. Par la suite, nous présentons une généralisation de la méthode d'injection de bruit dans les données inspirée du bruit inhérent à certains types de données permettant en outre une sélection de variables. Nous présentons un nouvel algorithme stochastique pour cette méthode, étudions ses propriétés de régularisation et l'appliquons au context de séquençage ARN de cellules uniques. Enfin, nous présentons une autre généralisation de la méthode d'Injection de bruit où le bruit introduit suit une structure déduite des paramètres du modèle, en tant que la covariance des activations des unités auxquelles elle est appliquée. Nous étudions les propriétés théoriques de cette nouvelle méthode qu'on nomme ASNI pour des modèles linéaires et des réseaux de neurones multi-couches. Nous démontrons enfin que ASNI permet d'améliorer la performance de généralisation des modèles prédictifs tout en améliorant les représentations résultantes.

# Acknowledgements

Je tiens tout d'abord à remercier les membres du Jury: Gaël Varoquaux, Julie Josse, Julien Chiquet, Véronique Stoven et Jean-Philippe Vert d'avoir accepté de faire partie de mon Jury de thèse et mettre une quantité de leur temps à disposition pour la lecture et la réception de mon travail. Je suis honoré de voir des chercheurs avec des contributions énormes dans le domaine et représentant plusieurs instituts prestigieux lire et évaluer mon travail. Je remercie particulièrement les rapporteurs pour leurs retours rapides et leurs remarques perspicaces dont j'ai déjà beaucoup appris, et j'espère surtout que mon travail leur a appris de nouvelles choses aussi !

Je tiens particulièrement à remercier mon directeur de thèse, Jean-Philippe Vert, sans qui ce travail n'aurait jamais vu le jour. Je le remercie tout d'abord pour sa confiance, sa patience à m'écouter, me lire et me corriger, et il en faut ! Je le remercie aussi de m'avoir formé à la rigueur et l'élégance scientifique, mais aussi de m'avoir permis à travers des conférences internationales et notamment la visite du Simons Institute à l'université de Berkeley de rencontrer les plus éminents chercheurs dans le domaine et des esprits aussi brillants que le sien. Sa contribution à ce travail est dans le reflet de chaque paragraphe de ce manuscript (et chaque équation) et sera certainement aussi présente dans mes futurs travaux, quels qu'ils soient.

C'est aussi à tout le CBIO, cette petite équipe de l'école des Mines de Paris à l'impact national et international immense, que je dois cette formation de vie et d'esprit. A commencer par les anciens, Jean-Louis, Elsa et Nelle qui m'ont accueilli à bras et livres ouverts et qui sont restés de chers amis malgré les bifurcations des chemins. C'est aussi à Thomas Walter, son enthousiasme intellectuel et sa gouvernance amicale, à Véronique Stoven, sa grande et profonde humanité et son soutien quotidien, ainsi que Chloé Azencott, sa perspicacité et sa vivacité, que je dois en grande partie la force psychologique dont tout doctorant a besoin pour venir à bout de son doctorat. C'est aussi au CBIO que j'ai noué de grands liens d'amitié avec des doctorants et futurs chercheurs brillants et modestes: J'ai nommé Joseph, Peter, Benoit , Marine, Victor, Romain et Lotfi avec qui j'ai partagé bien plus que des repas à la cantine, et que je suis fier d'avoir connu et le serai probablement encore plus dans quelques années! Je réserve un remerciement particulier à Joseph, non seulement pour les courses autour du jardin du Luxembourg dans le froid et les parties d'échecs sous la chaleur, mais aussi pour sa contribution importante au dernier chapitre sans qui les résultats qui y figurent n'auraient pu être réalisées.

le CBIO, c'est aussi une partie de l'unité U900, affilée à l'institut Curie en partenariat avec l'INSERM et l'école des Mines. Pour ma part, c'est là que j'ai passé la moitié de mon doctorat et c'est entre la Cafét du première étage du bâtiment de biologie de développement et ses quelques amphithéatres, et entre les discussions de biologistes, bio-informaticiens et statisticiens que j'ai eu mes meilleures idées. C'est d'abord tout le personnel de l'institut Curie, ses médecins, infirmiers et tout le corp médical qui traite et accompagne héroïquement des patients tous les jours, toutes les équipes de recherche qui modestement et loin des feux des médias, combat la maladie par le savoir, et tout le personnel qui veille à la survie et à l'épanouissement des équipes (je pense particulièrement aux services d'infrastructure telle que la plateforme bioinfo et au personnel de la cantine, qui égayaient particulièrement mes journées avec leur bonne humeur) que je remercie. Au sein même de l'U900, je tiens à

remercier vivement Emmanuel Barillot, directeur de l'unité, et Andrei Zinovyev pour leur confiance, les discussions stimulantes et leurs encouragements récurrents. Je tiens ensuite à remercier Caroline pour son aide immense dans la traversée des labyrinthes administratifs, son énergie et son humanisme splendides. Je remercie aussi Mihaly pour son soutien moral et intellectuel et son indication de l'importance des données single-cell qui a attiré mon intérêt et donc sans qui le chapitre correspondant n'aurait peut être pas vu le jour. Il est difficile ici de citer toutes les personnes qui ont traversé et enrichi ma route à l'unité et à l'institut et donc que ceux-ci me le pardonnent.

Je tiens ensuite à remercier particulièrement Stefan Wager et Ian Goodfellow pour les discussions fructueuses autour du dropout et l'injection de bruit, et surtout pour le temps précieux qu'ils m'ont accordé ainsi que pour l'amour de la science et de l'art de l'explication! Je suis chanceux d'avoir pu croiser le chemin de tels chercheurs éminents et de tant d'autres, qui m'ont transmis avant tout, l'amour de leur art. Un remerciement en découle donc naturellement pour mes anciens professeurs de mathématiques en Tunisie et en France qui m'ont transmis ce même amour universel.

Je dédie enfin ce travail à ma famille et particulièrement mes parents, qui m'ont tout d'abord créé, façonné et formé avant que la vie, les mathématiques et les nouvelles technologies s'emparent de moi. Je dédie aussi ce travail à Yann Carrier et Camille Viot, dont la curiosité et l'intelligence étaient plus vastes que ce monde qu'ils ont quitté trop tôt. Que leurs corps et esprits reposent en paix.

# Contents

# Liste of figures

# List of Tables

# Chapter 1

# Introduction

> " *People worry that computers will get too smart and take over the world, but the real problem is that they're too stupid and they've already taken over the world.*"
>
> Pedro Domingos

## Contents

## 1.1 Machine learning: past and present

According to different dictionaries and to research in the field, machine learning is a branch of artificial intelligence (AI) that employs a variety of statistical, probabilistic and optimisation techniques that allows computers to learn from past examples and to detect hard-to-discern patterns from large, noisy or complex data sets.

But what is really AI? One can trace already the idea of describing "intelligence" and the possibility to separate, study and even craft it, to the era of classical philosophers such as Descartes, Leibniz and Blaise Pascal, who attempted to describe the process of human thinking as the mechanical manipulation of symbols (Buchanan, 2005). This work found practical sense in the invention of the programmable digital computer in the 1940's [1], a machine based on the abstract essence of mathematical reasoning. This device and the ideas behind it inspired a handful of scientists to begin seriously discussing the possibility of building an electronic brain. The earliest research into learning machines was inspired by a confluence of ideas that became prevalent in the late 1940's, and early 1950's. At that time, recent research in neurology had shown that the brain was an electrical network of neurons that fired in all-or-nothing pulses. Claude Shannon's information theory described digital signals (i.e., all-or-nothing signals). Alan Turing's theory of computation showed that any form of computation could be described digitally. The close relationship between these ideas suggested that it might be possible to recreate an electronic brain. In 1943, Walter Pitts and Warren McCulloch analysed networks of idealised artificial neurons and showed how they might perform simple logical functions (McCulloch and Pitts, 1943). They were the first to describe what later researchers would call a neural network.

The term *machine learning* was first coined in 1959 by Samuel (1959) while constructing programs learning to play checkers, who loosely defined it as a *"Field of study that gives computers the ability to learn without being explicitly programmed"*. However the term remained unused as this aspect was implicitly considered a mere effect of building "machines that think" to quote Alan Turing's famous article (Turing, 1950). Eventually, it became obvious that the dream of a general machine learning algorithm has been largely underestimated, even though the field first witnessed many successes such as Newell and Simon's General Problem Solver (Newell et al., 1959) and Joseph Weizenbaum's ELIZA (Weizenbaum, 1966) that showed real progress towards the goals of machine based problem solving and the interpretation of spoken language respectively. During mid to end of the 1970s, AI was subject to severe criticism that was mainly due to the contrast between the strong initial optimism driving the field, practical issues such as limited computing resources and theoretical issues such as intractability of several essential problems (Karp, 1972). This has caused a certain public disillusionment that led to a strong decrease in funding but has also urged the field to focus on more applied and precise goals on one hand and to build, on the other hand, mathematical formalisms that would allow collaboration with more established and successful fields. The successful development of more focused programs applied in research, such as the DENDRAL program which effectively helped organic chemists in identifying unknown organic molecules, by analysing their mass spectra and using prior knowledge of chemistry (Lederberg, 1987), and in industry, such as the *R1* "expert system" which helped configure orders for new computer systems at Digital Equipment Corporation and resulting in tremendous savings and benefits by 1986 (McDermott, 1982). This led to the revival of funds for the field such as the "Fifth Generation" project in Japan followed by public and private investments in Britain and the United States (Russell and Norvig, 2016). Precise mathematical descriptions were also developed for corresponding "computational intelligence" methods like neural networks and evolutionary algorithms. Simultaneously, new ways of learning for neural networks (Hopfield, 1982; Rumelhart et al., 1986), for which research has continued minimally and particularly in physics, led to the revival and reunification of research around neural networks, marked for

[1] https://en.wikipedia.org/wiki/Colossus_computer

instance by the work of McClelland et al. (1986). Although the field still witnessed financial and philosophical difficulties in the end of 1980's and beginning of 1990's due again to a contrast of expectations and achievements, and a strong criticism of the symbol processing model of the mind (Brooks, 1990), this eventually led to the development and use of more sophisticated mathematical tools and more rigorous frameworks influenced for instance by the landmark work of Pearl (1988). Among the many new tools in use were Bayesian networks, hidden Markov models, information theory, stochastic modelling and classical optimisation (Russell and Norvig, 2016). AI was also present in different subfields focused on particular problems or approaches, sometimes even under different terms that disguised the too ambitious and ambiguous general term. This is where the term "machine learning" gained more popularity.

Retrospectively: focusing on isolated problems, the formalism of different mathematical and algorithmic frameworks, but also the increase of computing power and the availability of data, have all participated in forging the success of the field today, which resulted in applications beyond academic research, and led to the mainstreaming of terms such as machine learning and deep learning. Deep learning is part of machine learning methods based on the layers used in artificial neural networks ("deep" relates to the use of a relatively high number of hidden layers). Its specificity is that it is related to the first models implemented and defended by the fathers of AI and thus appears as one of the strongest signals of the field's come-back. Thanks to computational advances and various tricks (such as the use of graphical processing units), deep learning has recently achieved remarkable success on a wide range of tasks such as image classification (Krizhevsky et al., 2012), scene recognition (Zhou et al., 2014), image captioning (Chen and Lawrence Zitnick, 2015; Vinyals et al., 2015), speech recognition (Hinton et al., 2012a) and visual question answering (Antol et al., 2015) among other applications. Deep Learning models are immensely successful in unsupervised, hybrid and reinforcement learning as well (Mnih et al., 2015). However, only little is known about the theory behind this successful paradigm, although many studies already emerged (Schmidhuber, 2015; Lemberger, 2017). As an example of this success, figure 1.1 shows how the approximate number of published papers related to deep learning is growing exponentially over the previous years (until 2015), followed by the likewise evolution of one of the applications which our lab focuses on, which is bioinformatics, and that will be described in the next section.

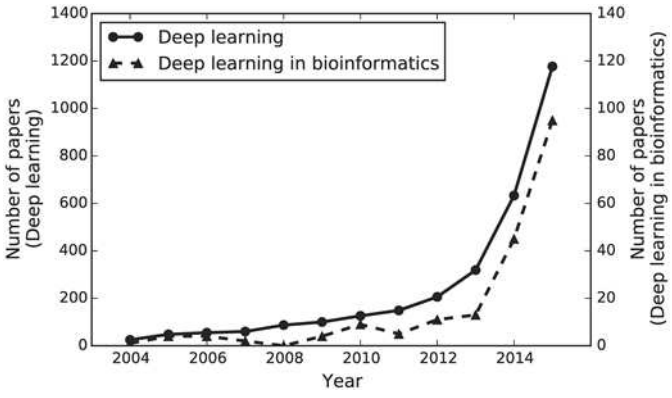



Figure 1.1: Approximate number of published deep learning articles by year. The number of articles is based on the search results on `http://www.scopus.com` with the two queries: "Deep learning", and "Deep learning"+"Bioinformatics". Figure from (Min et al., 2017).

## 1.2 Machine learning and bioinformatics

In all areas of biological and medical research, the role of the computer has been dramatically enhanced in the last to decades. One particular area that can best reflect this influence is

genome sequencing, which aims at mapping the base pairs of an organism's DNA or genome ($\sim$ 3 billion pairs for the human genome). These base pairs, also called nucleotides, are of four types, A, T, C and G, and are the building blocks of DNA. DNA sequencing methods used in the 1970's and 1980's were mainly manual, for example the original Maxam-Gilbert sequencing (Maxam and Gilbert, 1977) and the original Sanger sequencing (Sanger and Coulson, 1975). The shift to more rapid, automated sequencing methods in the 1990's finally allowed for sequencing of whole genomes and as a result enhanced the generation of more reliable and precise data (Smith et al., 1986; Hunkapiller et al., 1991). Sequencing of nearly an entire human genome was first accomplished in 2000 partly through the use of simple or hierarchical shotgun sequencing technologies (Consortium et al., 2001; Venter et al., 2001). Since then, and thanks to advances both in computing power and biological research that accelerated DNA sequencing, costs witnessed a fast decrease, which even surpassed largely the Moore's law by 2008 with the advent of what is termed as second generation sequencing techniques, also known as the next-generation sequencing (NGS), which relies on massively parallel sequencing of short DNA fragments. As shown by figure 1.2, this resulted in a dramatic increase in the number of genomes sequenced, further increasing the pressure towards modelling, synthesis and interpretability of obtained data and its analysis.

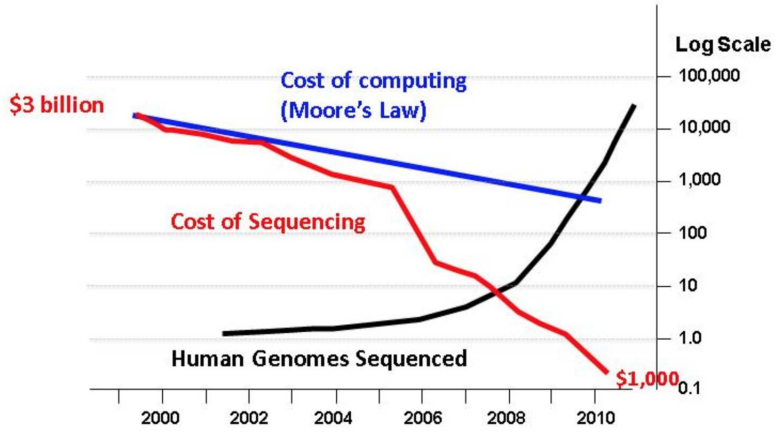



Figure 1.2: The evolution of the cost of computing (blue line), the cost of sequencing (red line) and the number of sequenced human genomes (black line), from 1999 to 2011, on a log scale (adapted from *The Economist*).

The exponential growth of the amount of biological data available raises two important technical issues: on one hand, efficient information storage and management, and on the other hand, the extraction of useful information from these data. These issues led to the development to an important interdisciplinary field through the design of methods and software tools for biological data: Bioinformatics (Hogeweg, 2011). The second issue of understanding and transforming the data into insights is particularly one of the main challenges of machine learning applied in bioinformatics and requires the development of adapted tools capable of transforming all the available biological datasets into meaningful and statistically robust knowledge about the mechanism, its biomarkers or its outcome. These tools and methods should allow to go beyond a mere description of the data and provide knowledge in the form of testable models. By this simplifying abstraction that constitutes a model, we will be able to obtain predictions of the system.

In our Centre for Computational Biology CBIO at MINES ParisTech, we exactly focus on developing and applying statistical methods and machine learning models on a wide variety of biological data, such as biomedical images, genomic datasets (related to the structure, function, evolution, mapping, and expression of genomes) or proteomic datasets (related to the structure, function, and interactions of proteins), with a focus on medical applications, in consideration with the Curie Institute partnership.

Two main paradigms exist in the field of machine learning: supervised and unsupervised learning. Both have potential applications in biological and medicine research.

- In supervised learning, objects in a given collection are classified using a set of attributes, or features. The result of the classification process is a set of rules that prescribe assignments of objects to classes based solely on values of features.
  In a biological context, examples of object-to-class mappings can be tissue gene expression profiles to disease group, and protein sequences to their secondary structures. The features in these examples can be respectively the expression levels of individual genes measured in the tissue samples and the presence/absence of a given amino acid symbol at a given position in the protein sequence. The goal in supervised learning is to design a system able to accurately predict the class membership of new objects based on the available features. Besides predicting a categorical characteristic such as class label (classification), supervised techniques can be applied as well to predict a continuous characteristic of the objects (regression).

- In contrast to the supervised framework, in unsupervised learning, no predefined class labels are available for the objects under study. In this case, the goal is to explore the data and discover similarities between objects. Similarities are used to define groups of objects, referred to as clusters. In other words, unsupervised learning is intended to unveil natural groupings in the data. One example and now quite in the trend of bioinformatics is the discovery of a disease or a tumour hidden subtypes thanks to molecular biomarkers (van Rooden et al., 2010; Grün et al., 2015; Segerstolpe et al., 2016; Li et al., 2017). Unsupervised learning can also be used in density estimation, data compression or the extraction of new features (or representations) from the data to perform other tasks (Samacoits et al., 2018).

Depending on the labels availability and the nature of data, other paradigms such as semi-supervised learning, online and reinforcement learning exist.

In this manuscript we will focus on the supervised learning setting that we formalise in the next section. As for the used biological data in this manuscript, and as part of the CBIO team, we leverage different types of labeled databases to demonstrate the performance of the studied or designed models throughout the manuscript. Biological datasets are therefore here not thouroughly studied as not central to the thesis subject, although the models study might reveal interesting remarks about the features selected or the biological signal to noise in the data. The biological datasets studied include:

- A ”big” dataset that describes genotype and clinical information concerning thousands of Rheumatoid Arthritis patients treated with different therapies. This data is used in the RA challenge and will be described more precisely in the RA challenge chapter.

- Two popular gene expression microarray datasets used in breast cancer prognosis and metastasis analysis. These datasets will be used to test the performance of Input Noise Injection methods and will be more precisely described in the second chapter.

- A total of 4 single-cell RNA sequencing datasets describing gene expressions at the cell level within different conditions of mice tissues. These datasets will be used to assess the performance of our designed DropLasso method and will be more precisely described and studied in chapter 3.

## 1.3 The supervised learning setting

Supervised learning is simply a formalisation of the idea of learning from known examples. In supervised learning, the learner (typically, a computer program) is provided with to sets of

data, a training set and a test set. The idea is for the learner to "learn" from a set of labeled examples in the training set so that it can identify the labels of the examples in the test set with the highest possible accuracy. That is, the goal of the learner is to develop a rule, a program, or a procedure that classifies new examples (in the test set) by analysing examples it has been given that already have a class label.

### 1.3.1 Setting and notations

More formally, let us denote $\mathcal{X}$ the set of examples or the inputs and $\mathcal{Y}$ the set of labels or the outputs. In the supervised learning setting, we are given a series of $n$ pairs of the form $(x_i, y_i)_{i=1,...,n}$ that are assumed to be $n$ realisations of independent and identically distributed pairs of random variables from a joint distribution $\mathcal{P}$ on $\mathcal{X} \times \mathcal{Y}$ , and the goal is to find or construct a good mapping $f : \mathcal{X} \mapsto \mathbb{H}$ that approximates the true labels. For example:

- $\mathcal{X}$ is a set of images that contains a handwritten number and $\mathcal{Y}$ is the set of associated numbers. If a picture is coded through the grey level of its pixels, then $\mathcal{X} = [0, 1]^d$ for $d$ pixels , $\mathcal{Y} = \{0, ..., 9\}$
- $\mathcal{X}$ is a set of patients from which we have collected clinical measures (as age, blood pressure) and $\mathcal{Y}$ is the strength of response to a certain treatment, or the probability of success of the treatment. Depending on the clinical variables, we will generally see $\mathcal{X}$ as $\mathbb{R}^d$ for $d$ measures and $\mathcal{Y} = \mathbb{R}$ or $\mathcal{Y} = [0, 1]$
- $\mathcal{X}$ is a set of cell of which we have measured the gene expression levels, and $\mathcal{Y}$ is the state of the cell (as the stage of development). Then $\mathcal{X} = \mathbb{R}^d$ for d genes and $\mathcal{Y} = \{0, ..., n\}$ here $n$ is the number of possible states.

In order to assess how good a function $f$ can be, we need to define a measure called in this context the loss function $L : \mathcal{Y} \times \mathbb{H} \mapsto \mathbb{R}+$, which compares the model predictions to the ground truth. Examples of classical loss functions that we will focus on in this manuscript include:

- The square loss for regression, where $\mathcal{Y} = \mathbb{R}$ and:

$$L_{\text{square}}(y, \hat{y}) = (y - \hat{y})^2 \ .$$

- The logistic loss function for classification where $\mathcal{Y} = \{-1, 1\}$ and:

$$L_{\text{logistic}}(y, \hat{y}) = \log\left(1 + \exp(-y.\hat{y})\right) \ .$$

- The cross-entropy loss function with softmax for classification where $\mathcal{Y} = \{0, K - 1\}$, $K$ being the number of classes and $y^{(k)}$ the k-th component of the label's hot-encoding [2], and:

$$L_{\text{cross-entropy}}(y, \hat{y}) = -\sum_{k=1}^{K} y^{(k)} \log\left(\frac{e^{\hat{y}^{(k)}}}{\sum_{i=1}^{K} e^{\hat{y}^{(i)}}}\right) \ .$$

We aim to construct a mapping that produces on average, a small error between the output and the true labels. This average error cost is called the risk and defined as:

$$R(f) = \mathop{\mathbb{E}}_{(X,Y)\sim\mathcal{P}} \left[L\left(f(X), Y\right)\right] \ . \tag{1.1}$$

Since we only have $n$ realisations of labeled examples and that the joint distribution $\mathcal{P}$ is usually unknown, one relies on what is called the empirical risk as an approximation, which

[2] the representation of categorical variables as $K$-dimensional binary vectors

is defined as the average of the loss between the mapping output and the true labels over all observed labeled data points:

$$R_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n} L\left(f(x_i), y_i\right) . \tag{1.2}$$

Looking for a mapping $f$ that minimises the empirical risk $R_{emp}(f)$ is known as the *Empirical Risk Minimisation* (ERM) framework.

Very often, $f$ will be a parametric model that depends on a vector of weights $w$, which dimension will be related to the complexity of the model. For parametric models, we will denote $f$ as $f_w$ and the optimisation problem will become finding the minimum of $R_{emp}(w)$, where:

$$R_{emp}(w) = \frac{1}{n}\sum_{i=1}^{n} L\left(f_w(x_i), y_i\right) . \tag{1.3}$$

### 1.3.2 Gradient Descent

It has often been proposed (Rumelhart et al., 1986) to minimise the empirical risk $R_{emp}(w)$ using gradient descent (GD). Each iteration updates the weights on the basis of the gradient of $R_{emp}(w)$. At iteration $t$, the weights are indeed updated as follows:

$$\begin{aligned} w^{(t+1)} &= w^{(t)} - \gamma \nabla_w R_{emp}(f_{w^{(t)}}) \\ &= w^{(t)} - \frac{\gamma}{n}\sum_{i=1}^{n} \nabla_w L(f_{w^{(t)}}(x_i), y_i) , \end{aligned}$$

where $\gamma$ is an adequately chosen learning rate and $n$ is again the number of available training samples.

Under sufficient regularity assumptions, when the initial estimate $w^{(0)}$ is close enough to the optimum, and when the learning rate $\gamma$ is sufficiently small, this algorithm achieves linear convergence, that is, $-log(\rho) \sim t$ , where $\rho$ represents the residual error (Dennis Jr and Schnabel, 1996). GD is thus a simple and general optimisation algorithm that is suitable for any parametric model, however it does suffer certain limitations:

- It is only suited for differentiable functions (which is the case for the mentioned loss functions)
- It is not guaranteed to converge to the global minimum when it exists if the loss function is not convex.
- It needs to compute all the gradients with respect to each data sample, which can make an iteration very slow if the the gradient computation cost is high.

Other minimisation algorithms and variants of GD have been designed to circumvent theses limitations when they occur. The goal here is not to review the associated optimisation algorithms (see (Sra et al., 2012; Bach et al., 2012) for a complete survey), but rather to justify our later extensive use of the stochastic gradient descent (SGD).

### 1.3.3 SGD

For our mentioned loss functions, disadvantage of GD is that sometimes it may be too expensive to compute the gradient of a function, especially in the case of a large number of observations (referred to as large-scale optimisation). The basic idea of SGD is to instead use an estimator for the empirical risk gradient at each iteration. An obvious estimate is the gradient at one randomly chosen observation. SGD is therefore a randomised version of

GD that requires in its classical form a single randomly picked example $x^{(t)}$ and updates the weights at iteration $t$ as follows:

$$w^{(t+1)} = w^{(t)} - \gamma_t \nabla_w L(f_{w^{(t)}}(x_t), y_t)\,.$$

Under two conditions on the decreasing learning rates $\gamma_t$, the Robbin-Siegmund theorem (Robbins and Siegmund, 1971) ensures convergence under surprisingly mild conditions, including for loss functions that are not everywhere differentiable. These conditions can be ensured by the convergence of $\sum_t \gamma_t^2$ and the divergence of $\sum_t \gamma_t$.

In cases where the model is complex or more generally when computing the gradient of the loss with respect to the weights is costly, as in the case of deep neural networks, SGD is widely used. Another case which we will see in this manuscript is when the empirical risk does not have a closed-form gradient or when its gradient can not be directly evaluated, as it can be expressed as the expectation of another random variable for instance. In this case an estimate of the gradient at one sample can replace the whole expectation (examples will be given in the next chapter).

Many improvements on the basic SGD algorithm have been proposed and used. A first obvious extension to improve the gradient estimate and thus potentially accelerate the algorithms is to take the gradient estimate over a small set of observations, which is termed as mini-batch SGD. Making use of averaging and moments have also shown improvement in speed of convergence of SGD (Ruder, 2016), which can be critical especially for multilayer feed-forward networks. Thus, we make use in the last chapter the Adam (Adaptive Moment Estimation) which can basically be seen as SGD with an adaptive learning rate that is estimated from the updated weights at each iteration (Kingma and Ba, 2014).

### 1.3.4 Linear models

The simplest form of parametric models (and neural networks) are linear models, with the weight vector having the same dimension $d$ as the observations, and:

$$f_w(x) = \sum_{j=1}^{d} w_j x_j + b = w \cdot x + b\,.$$

where $w \cdot x$ is the scalar product between $w$ and $x$, and $b$ is called the bias term. One can remove the bias term from the model's expression thanks to the bias trick, which consists in adding a bias dimension with a constant of 1 to $x$ and thus also augmenting $w$ dimension by 1. We will thus only consider the learning of the weights $w$ and omit this addition for linear models in the future. In the case of a linear model, the empirical risk to minimise becomes:

$$R_{emp}(w) = \frac{1}{n} \sum_{i=1}^{n} L\left(w \cdot x_i, y_i\right)\,.$$

### 1.3.5 Artificial neural networks

In this section, we describe a broader family of models that we already mentioned under the name of neural networks. Neural networks terminology is inspired by the biological operations of specialised cells called neurons. A neuron is an information-processing unit that is fundamental to the operation of a neural network. The block diagram at figure 1.3 below shows the model of a neuron, which forms the basis for designing the larger family of neural networks. Here, we identify three basic elements of the neural model:

1. A set of synapses, or connecting links, each of which is characterised by a weight or strength of its own. Specifically, a feature $x_j$ at the input of synapse $j$ connected to

neuron is multiplied by the synaptic weight $w_j$. Unlike the weight of a synapse in the brain, the synaptic weight of an artificial neuron may lie in a range that includes negative as well as positive values.

2. An adder for summing the input signals, weighted by the respective synaptic strengths of the neuron; the operations described here constitute a linear combiner

3. An activation function $\sigma$. There are several canonical functions that will be mentioned in this manuscript, mainly the:

   - Linear activation: $\sigma(z) = z$ .
   - Sigmoid activation: $\sigma(z) = \frac{1}{1+\exp(-z)}$ .
   - RELU activation: $\sigma(z) = z \cdot \mathbf{1}(z > 0)$ .

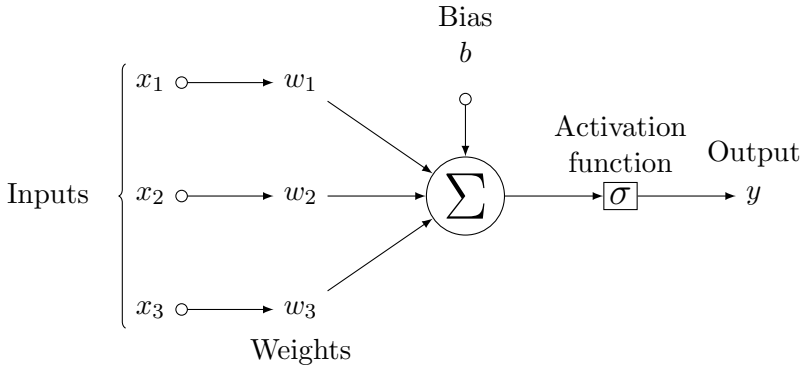



Figure 1.3: Architectural graph of a neuron

A linear model is thus the simplest example of a network with one unit and a linear activation.

In a multi-layer network, the neurons are organised in the form of layers. let us introduce some useful notations here to construct a neural network with $H$ hidden layers. Now for a layer $l \in [1, H]$ let us assume we have a number of $d_l$ units, and thus by concatenating the weights we obtain a weight matrix $W^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ and a bias term $b^{(l)} \in \mathbb{R}^{d^{(l)}}$, with the convention $d^{(0)} = d$. The network defines a function $f_w : \mathbb{R}^d \to \mathbb{R}^{d^{(H)}}$ given for any $x \in \mathbb{R}^d$ by $f_w(x) = y^{(H)}$, where $y^{(l)}$ is the layer's $l$ output, defined recursively for $l = 0, \ldots, H$ by $y^{(0)} = x$ and, for $l \in [1, H]$:

$$\begin{cases} z^{(l)} &= W^{(l)} y^{(l-1)} + b^{(l)} , \\ y^{(l)} &= \sigma^{(l)}(z^{(l)}). \end{cases}$$

This hidden layer creates an intermediate representation of the input data. The essence of *deep learning* is to model a hierarchy of multiple representations. Alternatively, one can see a multilayer network as a complex function defined as compositions of activations functions and bias additions:

$$f(x) = \sigma(W^{(H)}\sigma(W^{(H-1)}(\cdots\sigma(W^{(1)}x + b^{(1)})\cdots) + b^{(H-1)}) + b^{(H)}).$$

A simple example of a two hidden layers neural network architecture that we will later use in the last chapter is given in figure 1.4.

Multi-layer neural networks have many desired properties. The first and foremost is that multi-layer neural network are *universal function approximaters*. Indeed, the addition of the hidden layers in multi-layer neural networks overcomes the limitations of linear models that are only capable of learning linearly separable patterns as it has been pointed out in (Minsky and Papert, 2017). It was first proved in (Cybenko, 1989) that every continuous

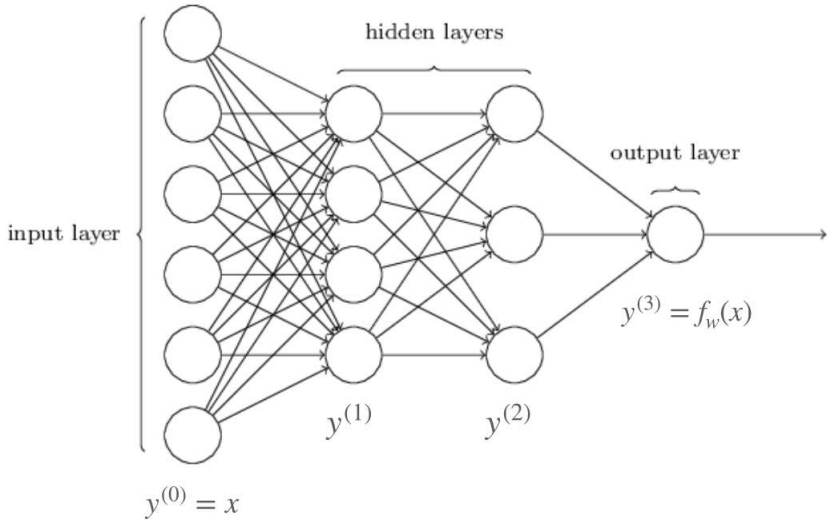



Figure 1.4: Architecture of a 3 layers network with one input layer, two hidden layers and an output unit.

function, mapping intervals of real numbers to another interval of output numbers can be approximated with arbitrarily precision by using sigmoid activation function in a multi-layer neural network. However, the universal approximation theorem does not answer how to adjust the weights in neural networks to improve a given loss function. From a practical point of view however, and for many years, it has always been a challenge to train neural networks that have one or more hidden layers. It is until discovering error back-propagation (Rumelhart et al., 1986) that learning by GD provided an efficient algorithm to train the neural networks. Back propagation is used to compute efficiently the gradients at each iteration by exploiting the chain rule to the graph structure of model. Although being extremely interesting, we will bypass here the full description of the backpropagation trick and may refer the curious readers to the original paper. A neural network may therefore be used for classification and regression, the output passed through the least square error, previously defined, in the latter case. Despite its complexity, and thanks to back-propagation and other recent tricks, this model can be trained with standard GD or SGD techniques.

**Convolutional Neural Networks (CNNs)** will also be used in the last chapter of this manuscript. CNNs are analogous to traditional dense multilayer neural networks in that they are comprised of neurons that get their weights optimised through learning. Each neuron will still receive an input and perform a operation (such as a scalar product followed by a non-linear function), the basis of all neural networks. From the input raw image vectors to the final output of the class score, the entire of the network will still express a single perceptive score function (the weight). The last layer will be again confronted to loss functions associated with the classes, and most of the regular tips and tricks developed for traditional neural networks still apply. The only notable difference between CNNs and traditional neural networks is that CNNs are primarily used in the field of pattern recognition within images. This allows us to encode image-specific features into the architecture, making the network more suited for image-focused tasks - whilst further reducing the parameters required to set up the model. CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, a CNN architecture has been formed. The basic functionality of the example CNN above can be broken down into four key areas:

1. As found in other forms of multilayer networks, the input layer will hold the pixel values of the image.

2. The **convolutional layer** will determine the output of neurons of which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume.

3. The **pooling layer** will then simply perform downsampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation

4. The fully-connected layers will then perform the same operations found in standard multilayer networks and attempt to produce class scores from the activations, to be used for classification or regression.

The building **convolutional layer** and **pooling layer** can be done in several ways depending on the convolutional kernel and the pooling method (although we talk generally about max-pooling) (Schmidhuber, 2015). Convolutional networks have some important desirable properties as are known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. The first successful applications of Convolutional Networks were developed by Yann LeCun in 1990's (LeCun et al., 1995). Of these, the best known is the LeNet architecture that was used to read zip codes, digits, etc... An example of the LeNet architecture with chosen numbers of units we used in the last chapter is shown in figure A.1.
The first work that popularised Convolutional Networks in Computer Vision was the AlexNet, developed by Krizhevsky, Sutskever and Hinton (Krizhevsky et al., 2014). The AlexNet was submitted to the ImageNet ILSVRC challenge in 2012 and significantly outperformed the second runner-up (top 5 error of 16% compared to runner-up with 26% error). The Network had a very similar architecture to LeNet, but was deeper, bigger, and featured Convolutional Layers stacked on top of each other.

## 1.4 The overfitting phenomenon

### 1.4.1 Optimisation versus generalisation

What makes machine learning, and more particularly supervised learning different from simple optimisation? Supervised learning, as we saw, shares a lot with the optimisation field, in terms of framework, theory and algorithms. However, the main and most fundamental difference is that in supervised learning, the model needs to have a small prediction error on *unseen examples* while it is trained on a different data set, whereas in optimisation one is usually interested in the convergence to the minimum of the function minimised or to minimising the approximate error to the minimum value. Here the function, as introduced earlier, to be minimised is the empirical risk, but one is ultimately interested in minimising the expected risk which is potentially unknown.

Maybe the most intuitive example illustrating this important remark, is the choice of the mapping:

$$f(x) = \begin{cases} y_i \text{ if } x = x_i \\ \text{any value otherwise} \end{cases},$$

such that the empirical risk is exactly equal to zero for any number of samples (the function is fixed given the training set), while the true risk is nonzero and arbitrarily high. Such functions only memorise the training points without learning anything about the underlying distribution.
One can argue that choosing a family of functions that does not contain this extreme example will solve the problem. But one can already observe the same problem when the models class is for instance a family polynomial functions of degree $M \leqslant 10$ trained on a simple simulation setting (see figure 1.5). What is happening here is that the more flexible polynomials with larger values of M are becoming increasingly tuned to the random noise on the target values.

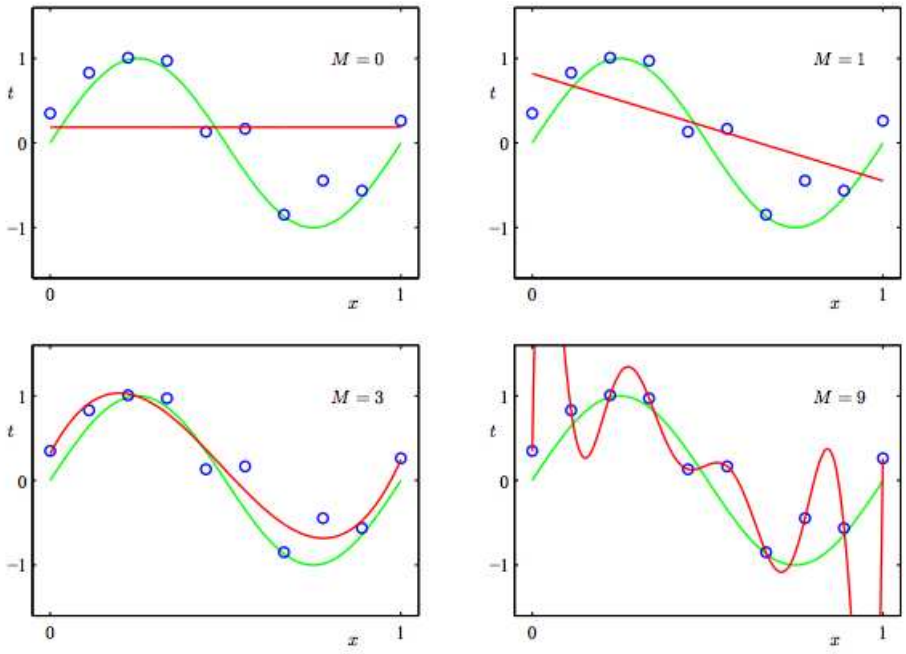



Figure 1.5: Plots of polynomials having various orders of M, shown as red curves, fitted to a simulated data set, from (Bishop, 2006)

Therefore, and perhaps counter-intuitively, this expected risk is not minimised by learning the training data as precisely as possible. The main reason behind is that an extremely close fit to training data tends to generalise poorly to future data, because such a fit inevitably entails fitting random aspects of the sample (i.e., noise) as well as regular components. Any model that learns every quantitative detail of the training data inevitably including many that will never be repeated misses the broader regularities in the data. Fitting training data too closely in the sense of fitting noise as well as real trends is often referred to as overfitting, while fitting it too loosely: missing real trends as well as noise is called underfitting. This basic tradeoff arises in a wide variety of settings, and seems to be fundamental to the very nature of generalisation. Every real data source involves a mixture of regular and stochastic elements, and effective generalisation requires finding the right balance between them so that the regular (repeatable) components may be learned and the noise disregarded. One can write this precisely in what is called the bias-variance tradeoff.

### 1.4.2 The bias-variance tradeoff

The bias-variance tradeoff refers to a result illustrating the expected behaviour of a parameter estimate, with respect to the "true" parameters it approximates. Bias represents *underfitting*, and variance represents *overfitting*. This tradeoff is reflected in the *no free lunch theorems*[3]. This illustrates the power of the implicit assumptions we make when modelling data, for example, that the ground truth function is smooth with Gaussian noise. If the data is uniformly random, there will be no signal to learn.

Let $Y = f(x) + \epsilon$ be the ground truth value with $f(x)$ the true signal for data point $x$, and a random noise $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Let our model estimate be denoted by $\hat{f}(x)$. Then, the mean square error between an estimate and the true parameters, averaged over all possible data,

[3]No free lunch in machine learning refers to the fact that, averaged over all possible ground truths, no model is better than random guessing

$$\begin{aligned}\mathbb{E}[(Y-\hat{f}(x))^2] &= \mathbb{E}[((f(x)+\epsilon-\mathbb{E}[\hat{f}(x)])+(\mathbb{E}[\hat{f}(x)]-\hat{f}(x)))^2] \\ &= \sigma^2+\mathbb{E}[f(x)-\mathbb{E}[\hat{f}(x)]]^2+\mathbb{E}[(\hat{f}(x)-\mathbb{E}[\hat{f}(x)])^2] \\ &= \text{noise}+\text{bias}^2+\text{variance}\end{aligned}$$

The discrepancy therefore depends both on the variance of the parameters around its on mean, and the bias (the gap between this estimate mean and the true parameters[4]). An unbiased estimate is a parameter model that converges (in probability) to a hypothetical true parameter set as data increases, but it may entail a higher variance, making it more likely to draw a bad estimate some of the time. Hence, if we wish to be more certain about the optimality of our estimate, it may be prudent to allow some bias if it sufficiently reduces the variance. In conclusion, the bias-variance tradeoff illustrates that on average it may benefit to use simpler models, even though this invokes a small bias.
The bias variance tradeoff during training is illustrated in the figure 1.6 below.

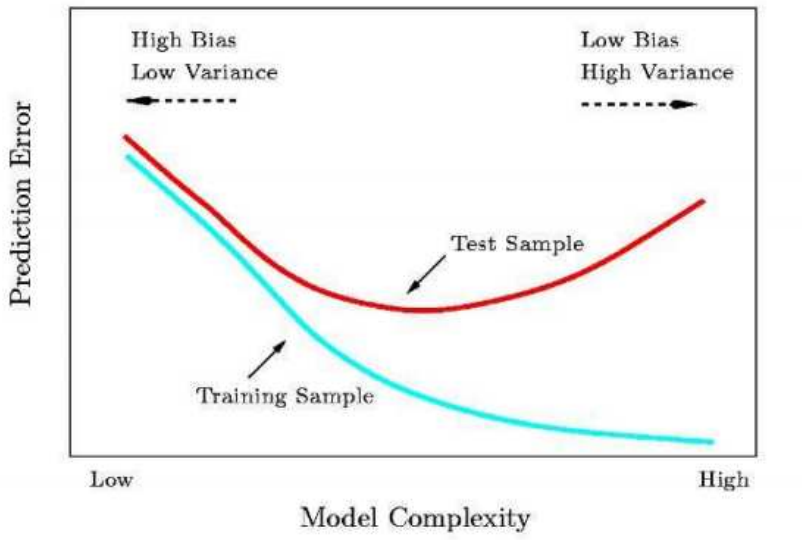



Figure 1.6: The bias-variance tradeoff in learning, from (Friedman et al., 2001)

## 1.5 Assessing overfitting

As presented, overfitting may seem as a merely theoretical problem that can be easily avoided by either getting more data or by reducing the complexity of the data. First, overfitting is actually not an easy problem in general, because the potential for overfitting depends not only on the number of parameters and data, but also the bias in the training data values or distribution, the conformability of the model structure with the data shape. From a practical point of view, overfitting can lead to high-profile false discoveries dramatic consequences that can go from the selection of spurious features that results in various additional costs and a sometimes sensitive instability in feature selection such as that of biomarkers of complex traits and disease (Haury and Vert, 2010) and therefore a poor academic reproducibility

[4]The flaw here is that we require the structural assumptions of the model to be correct before we can talk about "true" parameters, which is rarely the case. This discrepancy is known as "structural error". Even if the model structure is correct, there is an inescapable error of noise that all models share called the *noise floor*. The noise floor is the limit of the learning curve (the trend of improvement in generalisation error as data is added) for an unbiased estimate.

(Ruschhaupt et al., 2004), to failures in trend tracking systems such as the example Google Flu trends (Lazer et al., 2014) or unethical learning algorithms in everyday use (Zou and Schiebinger, 2018). This highlights the trickery of overfitting and press on the use of a principled way to assess overfitting in most cases.
Looking again at the definition of overfitting, the most natural way to estimate the generalisation performance of a learning algorithm is to simulate a setting of unseen dataset by partitioning the given data into a training set and a validation set, to estimate the model parameters on the training set and to test the model's ability to predict on the initially hidden validation set. This is called **hold-out validation**. This method is widely in the evaluation of multilayer networks performance in the case of large datasets as a fast evaluation procedure. The downside is that this procedure does not use all the available data and the results are highly dependent on the choice for the training/test split. These problems can be partially addressed by repeating hold-out validation multiple times and averaging the results, but unless this repetition is performed in a systematic manner, some data may be included in the test set multiple times while others are not included at all, or conversely some data may always fall in the test set and never get a chance to contribute to the learning phase.
To deal with these challenges and utilise fully the available data, one can use **k-fold cross-validation**. In k-fold cross-validation the data is first partitioned into k equally (or nearly equally) sized segments or folds. Subsequently $k$ iterations of training and validation are performed such that within each iteration a different fold of the data is held-out for validation while the remaining $k-1$ folds are used for learning. The cross-validation procedure can be used for three possible reasons:

- To estimate the generalisation performance of the learned model from available data using one algorithm.
- To compare the performance of two or more different algorithms and find out the best algorithm for the available data.
- To compare the performance of two or more variants of a parameterised model, such as the same model with different hyper-parameters tuning.

If the goals are both about model selection (for example selecting the best model hyper-parameters) and its generalisation assessment, a k-fold cross-validation with both a validation and test set can be used. Again, one by one, a set is selected as test set. Then, one by one, one of the remaining sets is used as a validation sets and the other k-2 sets are used as training sets until all possible combinations have been evaluated. The training set is used for model fitting and the validation set is used for model evaluation for each of the hyper-parameter sets. Finally, for the selected parameter set, the test set is used to evaluate the model with the best parameter set. This procedure however requires additional computations and requires a sufficient number of samples so that the learning algorithm does not underfit at training (Arlot and Celisse, 2010).
All of these validation methods will be used across at the different experimental parts of this manuscript for both model selection and performance evaluation. Figure 1.7 shows a visualisation the different methods.

In order to assess the generalisation performance on a validation or a test set, several performance measures can be computed depending on the the context (regression or classification):

In the regression setting, given $\hat{y}_i$ the value of the prediction for a sample $x_i$ and $y_i \in \mathbb{R}$ the true response , we will mostly use the Mean Squared Error $\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$ or Pearson's correlation between $\hat{y}_i$ and $y_i$.

In the classification setting, there are several measures at disposal. When $y_i \in \{-1, +1\}$ for instance, the prediction $\hat{y}_i$ can be computed by thresholding the prediction $f(x_i)$ at

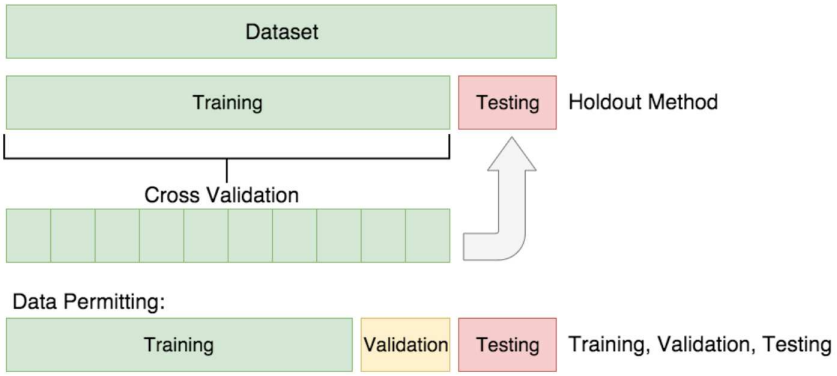



Figure 1.7: Visual Representation of hold-out and cross-validation

$\theta$, usually equal to 0. When $f(x_i)$ is greater than $\theta$, $\hat{y}_i = +1$ and $\hat{y}_i = -1$ otherwise. Then one can use the classical quantities described in table 1.1 in order to define several possible accuracy measures :

Table 1.1: **Classification setting**: Four necessary measures. TN : True Negatives; FN: False Negatives; FP: False Positives; TP: True Positives.

| | | $y$ | | |
|---|---|---|---|---|
| | | -1 | +1 | |
| $\hat{y}$ | -1 | TN | FN | $\hat{n}_-$ |
| | +1 | FP | TP | $\hat{n}_+$ |
| | | $n_-$ | $n_+$ | $n$ |

- classification accuracy: $\frac{TP+TN}{n}$
- Precision: $\frac{TP}{\hat{n}_+}$
- Sensitivity/True Positive Rate (TPR)/Recall: $\frac{TP}{n_+}$
- Specificity/True Negative Rate (TNR): $\frac{TP}{n_-}$
- AUC/AUROC: Area Under the ROC (Receiver Operating Characteristic) Curve
- AUPR: Area Under the Precision/Recall Curve

The ROC (resp. PR) curve is drawn by successively recomputing TPR and FPR (resp. Precision and Recall) for an increasing discrimination threshold $\theta$. AUC and AUPR are then evaluated by computing the area under these curves. We will use more the AUC and classification accuracy throughout this manuscript, and occasionally the AUPR.

Testing the accuracy of the model on an unseen dataset allows in principle to assess if the model is overfitting, but does not correct for this problem. A central goal of reseach in machine learning has been to design and develop methods to prevent overfitting and thus improve generalisation, which is the essence of machine learning. These methods fall under the hat of what is called *regularisation*.

## 1.6 Preventing overfitting

The phenomenon of overfitting through the previous examples illustrates why the complexity of the space $\mathcal{F}$ of functions from which the model will be chosen needs to be controlled. The choice of $\mathcal{F}$ will typically encode assumptions we are willing to make about the data or some prior knowledge about the problem, target or features.
The most general assumption that the vast majority of learners rely on is smoothness, with

the idea that small changes in the input should lead to small changes in the output. It is also common to rely on different assumptions, such as enforcing sparsity, invariance, or just directly constraining the search space $\mathcal{F}$. Overall, it is not possible to build a successful predictor without making any assumptions about the underlying data distribution $\mathcal{P}$. This is in essence the message conveyed by the no free lunch theorem (Wolpert, 2002; Wolpert and Macready, 1997).
For parametric models which are considered in this manuscript, constraining the model amounts to setting constraints on the value or structure of its parameters. The new constrained problem will hence be:

$$\begin{cases} w = \min\limits_{w \in \mathbb{R}^D} R_{emp}(w) \\ \quad \Omega(w) \leq C\,, \end{cases} \tag{1.4}$$

where the function $\Omega(w)$ is a scalar model-dependent function usually called the penalty, and $C$ a scalar setting the intensity of such a penalty.

The process of introducing additional constraints as a technique to combat the overfitting problem is known as *penalisation* or more often *regularisation*, which will be the mainly used term in this manuscript. The choice of which constraints to add, how and when to introduce these constraints in the optimisation problem has led to several families of regularisation methods, going therefore beyond regularisation as first defined for general ill-posed problems both in why and how it is used (Kukačka et al., 2017). Regularisation has been indeed a central and active research topic in machine learning as it is a key component for ensuring good generalisation (Girosi et al., 1995; Bishop, 2006) and has witnessed even more interest with the recent popularisation of deep neural networks. In the latter case, complex networks with several hidden layers tend to have several orders of magnitude more parameters than training examples, statistical learning theory (Abu-Mostafa, 1989) indicates that regularisation becomes even more crucial.
Different taxonomies have been proposed to distinguish and classify different regularisation techniques. Following the ERM framework and the new constrained problem 1.4, Kukačka et al. (2017) distinguish 5 different aspects of the problem on which regularisation can act to influence the values of the weights $w$:

- The training set: $(x_i, y_i)_{i=1,\dots,n}$ (such as data augmentation, normalisation, adversarial training etc ...).
- The selected model family: $\mathcal{F}$ (such as restraining the set $\mathcal{F}$ to linear models, convolution or pooling layers).
- The choice of the regularisation term $\Omega$ (such as $\ell_2$-norm or $\ell_1$-norm regularisation).
- The loss function for defining the risk $R_{emp}$ (such as the choice of a robust loss (Huber, 1992; Holland and Welsch, 1977)).
- The optimisation procedure itself (such as the choice of initialisation, update and termination rules).

We have used along this manuscript and will focus in this section on the first three mentioned categories and mainly: adding a regularisation term, data augmentation and ensemble learning, as we mainly use and compare with these methods in the next chapters.
Although quite straightforward and easy to apply, this classification of regularisations methods maintains some ambiguity for many regularisation methods that can belong to different categories or methods that do not clearly fit in one category as noticed already by the authors in (Kukačka et al., 2017).
Another line of work distinguishes two main types of regularisation: explicit and implicit regularisation (Neyshabur, 2017; Lemberger, 2017):

- Explicit regularisation techniques are those specifically and solely designed to constrain the effective capacity of a given model in order to reduce overfitting. Furthermore, explicit regularisers are not a structural or essential part of the network architecture, the data or the learning algorithm and can typically be added or removed easily.

- Implicit regularisation is the reduction of the generalisation error or overfitting provided by characteristics of the network architecture, the training data or the learning algorithm, which are not specifically designed to constrain the effective capacity of the given model.

Although it can be seen as rather a qualitative interpretation of regularisation mechanisms, and even if it might bear similar ambiguities of methods related to both categories, this is in our opinion an interesting classification that underlines a striking gap between these two categories despite shared methods. As it will be later discussed. In the remaining of this section, we describe some popular regularisation techniques which have proven to be very effective in practice and that we will be using or comparing novel methods with throughout our work. Because of the importance of this topic, there is a huge amount of work that has been done in this area and covering all the aspects of regularisation is beyond the scope of this manuscript.
Coming back to the ERM setting, If $R_{emp}$ and $\Omega$ are both convex and under weak additional assumptions (see, e.g., (Boyd and Vandenberghe, 2004) Section 5.2), equation 1.4 can be written in its Lagrangian form:

$$\min_{w \in \mathbb{R}^D} \{R_{emp}(w) + \lambda \Omega(w)\} \ . \tag{1.5}$$

$\Omega$ is again the regulariser appearing in 1.4 as a constraint and now as a penalty term. The parameter $\lambda \in \mathbb{R}_+$ is called the regularisation parameter and controls the balance between the empirical risk and the model complexity.

Different penalties encode implicitly for different assumptions or preferences about the model. Many penalties have therefore been proposed over the last decades depending on the model, goal and data at hand, creating a sometimes confusing diversity (Bishop, 2006; Ma and Huang, 2008). We first briefly describe two main penalties for their wide use and characteristic properties.

### 1.6.1 $\ell_2$-norm regularisation

$\ell_2$-norm regularisation is a commonly used technique in machine learning, also sometimes referred to as *ridge regression* in the context of linear regression with least squares, as *Tikhonov regularisation* in the optimisation context, and as *weight decay* in the artificial neural networks literature (Hoerl and Kennard, 1970; Krogh and Hertz, 1992). It works by adding a quadratic term to the empirical risk, which results in the new objective to minimise :

$$R_{emp}(w) + \lambda \|w\|_2^2 = \frac{1}{n} \sum_{i=1}^{n} L\left(f_w(x_i), y_i\right) + \lambda \sum_{j=1}^{d} w_j^2 \,, \tag{1.6}$$

where $\lambda$ is again a hyperparameter which appropriate value needs to be chosen as part of the training process (for example by using the validation data set as previously indicated). By choosing a value for this parameter, we decide on the relative importance of the regularisation term versus the data-dependent risk term. If the loss function is convex, then the resulting modified empirical risk is again convex (and smooth if the loss function is smooth), allowing a variety of methods to be used for solving (Boyd and Vandenberghe, 2004), the ridge penalty being smooth and convex.

In order to gain an intuitive understanding of how $\ell_2$ regularisation works against overfitting, one can consider the linear setting with least squares loss with a given dataset

$(x_i, y_i)_{i=1,\dots,n}$. In this case, and if the design matrix $X = (x_{i,j})_{i=1\dots n;j=1\dots d}$ (that is each row is the observation $x_i$) is such that $X^\top X$ invertible, then the minimum of the empirical risk, also called the *Ordinary Least squares* (OLS) solution is:

$$\hat{w}_{OLS} = \underset{w\in\mathbb{R}^d}{\arg\min} \|Y - Xw\|_2^2 = (X^\top X)^{-1} X^\top Y \,, \tag{1.7}$$

where $Y = (y_i)_{i=1\dots n}$ is the column vector embedding of the training labels. This solution however, is not always possible to choose, in particular when the number of features $d$ exceeds the number of samples $n$, making the matrix $X^\top X$ singular.

The solution to the $\ell_2$-norm regularised problem 1.6 in the linear least squares case, termed as ridge, however always exists:

$$\hat{w}_{ridge} = \underset{w\in\mathbb{R}^d}{\arg\min} \|Y - Xw\|_2^2 + \lambda \|w\|_2^2 = (X^\top X + \lambda I_d)^{-1} X\top Y \,, \tag{1.8}$$

where $I_d$ is the identity matrix. The addition of the $\ell_2$-norm penalty first makes the problem nonsingular, even if is not of full rank, and therefore improving the stability of the estimation which was the main motivation for ridge regression when it was first introduced in statistics (Hoerl and Kennard, 1970).
Another crucial and related property of the $\ell_2$-norm regularisation that allows for a potential barrier to overfitting is the shrinkage in coefficients resulting from 1.8. This monotone decreasing function of $\lambda$ basically reduces the search space $\mathcal{F}$ and might therefore improve performance in case of high dimensional data or in the case of overfitting more broadly. This explains the related term of weight decay widely used for decades in the literature of neural networks as a simple yet effective technique in improving generalisation performance (Krogh and Hertz, 1992).

An other interesting interpretation of $\ell_2$-norm regularisation is provided by Bayesian statistics, that consider the penalty as a prior distribution on $w$: assume that $p(w) = \mathcal{N}(0, \frac{1}{\lambda} I_d)$ and $p(Y|X, w) = \mathcal{N}(Xw, 1)$. Then minimising minus the log-likelihood, that is looking for the most likely model given the training data, is equivalent to solving problem 1.8. Indeed, by definition of the conditional density, the posterior distribution of the model is written as:

$$\begin{aligned} p(w|X, Y) &= \exp\left(-\frac{1}{2}\|Y - Xw\|_2^2 - \frac{\lambda}{2}\|w\|_2^2\right) \\ &= \exp\left\{\frac{1}{2}(w - \Gamma_\lambda X^\top Y)^\top \Gamma_\lambda^{-1}(w - \Gamma_\lambda X^\top Y)\right\} \,, \end{aligned}$$

where $\Gamma_\lambda = (X^\top X + \lambda I_d)$. We thus recover the *posterior* distribution

$$p(w|X, Y) = \mathcal{N}(\Gamma_\lambda X^\top Y, \Gamma_\lambda)$$

The Maximum *a posteriori* (MAP) is therefore the mean $\Gamma_\lambda X^\top Y$, corresponding to the ridge solution 1.8. We can see now how increasing will yield an estimator with both a smaller mean, which justify the term shrinkage and might add bias the weights, and a smaller variance, which can potentially reduce overfitting and improve stability (Vinod, 1978).

$\ell_2$-norm regularisation does however have a drawback of not being invariant to linear scaling of the training data (Bishop, 2006). It also does not shrink potentially irrelevant features coefficients to zero, which plays against the portability of the method in the case of a growing high dimension data in many fields and its usefulness in prediction problems where feature selection is an important part, such in biomarker driven models (Ma and Huang, 2008). An alternative is the $\ell_1$-norm regularisation.

### 1.6.2 $\ell_1$-norm regularisation

The $\ell_1$-norm penalty is proposed by Tibshirani (1996), with inspiration from the work of Breiman (1995) and others, and is defined as the minimisation of the following objective:

$$R_{emp}(w) + \lambda \|w\|_1 = \frac{1}{n} \sum_{i=1}^{n} L\left(f_w(x_i), y_i\right) + \lambda \sum_{j=1}^{d} |w_j| \ . \tag{1.9}$$

In the signal processing literature, the lasso is also known as basis pursuit (Chen and Donoho, 1994). In fact, although this penalty based regularisation was originally introduced as a regression method, regularisation by the $\ell_1$-norm has drawn a lot of attention in many communities. Indeed, an important property of the $\ell_1$-norm regularisation is that it can generate an estimation of $w$ with exact zero coefficients, which denotes that the corresponding features are eliminated during the classifier learning process. In other words, the $\ell_1$-norm regularisation encourages sparsity, making it an efficient alternative to subset selection that can be used for feature selection. Computing the optimum of 1.9 is a quadratic programming problem when the risk is quadratic, but efficient algorithms are available with the same computational cost as for ridge regression (Efron et al., 2004). Other theoretical properties have been studied for the regression (Osborne et al., 2000) and also the classification setting, including the logistic loss (Lee et al., 2006). From a bayesian view, $\ell_1$-norm regularisation can also be interpreted as the introduction a prior, this time the *Laplace* distribution, which assigns more weight to regions near zero than the normal prior.

Although the $\ell_1$-norm regularisation has been very popular since its introduction among applied statistical fields (Ma and Huang, 2008), the research around it and especially the design of related variants is still an active research area (Zou and Hastie, 2005; Tibshirani et al., 2015). One reason lies behind some pitfalls proper to this penalty, such as the limited number of selected variables (at most $n$ variables when $d > n$), and more importantly its instability and potentially poorer performance when pairwise correlations between features are very high (Zou and Hastie, 2005; Hansen, 2016). One way to improve the stability and generalisation performance of this method is to use *ensemble learning* as in (Meinshausen and Bühlmann, 2010), which we present in the sext section.

### 1.6.3 Ensemble learning

Ensemble learning is another way to perform explicit regularisation when the model is overfitting due to a high variance.
In general terms, Ensembles are sets of learning machines that combine in some way their decisions, or their learning algorithms, or different views of data, or other specific characteristics to obtain more reliable and more accurate predictions in supervised and unsupervised learning problems (Dietterich, 2000; Kuncheva, 2004). A simple example is represented by the majority vote ensemble, by which the decisions of different learning machines are combined, and the class that receives the majority of "votes" (that is, the class predicted by the majority of the learning machines) is the class predicted by the overall ensemble.
Several theories have been proposed to explain the characteristics and the successful application of ensembles to different application domains. For instance, Breiman (Breiman, 1996c) and Friedman (Friedman and Hall, 2007) interpreted the improved generalisation performance of ensembles of learning machines in the light of the bias-variance analysis, that will be sketched briefly below.

In fact, an extension of decision trees, that we will mention again in the next chapter, is the now popular technique of *random forests* (Breiman, 2001). Random forests construct multiple decision trees from subsets of the training data. Prediction is then done in an *ensemble* by aggregating the predictions of individual trees, as follows:

$$f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x})\,,$$

where the models $f_i$ are individual decision trees trained on random subsets of the data. In such a scheme they are known as *weak learners*.

Now, recall that,

$$\begin{aligned}\mathrm{Var}[X+Y] &= \mathbb{E}[(X+Y)^2] - \mathbb{E}[X+Y]^2 \\ &= \mathbb{E}[X^2 + 2XY + Y^2] - \mathbb{E}X^2 - 2\mathbb{E}[X]\mathbb{E}[Y] - \mathbb{E}[Y]^2 \\ &= \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}[X,Y]\end{aligned}$$

It is easy to then show that:

$$\mathrm{Var}\Big[\sum_i X_i\Big] = \sum_i \mathrm{Var}[X_i] + 2\sum_i \sum_{j\neq i} \mathrm{Cov}(X_i, X_j).$$

Consider then that in our random forest, each learner has a variance of $\sigma^2$ and:

$$\mathrm{Cov}(f_i, f_j) = \rho \;\forall\, i, j;\; i \neq j\,.$$

Then:

$$\mathrm{Var}\Big[\frac{1}{M}\sum_i f_i\Big] = \frac{\sigma^2}{M} + \rho\frac{M-1}{M}\sigma^2 = \frac{1+\rho(M-1)}{M}\sigma^2\,. \tag{1.10}$$

Thus, the ratio is reduced by, $M/(1+\rho(M-1))$. We therefore aim to de-correlate the individual trees. In the Random Forest method, this is done in two ways: First, the training set is sampled (with replacement) to create $M$ *bootstrapped* training sets of size $N$. Secondly, each learner (decision tree) is trained at each step on a random subset of the available features. The size of the subset and the number of trees $M$ are model hyper-parameters.

Ensemble learning is also a generic method that does have its own pitfalls. Indeed, in many settings ensemble learning does not sensitively help generalisation performance (Kuncheva, 2004). It also increases linearly computation time if the procedure is not communicative nor parallelised making its application for complex models such as *deep neural networks* very rare. It also reduces interpretability especially in the case of a complicated aggregation step.

We finally present an implicit regularisation method which builds on the intuition of improving generalisation by providing more training samples: data augmentation.

### 1.6.4 Data Augmentation

If the data set used for training is not large enough, which is often the case for many real life test sets, then it can lead to overfitting. A simple technique to get around this problem is by artificially expanding the training set.

In the case of image data, this fairly straightforward for example we should be able to subject an image to the following transformation without changing its classification (see figure 1.8):

- Translation: moving the image by a few pixels in various directions
- Rotation
- Reflection
- Skewing

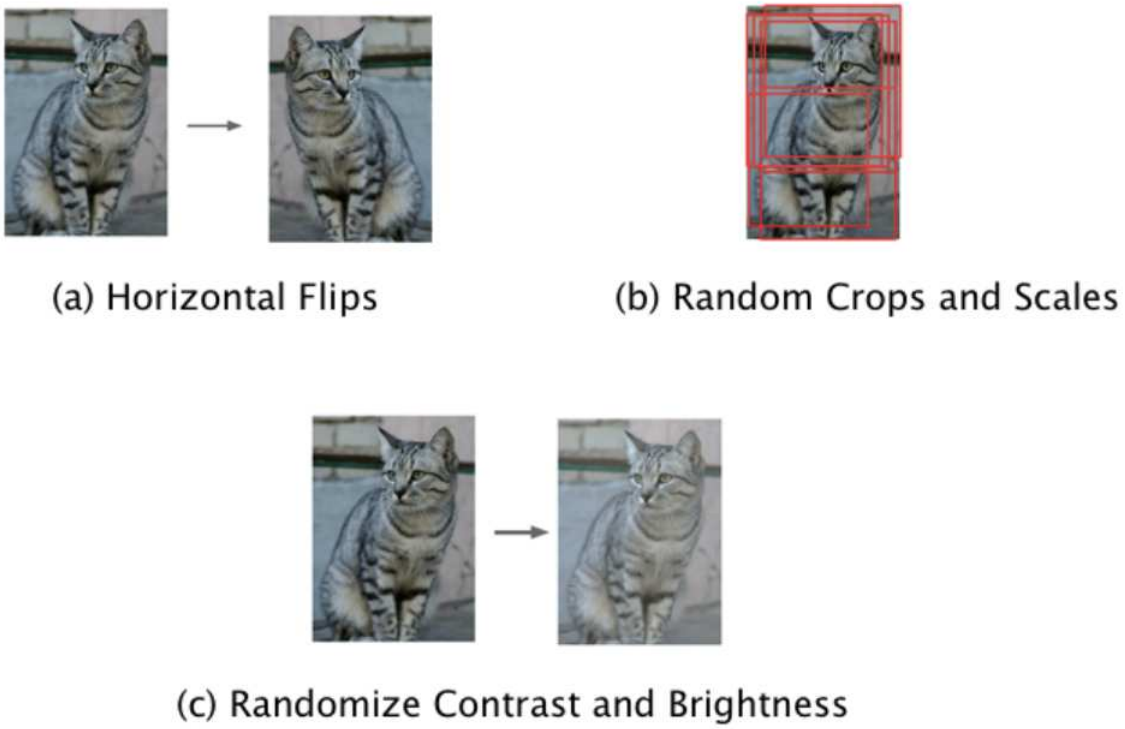



Figure 1.8: Some examples of data augmentation for a particular image.

- Scaling: changing the size of the image while preserving its shape
- Changing contrast or brightness

Data augmentation is an almost ubiquitous technique in neural networks training for instance, especially for computer vision tasks, which can be regarded as an implicit regulariser because it improves regularisation without directly constraining the model. Introduced in different forms and for different models and datasets in the late 80's and early 90's (Simard et al., 1992; Sung and Poggio, 1994; Schölkopf et al., 1996), it has been identified as a very important element until today (Wang and Perez, 2017) and in many modern successful models, like AlexNet (Krizhevsky et al., 2014), All-CNN (Springenberg et al., 2014) or ResNet (He et al., 2016), for instance. In domains other than computer vision, data augmentation has also been proven effective, for example in speech recognition (Jaitly and Hinton, 2013), music source separation (Uhlich et al., 2017) or text classification (Xu et al., 2016).

In addition to the traditional data augmentations utilising a particular transformation to directly synthesise derived images with the same label, a recent approach is to use generative models (such as Generative Adversarial Networks (Goodfellow et al., 2014)) in order to artificially augment the data (Mikołajczyk and Grochowski, 2018). For both approaches, implementations fall under two categories: data augmentation before training of the prediction model, and data augmentation while training, mostly through Monte Carlo Markov Chains (MCMC) and Expectation Maximisation (EM) methods (Van Dyk and Meng, 2001). All these approaches can both be extremely slow to converge (Van Dyk and Meng, 2001) and some schemes have already been developed for accelerating these implementations (Meng and Van Dyk, 1999; Liu and Wu, 1999) but these are not always possible to apply. Besides, even though some authors have reported the impact of data augmentation on the performance of their models and, in some cases, a comparison of different amount of augmentation (Graham, 2014) and effect, the literature lacks, to our knowledge, a systematic analysis of the impact of data augmentation on generalisation. Last but not least, a pitfall of data augmentation is that, as it is applied, seems to be an ad hoc procedure dependent on the data at hand and

the task at hand, which in contrast with the previous regularisation methods, is flexible but requires the expert prior knowledge about the problem and can be difficult to select and to reproduce (Ratner et al., 2017).

### 1.6.5 Bridging the gap: towards data-dependent regularisation

We presented through the last sections different regularisation methods, relying on different assumptions, either explicitly or implicitly. Explicit regularisations such as adding a penalty to the ERM problem are well studied and different efficient implementations have been developed (Sra et al., 2012). Besides individual limitations, we have seen that these methods enforce properties such that smoothness or sparsity that are often not coming from a prior knowledge but rather generic assumptions that might be later useful such as feature selection (Bishop, 2006). Many approaches have been designed to complement these methods with existing prior knowledge about the data, the model or the task at hand (Lauer and Bloch, 2008; Huang et al., 2011b). Structured sparsity-enforcing regularisation methods, for instance, allow to additionally incorporate particular prior assumptions on the structure of the input variables, such as overlapping groups, non-overlapping groups and acyclic graphs (Yuan and Lin, 2006; Obozinski et al., 2011). Examples of uses of structured sparsity methods include face recognition (Jia et al., 2012), magnetic resonance image (MRI) processing (Chen and Huang, 2012), and analysis of genetic expression in breast cancer (Jacob et al., 2009) among other applications.

Despite these several successful applications, and maybe also because of it, the proliferation and diversity of these particularly designed explicit regularisation methods, exploiting a particular prior knowledge, can be sometimes confusing. Moreover, the effect of the addition of the same prior knowledge in the learning problem depends drastically on the training dataset and the particular regularisation method (Lavi et al., 2012).

On the other hand, considered implicit regularisation methods such as data augmentation has also, despite being a classical technique, witnessed a growing popularity in recent years. As it was already described, in this case there is a clear lack of generic methods for augmenting the data, although interesting research that aims to automatically learn useful data transformations has been prublished very recently (Lemley et al., 2017; Cubuk et al., 2018).

In all cases, bridging the gap between explicit generic and implicit data-dependent regularisation appears to be a very promising direction both in methodology and in theoretical understanding. In fact, recent and most popular regularisation methods in training of deep neural networks such as *dropout* (Srivastava et al., 2014), *batch normalisation* (Ioffe and Szegedy, 2015) or *Cutout* (DeVries and Taylor, 2017) provide state-of-the-art generalisation results by applying particular deterministic or stochastic transformations to the data or the hidden layers' input (see (Kukačka et al., 2017) for a complete survey). Interestingly, the analysis of these different methods, in spite of a lack of theoretical understanding of their efficiency (Baldi and Sadowski, 2013), have started still to reveal interesting dualities between data transformation and model regularisation (Wager et al., 2013; Bouthillier et al., 2015), that might lead to a better understanding of generalisation properties given some particular data properties and lead to a much awaited unification in regularisation theory beyond the Vapnik-Chervonenkis dimension theory (Abu-Mostafa, 1989).

## 1.7 Thesis and contributions

This thesis explores a family of regularisation methods in the context of supervised learning that are based on *Injecting Noise on the Input data* that we denote by (INI). We will first give an overview of a real supervised learning problem in the context of a competitive framework, exhibiting the risks of learning from high-dimension bioinformatics data and corresponding

vulnerabilities to overfitting. We then present, review and study (INI) theoretically and empirically: its regularisation properties and efficiency. We then describe novel generalisations of (INI) methods and particularly the *dropout* termed as *DropLasso*, *Structured Noise Injection* (SNI) and *Adaptive Structured Noise Injection* (ASNI) that we design and study in the last two chapters.

### 1.7.1 The Rheumatoid Arthritis challenge

As a first real-data application of supervised learning, I participated among the team *Outliers* in the design of predictive models of the response to treatment of patients with Rheumatoid Arthritis. Rheumatoid Arthritis is a chronic autoimmune disease that causes the inflammation of joints. The work described in this chapter was held in the context of the Rheumatoid Arthritis Responder Challenge as a comparative evaluation framework. Our team obtained the best performance results for the clinical and the genetic models, thanks to different strategies to overcome overfitting. We were then invited to take part in the collaborative phase of the challenge, where we focused on the following question: can the addition of genetic data improve the prediction made based only on simple clinical covariates?

Part of the work presented in this chapter has been published in (Sieberts et al., 2016).

### 1.7.2 (INI) framework and a new approximation of dropout

The idea of noise injection in the data from which we want to learn seems at first sight quite counterintuitive. However, the idea of adding randomness in algorithms has been present for decades in the community of computer science and has started to emerge early in the machine learning community with the development of optimisation algorithms for neural networks three decades ago. This technique has recently recaptured the community interest with the regain of popularity of multilayer neural networks and their increasing complexity that emphasises the need of their regularisation. Despite this revival of research interest around noise injection variants and applications, there is still a lack of clear definition for this set of methods and their interpretations from different frameworks views. In this chapter:

1. We provide an overview of Noise Injection in supervised learning from the point of view of different supervised learning settings.

2. We reformulate the Input Noise Injection framework in the supervised learning setting with general distributions and noising functions.

3. We summarise intuitions, algorithms and a part of theoretical justifications (in linear models) around the use of (INI).

4. We present a brief overview of the popular *dropout* method and related works.

5. We provide a novel approximation for *dropout* leading to new insights for linear and non-linear models and with general (potentially non-smooth) loss functions.

6. We complement this chapter by a series of experiments about the effectiveness of (INI) in improving the generalisation performance of linear models in the supervised learning setting. These experiments are performed on simulations and benchmark datasets in vision recognition, document classification and cancer prognosis tasks.

### 1.7.3 DropLasso: a robust variant of Lasso for single cell RNA-seq data

During the last decade, high-throughput sequencing methods have revolutionised the entire field of biology, as we have briefly noticed. Thanks to recent technical improvement, it is possible today to obtain genome-wide transcriptome data from single cells using high-throughput sequencing (scRNA-seq). The main advantage of scRNA-seq is that the cellular

resolution and the genome wide scope makes it possible to address issues that are intractable using other methods, e.g. bulk RNA-seq. However, to analyse and build models using single cell RNA-seq data, novel methods are required as some of the underlying assumptions for the methods developed for bulk RNA-seq experiments are no longer valid. The technology suffers for instance from stochastic under-amplification of random regions which results in what is called *dropout errors* (Kharchenko et al., 2014).

In this context, we propose a new method called DropLasso to classify, but also learn a molecular signature from single cell RNA-seq data. DropLasso is an extension to the dropout regularisation technique, popular in neural network training, that embeds feature selection through its fusion with the $\ell_1$-norm regularisation. We explain how it can be suited to the particular setting of single cell RNA-seq data, and we theoretically provide insights about its effect and how it can be related to a data-dependent modified version of elastic net in the least squares and in the general case. We provide comparative classification results on simulated and real scRNA-seq data, suggesting that DropLasso may be better adapted than standard regularisations for supervised learning from this type of data, and how it can lead to the novel discovery of potentially interesting biomarkers.
DropLasso is freely available as an R package we developed, at `https://github.com/jpvert/droplasso`.

### 1.7.4 Adaptive Structured Noise Injection (ASNI)

In recent years, deep learning has been tremendously successful in many important applications of machine learning, such as image classification and object recognition among other applications (Krizhevsky et al., 2017). Bengio et al. (2013) argue that the success of deep learning is in part due to the capability of neural networks to build incrementally better representations that expose the relevant variability, while at the same time discarding nuisances. The interpretation of deep neural networks as a way of creating successively better representations of the data has already been suggested and explored by many. Most recently Tishby and Zaslavsky (2015) put forth an interpretation of deep neural networks as creating sufficient representations of the data that are increasingly minimal.
During the last year of my thesis, an interesting related observation appeared when I was experimenting with multilayer networks training with and without dropout or other regularisation schemes. Figure 1.9 illustrates this phenomenon, which mainly resides in the striking synchronicity between the evolution of the test accuracy of the network and that of the correlations strength between the units activations of the hidden layers (as evaluated by the Frobenius norm of their correlation matrix as will be detailed later in the corresponding chapter). This phenomenon was general in that it was happening in all networks I tried (including or not convolutional layers), in most datasets and with or without dropout.

One may argue that this phenomenon is simply due to the fact that at the first stage, the network is learning useful representations which results in an increase of accuracy, or maybe that in the contrary learning useful information disentangles automatically the representations and results in more useful ones. This interesting observation in all cases conveys a series of questions and research directions: Can one improve accuracy by disentangling more the hidden layer units, e.g. reducing the correlations between their activations? Does dropout improve generalisation accuracy by actually reducing the correlations between units? Is there a way to make a version of an *aggressive* or rather a *selective* form of dropout, or noise injection in general, that selects somehow correlated units and prevents their co-adaptations?

In fact, this idea to enforce the choice a minimal set of representations is not new in statistical learning and even prior to its development. The idea of diminishing redundancy in representations can indeed be traced back to early investigations about learning in general. For instance, in 1961 Barlow (1959) already emitted three hypotheses about the operational reception and the response to stimuli information by the nervous system, the third of which theorises that ”reduction of redundancy is an important principle guiding the organisation

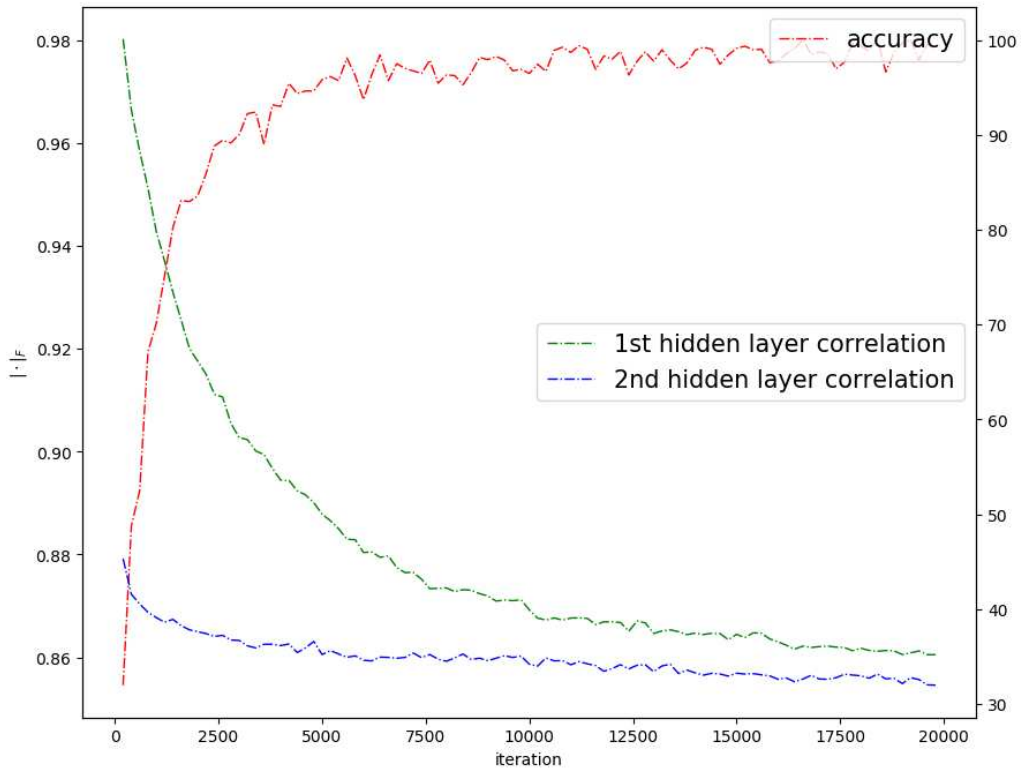



Figure 1.9: Evolution of test classification accuracy during the training of a 2 hidden layers feed forward neural network (see figure 1.4) on MNIST, without any regularisation (red: Left scale), *versus* the correlation matrix norm of the first and second hidden layer activations (blue and green: Right scale)

of sensory messages and is carried out at relays in the sensory pathways". Since then, this idea has been carried out not only in the understanding of learning efficiency, but also in the design of popular methods in various areas of machine learning such as component analysis like ICA (Hyvärinen, 2013) or feature selection (Peng et al., 2005). The idea of penalising representations correlations while learning also naturally emerged as a way to avoid overfitting and thus improving the performance of predictive models in supervised learning. In the context of ensemble learning, diversity is a good feature that improves the ensemble performance when traded with each model bias (Kuncheva and Whitaker, 2003; Dietterich, 2000). In many deep neural network models parameters show a significant amount of redundancy (Denil et al., 2013). Several strategies have been developed to implement this diversity in the training of one or several neural networks such as pruning units or connections (Hassibi and Stork, 1993; LeCun et al., 1990; Mariet and Sra, 2016). However these techniques have mostly been concerned with networks memory footprints and speed, and less with their accuracy and representations quality. Other techniques have recently used the same concept in order to train the same network architecture while encouraging diversity between units of each layer while learning in order to improve generalisation performance (Cogswell et al., 2015; Desjardins et al., 2015; Luo, 2017).

In this chapter, we first show that dropout and its continuous multiplicative gaussian noise do not necessarily reduce correlations between units, and we introduce a novel form of Adaptive Structured Noise Injection (ASNI) as a generalisation of the independent noise injection scheme, where the structure of the noise applied on one layer follows the covariance of that layer units' activations. We theoretically study the effect of ASNI and empirically show that (ASNI) is more efficient than Independent noise injection such as dropout in preventing overfitting in multilayer dense and convolutional neural networks, and that it succeeds to disentangle hidden representations and improve their quality.

# Chapter 2

# The RA responder challenge

"*In theory, there is no difference between theory and practice. In practice, there is.*"

Yogi Berra

Contents

**Abstract**

Rheumatoid arthritis (RA) affects millions world-wide. While anti-TNF treatment is widely used to reduce disease progression, treatment fails in almost one-third of patients. A rigorous community-based assessment of the utility of SNP (single-nucleotide polymorphisms) data for predicting anti-TNF treatment efficacy in RA patients was performed in the context of a DREAM Challenge http://www.synapse.org/RA_Challenge. An open challenge framework enabled the comparative evaluation of predictions developed by different research groups using the most comprehensive available data and covering a wide range of modelling methodologies.

Among 73 teams, our team participation under the name of "Outliers" was rewarded with the best model prediction results. We do believe that our model resisted best to overfitting thanks to a residual fitting strategy, a biologically relevant feature selection and a careful cross-validation. Finally, the challenge conclusion was that the teams models "formally confirm the expectations of the rheumatology community that SNP information does not significantly improve predictive performance relative to standard clinical traits, thereby justifying a refocusing of future efforts on collection of other data".

This chapter is largely based on our team work from May 2014 to November 2014, and the challenge findings reported in *Nature Communications* in 2016 (Sieberts et al., 2016).

**Résumé**

La polyarthrite rhumatoïde (PR) touche des millions de personnes dans le monde. Bien que le traitement anti-TNF soit largement utilisé pour réduire la progression de la maladie, le traitement échoue chez près du tiers des patients. Une évaluation rigoureuse de l'utilité des données de polymorphisme mono-nucléotidiques (SNP pour prédire l'efficacité du traitement anti-TNF chez les patients atteints de PR a été réalisée par un ensemble d'équipes dans le contexte du RA DREAM Challenge http://www.synapse.org/RA_Challenge auquel j'ai participé dans le sein de l'équipe "Outliers".

Parmi les 73 équipes, notre participation a été récompensée par les meilleurs résultats de prédiction finale. Nous pensons que notre modèle a mieux résisté au sur-apprentissage grâce à une stratégie d'ajustement résiduel, à la sélection de variables biologiquement pertinents et à une validation croisée minutieuse. Enfin, une conclusion principale du Challenge était que les modèles dans l'ensemble confirmaient formellement les attentes de la communauté des rhumatologues selon lesquelles les informations du SNP n'amélioraient pas de manière significative la performance prédictive par rapport aux traits cliniques standard, ce qui justifierait un recentrage des efforts futurs sur la collecte d'autres données.

Ce chapitre repose sur les travaux de notre équipe de Mai 2014 à Novembre 2014, et les résultats de la compétition peuvent être consultés depuis 2016 (Sieberts et al., 2016).

## 2.1 Introduction

### 2.1.1 Rheumatoid Arthritis

The first recognised description of RA was made in 1800 by Dr. Augustin Jacob Landre-Beauvais (1772 to 1840) in Paris. Rheumatoid Arthritis is now recognised as an autoimmune disease that results in a chronic, systemic inflammatory disorder that may affect many tissues and organs, but principally attacks flexible synovial joints (Gibofsky, 2012). It can be a disabling and painful condition, which can lead to substantial loss of functioning and mobility if not adequately treated.

The process involves an inflammatory response of the capsule around the joints synovium secondary to swelling of synovial cells, excess synovial fluid, and the development of fibrous tissue in the synovium. The pathology of the disease process often leads to the destruction of articular cartilage and fusion of the joints, as shown in figure 2.1. Rheumatoid Arthritis can also produce diffuse inflammation in the lungs, the membrane around the heart, the membranes of the lung, and white of the eye (sclera), and also nodular lesions, most common in subcutaneous tissue.

About 0.6% of the United States adult population has RA, women two to three times as often as men. Onset is most frequent during middle age, but people of any age can be affected.

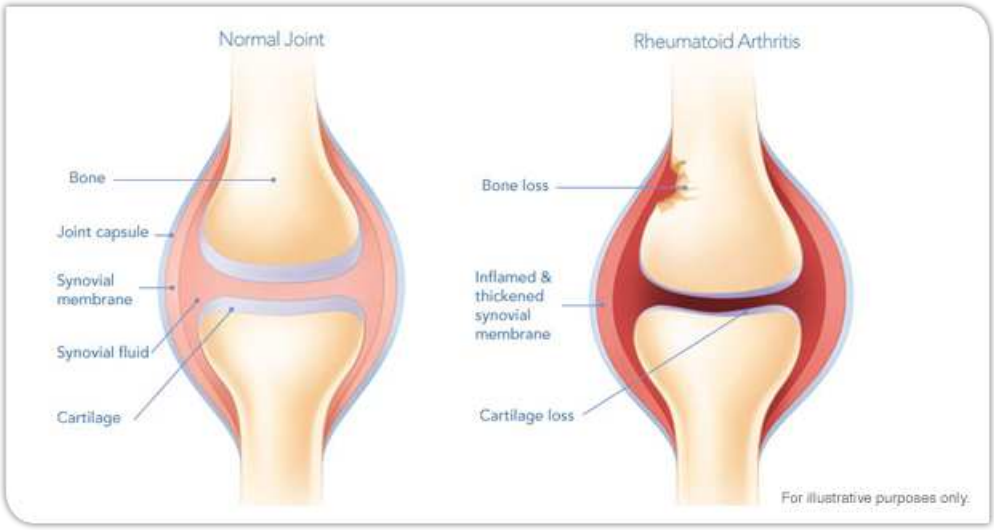



Figure 2.1: Normal joint *versus* Rheumatoid Arthritis affected joint.

Although the cause of RA is unknown, autoimmunity plays a big part, and RA is a systemic autoimmune disease. It is a clinical diagnosis made on the basis of symptoms, physical exam, radiographs (X-rays) and labs. Treatments are pharmacological and non-pharmacological. Non-pharmacological treatment includes physical therapy, orthoses, occupational therapy and nutritional therapy but these don't stop the progression of joint destruction. Analgesics, suppress symptoms, but don't stop the progression of joint destruction either.

RA is treated in part with disease-modifying anti-rheumatic drugs, including those that block the inflammatory cytokine, tumour necrosis factor-$\alpha$ (anti-TNF therapy). While anti-TNF treatment is effective in reducing disease progression, response is variable with nearly one-third of RA patients failing to enter clinical remission (McInnes and Schett, 2011; Wijbrandts et al., 2008). Given the expense of these drugs (around 5000 to 10,000 GBP per patient per year in the UK) and the potential for detriment to non-responding patients, the identification of predictors of response from pre-treatment (baseline) characteristics would be of great clinical, societal and economic benefit. The importance of this question motivated the

formation of the UK Maximising Therapeutic Utility in RA (MATURA) consortium (Barton and Pitzalis, 2016), which has the wider remit of using blood-based biomarkers and synovial pathobiology to inform the stratification of all stages of RA treatment. As for today, no substantive methodology exists that can be used to a priori identify anti-TNF non-responders (Tak, 2011). In this context, this RA challenge [1] as part of DREAM challenges, was organised, in order to provide a framework for evaluating different genetic and clinical models predictive of the anti-TNF response and capable of identifying potential good and poor responders, using the most comprehensive available data to date.

### 2.1.2 GWAS and RA

An important part of the collected data concerned patients genotypes and more precisely particular biological individual features: called SNPs. A body of literature has already explored in fact the relationship between these features and the occurrence of RA. This was an initial motivation for the overall project but also a way for our team to gather some prior knowledge and potentially improve our models through the use of the existing literature.

Single nucleotide polymorphisms, frequently called SNPs (pronounced "snips"), are the most common type of genetic variation among people. Each SNP represents a difference in a single DNA building block, called a nucleotide. For example, a SNP may replace the nucleotide cytosine (C) with the nucleotide thymine (T) in a certain stretch of DNA.

SNPs occur normally throughout a person's DNA. They occur once in every 300 nucleotides on average, which means there are roughly 10 million SNPs in the human genome. Most commonly, these variations are found in the DNA between genes. They can act as biological markers, helping scientists locate genes that are associated with disease. When SNPs occur within a gene or in a regulatory region near a gene, they may play a more direct role in disease by affecting the gene's function.

Most SNPs have no effect on health or development. Some of these genetic differences, however, have proven to be very important in the study of human health (Visscher et al., 2012). Researchers have found SNPs that may help predict an individual's response to certain drugs, susceptibility to environmental factors such as toxins, and risk of developing particular diseases. SNPs can also be used to track the inheritance of disease genes within families. Today, Genome-wide association studies (GWAS) are focusing on identifying SNPs associated with complex diseases such as heart disease, diabetes, and cancer. In fact, the development of relatively cheap SNPs arrays, which allow to genotype an individual for a predefined set of loci, has been crucial to the onset of GWASs. These arrays, to date, typically contain from 200,000 to 2,000,000 SNPs. The first large GWAS, involving 14,000 individual with one of seven common diseases such as type 1 and type 2 diabetes, as well as 3,000 control individuals, dates back to 2007 (Burton et al., 2007). Since then, many studies have been conducted for thousands of complex traits and over 10,000 associations have been reported (Welter et al., 2013). The purposes of GWAS, beyond the search for associated loci, are multiple. Primarily, these studies are aimed at facilitating the identification of the underpinnings of complex diseases and ultimately driving translational advances. In the last decade, GWASs have successfully facilitated the discovery of biological mechanisms involved in several diseases (see (Visscher et al., 2017) for a review). One famous example is the discovery, through GWAS, of a SNP within the *Complement Factor H* (CMH) gene that conveys a significant increased risk in developing age-related macular degeneration (AMD) (Klein et al., 2005). The biological insight gained through this discovery has fuelled the development of a number of therapeutics that are today at preclinical or clinical stages (see (Black and Clark, 2016) for a review). Nonetheless, GWAS are today facing criticisms regarding its primary purpose. These criticisms notably point the difficulty to go from GWAS results to the identification of causal SNPs, and the fact that the vast majority of the discovered associations have small

[1] http://www.synapse.org/RA_Challenge

effects, i.e, correspond to small increased risk to develop a disease.

Following this trend, and due to the importance of research around Rheumatoid Arthritis and the development of genomic techniques, the last 60 years have seen mounting evidence to support the genetic basis of RA through the identification of genetic susceptibility variants. Prior to the development of GWAS, RA susceptibility loci were in fact discovered through candidate gene and linkage studies. Although these approaches led to the discovery of only a small number of loci, they identified the two most significant RA risk loci that remain to date: human leukocyte antigen (HLA- DRB1) and protein tyrosine phosphatase, non-receptor type 22 (PTPN22). Together HLA-DRB1 and PTP 22 are estimated to account for approximately 40% of the total genetic risk for RA (Orozco et al., 2006). One of the latest GWAS meta-analysis association study has brought the total number of confirmed RA risk loci to 34 (Stahl et al., 2010).

A comprehensive discussion of all the identified genetic loci is beyond the scope and the aim of this introduction. However, and in the case of our team, studying and using the biological literature about RA and including this prior knowledge in our model is in our mind one important factor of preventing overfitting in our final model, as it will be later discussed. Besides, these studies were an important part of the initial motivation of the challenge, since the evidence from association analyses (Cui et al., 2013),(Stahl et al., 2010) and the theoretical heritability estimates suggested that algorithms focusing on genetic variation may be predictive of patients response to RA therapies. This could be extremely useful for further medical research, as biomarkers provide a compelling opportunity to perform simple tests with high-potential impact on clinical care.

## 2.2 The challenge

### 2.2.1 Challenge design

The challenge was held over two phases: A two-step competitive phase and a collaborative phase. The training data for the first part of the competitive phase consisted of a previously published collection of anti-TNF-treated patients ($n = 2,706$) of European ancestry, compiled from 13 collections (Cui et al., 2013), of which the response variables from 675 patients were held-out as a leaderboard test set. All participants, including our team, were indeed provided with a leaderboard with real-time feedback, which evaluated the performance of their predictions in the remaining 675 individuals. To reduce the potential for overfitting or reverse-engineering of treatment outcomes from the leaderboard, each team was limited to 100 leaderboard submissions.

Over the course of the 16-week training period, 73 teams submitted a total of 4,874 predictions for evaluation on the leaderboard data. Upon completion of the training period, teams were allowed up to two final submissions per sub-challenge and final evaluation of algorithms was performed relative to a separate test data set consisting of data collected from 591 RA patients in the *Corrona certain* study (Pappas et al., 2014). Comparison with an independent, blinded test data set aimed at reducing the contribution to estimated accuracy of overfitting to the training data set (an effect that will be later investigated). The goal of this competitive phase was primarily is to identify the best genetic and clinical models accuracies in predicting patients response, and potential interesting biomarkers.

The 8 best-performing teams were invited to join the collaborative phase. In this phase, a collectively designed experiment was developed, in which each team independently performed analyses and challenge organisers performed a combined analysis. We will focus here on our methodology for the competitive phase, although we will also report final findings that were agreed on after the end of the collaborative phase. Figure 2.2 shows the mentioned details of the two phases and data used for the competitive phase.

The challenge contained two sub-challenges:

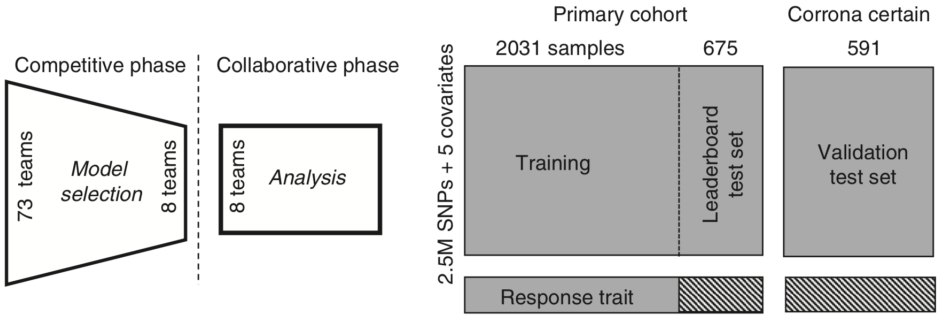



Figure 2.2: Challenge double-phase design (left figure) and corresponding data for the competitive phase (right figure)

- Predict treatment response as measured by the change in disease activity score (DAS28) in response to anti-TNF therapy
- Identify poor responders as defined by EULAR criteria for non-response (20% of the study population).

Since an estimate response quality defined in the second sub-challenge can be immediately deduced from the treatment response prediction of the first sub-challenge (see 2.2.2), we therefore focus on our regression method, although we report both sub-challenge methods and results. Interestingly, predicting independently the binary label: response/non-response did not improve the classification results in our case.

### 2.2.2 Challenge data

As mentioned, two separate data sets were provided to participants to train and test the predictive models at two phases of the challenge. Anti-TNF response differed slightly between the training and test data sets (21.7% and 35.7%, respectively), likely due to differences in inclusion criteria in the two cohorts, although demographic data were similar between the two. All teams remained blinded to outcomes from both the leaderboard and test data sets throughout the experiment. Harmonised data from all cohorts are now publicly available as a resource for use by the research community (doi:10.7303/syn3280809).

#### Genetic variables

The genetic variables consisted of processed SNP measurements. Processing was performed prior to the challenge, and consisted of quality control filtering and SNP imputation. The training set, derived from a meta-analysis containing 13 collections, was measured in 11 batches which were processed separately. The validation set was also measured and processed separately. Quality control filtering was performed within each batch. Individuals with $> 5\%$ missing data and SNPs with $> 1\%$ missing data were removed. SNPs which did not satisfy the Hardy-Weinberg equilibrium [2] or had a minor allele frequency (MAF) $< 1\%$ were also removed. While related individuals in the training set were removed, the single pair of related individuals found in the validation set was kept. Finally, while ethnicity outliers of non-European descent were removed from the training set, the validation set contained 93 patients of non-European descent which were not removed, but were not used for model scoring.

The measurement technology used within each batch was the same, but a variety of different technologies were used across batches. Each technology measured a different set of SNPs which resulted in only 20,000 SNPs which were measured across all patients. Since

[2] A principle stating that the genetic variation in a population will remain constant from one generation to the next in the absence of disturbing factors

the number of SNPs common to all technologies was small relative to the total number of SNPs, the missing SNP values were imputed. Each batch was imputed separately to a common European reference. Imputation resulted in estimates of $\sim$ 2.2 million SNPs for all individuals. Since the training set contained 104 SNPs that were not contained in the validation set, we did not use these SNPs to build models.

After imputation, two SNP representations were provided to challenge participants. The first representation was the imputed dosage, which is the estimated counts of the reference nucleotide. Since each individual has two copies of each DNA strand, the reference value could appear 0, 1 or 2 times and the imputation estimates could range from 0 to 2. The second representation was the genotype probabilities, which estimates the probability of having 0, 1 or 2 of the reference nucleotide. While the GWAS meta-analysis performed associations using the imputed dosages under the assumption of an additive genetic model, we chose to use the genotype probability representation because it would allow us to estimate genotype effects without requiring us to impose a functional relationship between each SNPs three genotype effects.

### Clinical variables

The following clinical variables were collected and could be used to predict drug response: age, sex, anti-TNF therapy, methotrexate use and baselineDAS. Patients in the training set were treated with one of three anti-TNF drugs: adalimumab, etanercept or infliximab. While most patients in the validation set were treated with the same drugs in the training set, some patients were treated with either golimumab or certolizumab. When evaluating the models on the validation set, the challenge organisers excluded the golimumab treated patients because participants were unable to successfully predict their drug response and included the certolizumab treated patients because certolizumab predictions had a similar performance to the three drugs observed in the training set.

### Clinical outcomes

The disease activity score DAS28 (which we will refer to as DAS) is a continuous measure of disease severity in rheumatoid arthritis. It is a non-negative score, with a higher value corresponding to a more severe disease status. The score is calculated using the number of swollen and tender joints (out of the 28 joints which are examined), the amount of inflammation markers observed in a blood sample and a score of how well a patient feels. Each patient had DAS measured before (baselineDAS) and after (endDAS) starting treatment. While all patients in the training set had endDAS measured 3-12 months after starting treatment, patients in the validation set all had endDAS measured 3 months after starting treatment. The continuous outcome used to evaluate a patient's response was the change in the disease activity score: $\Delta\text{DAS} = \text{baselineDAS} - \text{endDAS}$.

The decrease in disease severity was chosen to reflect the change in disease activity so that positive values of $\Delta$DAS corresponded to an improvement in disease status, which was the desired response for patients. A second drug response outcome was a binary responder status determined by the European League Against Rheumatism (EULAR) response criteria. The EULAR criteria uses different $\Delta$DAS thresholds to define the type of response based on the final disease activity of the patient, as shown in table 2.1. Since endDAS is a function of $\Delta$DAS and baselineDAS, the EULAR criteria can also be interpreted as selecting $\Delta$DAS thresholds as a function of initial disease activity. For the challenge, a responder was defined as a patient that had either good or moderate response, while a non-responder had no response.

### Other information

Two ancillary data sets were made also available for participant use. The first measured TNF-$\alpha$ protein level in HapMap cell lines (Choy et al., 2008). The second included blood RNA-seq

| **endDAS \ΔDAS** | >1.2 | 1.2 >... >0.6 | <0.6 |
|---|---|---|---|
| <3.2 | good response | moderate response | no response |
| 3.2 <... <5.1 | moderate response | moderate response | no response |
| >5.1 | moderate response | no response | no response |

Table 2.1: EULAR response criteria.

data and genotypes for 60 RA patients from the Arthritis Foundation-sponsored Arthritis Internet Registry, 30 of whom displayed high inflammatory levels and 30 of whom displayed low inflammatory levels. Inflammatory levels were assessed using blood concentrations of C-reactive protein (CRP), and elevated disease was defined as CRP $> 0.8$mg dl$^{-1}$, while low disease activity was defined as CRP$< 0.1$mg dl$^{-1}$. In addition to CRP levels, rheumatoid factor antibody levels and cyclic citrullinated peptide levels were also assayed. Genotypes were assayed on the Illumina HumanOmniExpressExome array. We didn't make use of these datasets for our best model. And at our knowledge, no team had.

## 2.3 Methods

### 2.3.1 Data processing

#### Feature selection

Since each patient had more than 2 million SNP measurements and the training set only contained 2,031 patients, the set of SNPs had to be reduced to ensure that non-spurious relationships between SNPs and drug response could be identified. We selected a subset of 160 SNPs from two sources:

1. First, we selected 105 SNPs previously associated with drug response in Rheumatoid Arthritis from the Pharmacogenomics Knowledgebase (Whirl-Carrillo et al., 2012). In PharmGKB. We found 110 SNPs associated with the following drugs: adalimumab, etanercept, infliximab, methotrexate and Tumor Necrosis Factor $\alpha$ (TNF-$\alpha$) inhibitors. Since 14 of these SNPs were not available in the challenge data, we used the HaploReg [3] database to substitute each of these SNPs with the most highly correlated SNP that was contained in the data set (Ward and Kellis, 2011). Of the 14 SNPs, we substituted 9 SNPs with their best replacement and removed 5 SNPs because 2 SNPs had replacements that were already contained in our PharmGKB SNP list and 3 SNPs did not have a replacement with a correlation of at least $r^2 > 0.5$ in the HaploReg database.

2. Additionally, we selected 55 SNPs that were significantly correlated with TNF gene expression in blood. These SNPs were collected by the Linked Open Data challenge team from the Blood eQTL Browser (Westra et al., 2013) using a p-value threshold of $10^{-10}$.

**Remark 1.** *This feature selection might seem quite restrictive. We tried indeed several strategies at the beginning with a gene-level models with thousands of features. These initial models did overfit so severely that we rapidly decided to use biologically based feature selection, also to enhance interpretability, central to the goals of challenge.*

#### Imputation

For clinical variables, we imputed all missing values with their mean value across patients. Imputation was performed separately on the training and validation sets. Categorical variables were transformed into indicator variables before being imputed. Patients in the validation

[3] https://pubs.broadinstitute.org/mammals/haploreg/haploreg.php

set who were treated with golimumab or certolizumab were treated as if their drug value was missing.

### 2.3.2 Pipeline

We chose to adjust the target variable ΔDAS using the clinical variable baselineDAS for several reasons. Since ΔDAS was defined as a function of baselineDAS (ΔDAS = baselineDAS - endDAS), modeling the relationship between baselineDAS and endDAS would force the model to explain the relationship between baselineDAS and ΔDAS which could not simply be explained by ΔDAS's definition. Modeling the relationship between baselineDAS and endDAS also allows us to incorporate natural constraints on endDAS: First, endDAS is defined to be non-negative. Second, the mean of endDAS increases as baselineDAS increases because patients with more severe initial disease activity are more likely to have more severe final disease activity. Third, the variance of endDAS increases as baselineDAS increases because patients with more severe initial disease activity have a larger range of possible improvements that they can achieve. Finally, endDAS is skewed towards 0 for fixed values of baselineDAS because most patients respond positively to the drugs, resulting in a larger decrease in their disease activities than the smaller proportion of patients who do not respond to the drugs. Figure 2.3 partially shows these constraints.

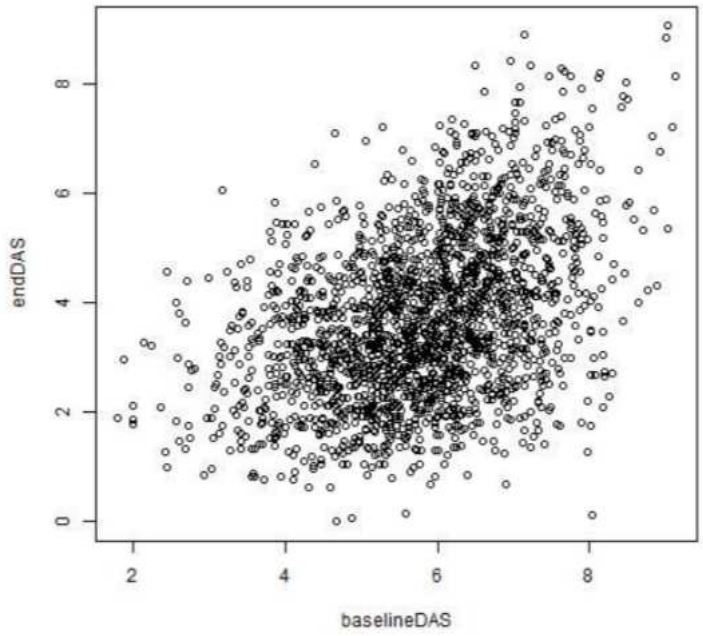



Figure 2.3: Plot of baselineDAS versus endDAS on the challenge training data

Given these observations, it was not appropriate to use a simple linear regression to model the relationship between baselineDAS and endDAS because the residuals would have been bounded, skewed and heteroscedastic. Instead, we used a gamma distributed generalised linear model (GLM) with log link (Nelder and Wedderburn, 1972), to represent endDAS's dependence on baselineDAS. Since the Gamma distribution is non-negative, able to represent skewed distributions and its variance increases as its mean increases (Thom, 1958), it is a much preferred tool here to respect the mentioned constraints.

After predicting the mean endDAS given baselineDAS, we observe that the residual errors have an increasing variance (see left figure in 2.4 ) Since the Gamma distributions fit with the model has a constant coefficient of variation, the residuals of this model were divided by their corresponding endDAS predictions so that their variances were no longer a function of baselineDAS (see figure 2.4, right). After removing the dependence of the residuals on the initial disease activity, we used an $\ell_1$-norm regularised linear model of the SNPs and

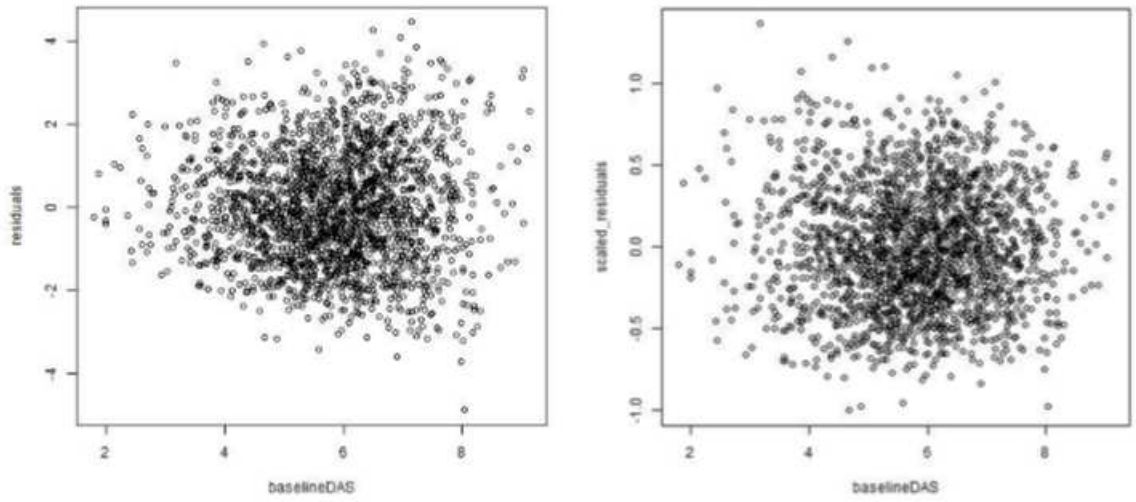



Figure 2.4: Plots of baselineDAS versus residuals (left figure) and scaled residuals (right figure)

the remaining clinical variables to predict the scaled residuals. We used a linear model to predict the residuals because a linear model is interpretable and would allow us to identify SNPs and clinical variables which were associated with the outcome. A single model was used for all drugs because drug specific models would be fit with a third of the patient samples and would not generalise to drugs not observed in the training set (certolizumab, golimumab). Since all of the drugs target the same pathway, fitting a single model for all drugs should improve our ability to identify SNPs which affect drug response through this common pathway. Regularisation penalties were only placed on the SNP coefficients because we expected only a small number of SNPs to be useful predictors based on the lack of associations discovered in the GWAS meta-analysis. These penalties were selected using 5-fold cross validation. The scaled residual predictions were converted to their original scale by multiplying them with the GLM endDAS predictions. The rescaled predictions were added to the GLM endDAS predictions to get the final endDAS predictions. Since endDAS must be non-negative, all negative final endDAS predictions were set to 0. $\Delta$DAS predictions were calculated by subtracting the final endDAS predictions from the given baselineDAS values.

In the challenge, we were also given the opportunity to submit predictions from a clinical model which did not incorporate any genetic data. Since we were allowed to use a clinical model that differed from our full model, we used a random forest of clinical variables to predict the scaled residuals in the full model because we were no longer required to use an interpretable model. This provided us the opportunity to evaluate if model interpretability was limiting predictive performance.The overall pipeline is shown in figure 2.5

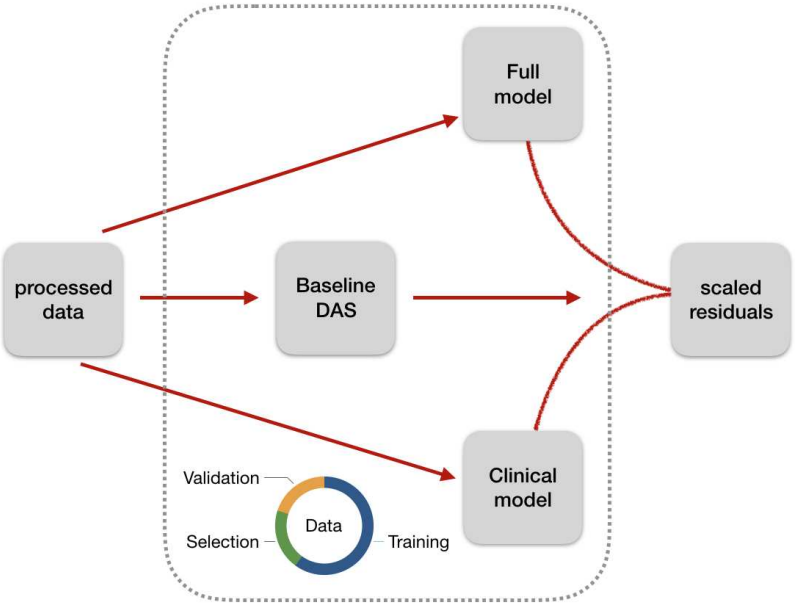



Figure 2.5: Methodology pipeline

## 2.4 Results

### 2.4.1 Comparative results

Our (team Outliers) full model had the best validation set performance, for both tasks, out of all of the teams participating in the final phase of the challenge (see also figures 2.6 and 2.7). Using bootstrap analysis of submission ranks, the organisers of the challenge concluded that our two submissions performed robustly better than all remaining solutions (Wilcoxon signed-rank test of bootstraps on the respectively first and second sub-challenge gave p-value of $5 \cdot 10^{-34}$ and $10^{-66}$).

The other teams used a variety of methods, including Gaussian process regression, gradient boosted classifiers, kernel methods, tree-based regression, linear mixed effect models, and ensembles of machine learning methods (Sieberts et al., 2016). Our final model results on the CORRONA validation set is shown in table 2.2.

Table 2.2: Final validation set results for our full and clinical models, evaluated as correlation for the regression sub-challenge and AUPR or AUC for the classification sub-challenge.

| Model | ΔDAS correlation | non-response AUPR | non-response AUC |
|---|---|---|---|
| clinical + genetic | 0.40201 | **0.516** | **0.6214** |
| clinical | **0.4044** | 0.5063 | 0.6018 |

Surprisingly, within the full model, cross validation of the $\ell_1$-norm regularised sub-model recommended that no SNPs be used to predict the scaled residuals. Since the full model's predictions did not utilise genetic information, the full model could be compared to the clinical model to evaluate the cost of preferring interpretable models in favour of complex models. The clinical model had comparable ΔDAS results to the full model, but had worse non-response results.(Table 2) The ΔDAS results demonstrate that an increase in model complexity did not correspond to an increase in performance. The fact that the full model was not capturing any genetic effect was at first disturbing, since finding genetic biomarkers would have been very interesting. The collaborative phase however and the comparison with other teams results have shown first that our simple full model without genetic contributions performed best and also had the best robustness to overfitting.

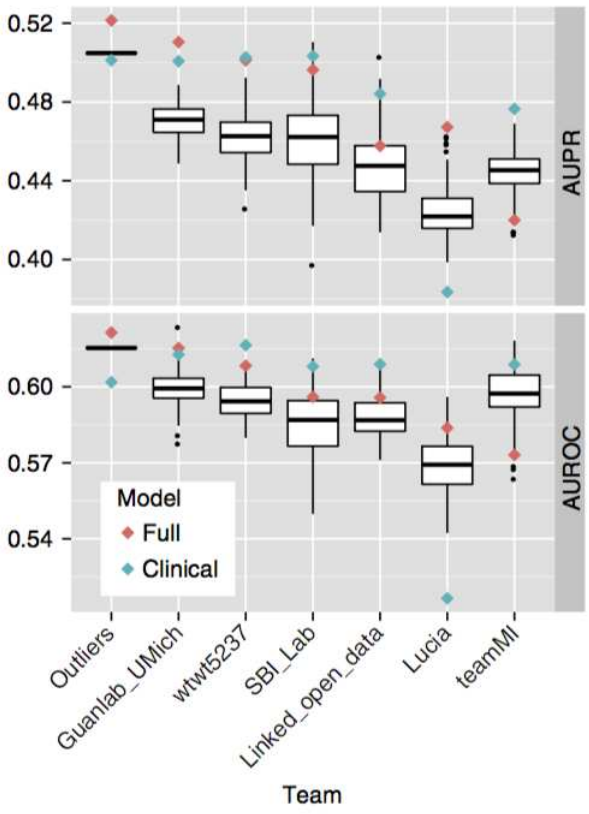



Figure 2.6: AUPR and AUC of each of the top 7 teams full model, containing SNP and clinical predictors, versus their clinical model, which does not consider SNP predictors. The box plot distributions represent the scores resulting from replacing the SNPs in each team's full model with randomly selected SNPs. Figure from (Sieberts et al., 2016)

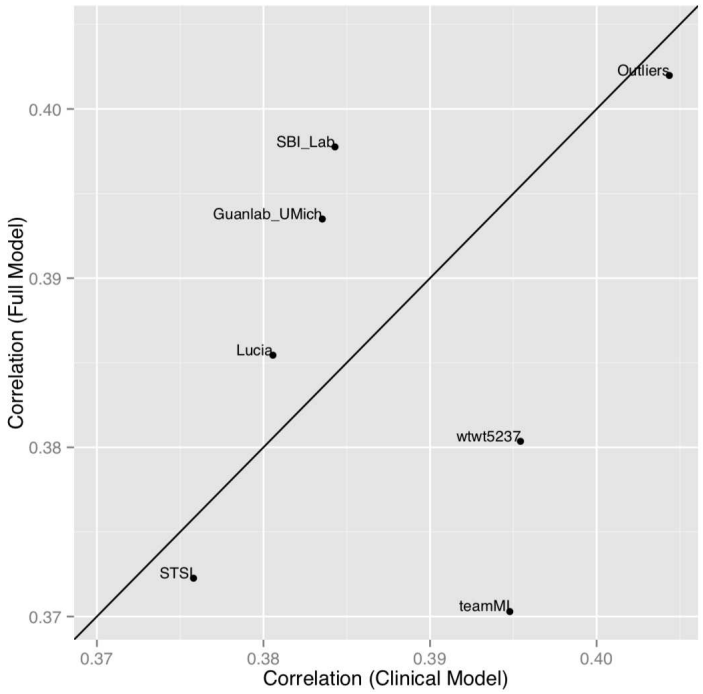



Figure 2.7: Full model versus clinical model performance: score (correlation with true values) of each of the top 7 teams full model (incorporating SNP and clinical data) versus their clinical model excluding SNP information, for the quantitative prediction sub-challenge. Figure from (Sieberts et al., 2016)

### 2.4.2 Results analysis: overfitting and genetic contribution

A comparison between the first leaderboard validation phase and the final CORRONA validation phase results unveils some interesting insights about the models performance. Figure 2.8 shows the behaviour of this performance in terms of AUC for the non-response classification task (left figure) and in terms of correlation for the ΔDAS regression task, for the best performing teams and at the two phases evaluation. A first striking fact is that all teams, including our team, observe an overfitting phenomenon in both tasks. While the two metrics for the classification sub-challenge showed positive correlation (r= 0.71 and 0.60 for AUPR and AUC, respectively) between the scores on the held-out leaderboard data and the scores on the test data, the quantitative prediction sub-challenge showed a negative correlation (r= -0.052) suggesting that overfitting was more severe in the case of the regression task, despite the demographic and other characteristic similarities between the primary cohort and the CERTAIN cohort used at the first and the second phase evaluations (see supplementary table 1 in (Sieberts et al., 2016)). Our team results however were the most robust to overfitting. The simplicity of the approach, but also the absence of spurious features selection such as genetic covariates in the full model, although indirectly through cross-validation, is the ground for this robustness.

Other teams were able to have a better full model than their clinical model, as shown in figure 2.7 (models above the diagonal). It is interesting to notice that the full models from the same teams, such as "Guanlab" who was ranked first at the first leaderboard phase, were the ones that suffered most of overfitting, according to figure 2.8. This ultimately raises the question of genetic contribution to the prediction problem of the given target: ΔDAS.

To explicitly test the ability of teams modelling techniques to detect weak genetic contribution, the challenge organisers first examined the contribution of feature selection to model performance. Most teams had used a combination of knowledge-based and data-driven evidence to perform dimensionality reduction in their model development (which was also our case, but the model selection step at cross-validation ruled out the presence of SNPs in our final model). To approximate the null distribution of the genetic models, each of the 7 top performing teams trained 100 models using an equivalent number of randomly sampled SNPs relative to their best-performing model. The results are shown in figure 2.6. For 5 of 7 classification algorithms, models using knowledge-mined SNP selection significantly outperformed models using random SNPs for AUPR, AUC or both at a nominal p-value $< 0.05$. However, It is surprising to observe that in most cases (except for our team "Outliers" and team "Lucia"), the only clinical variables model proved to be as good, and sometimes better than the full model. Pairwise comparison between clinical and full models across teams demonstrated no statistical difference (paired t-test p-value=0.85 and 0.82, for classification AUPR and AUC, respectively, and p-value0.65 concerning the continuous prediction task correlation) indicating that the contribution of SNP data to the prediction of treatment effect was not of sufficient magnitude to provide a detectable contribution to overall predictive performance.

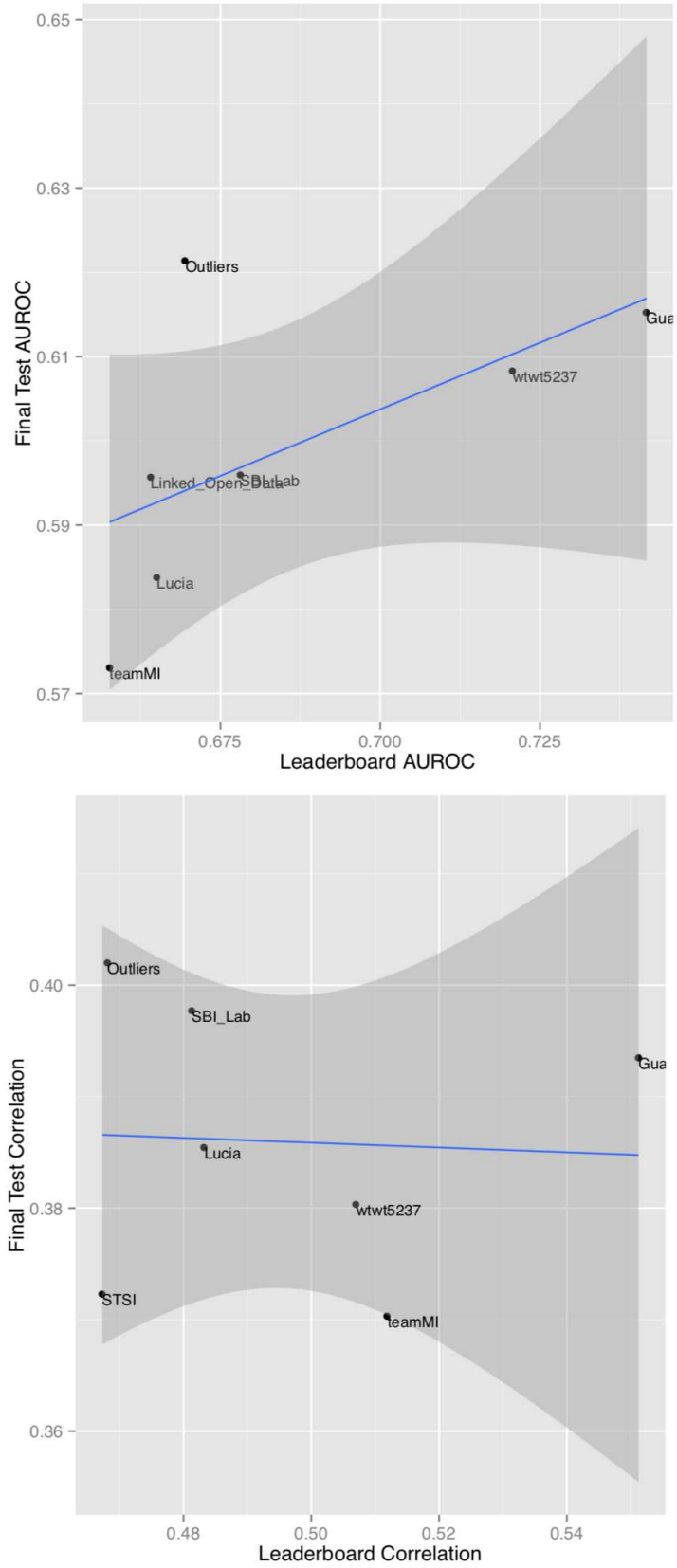



Figure 2.8: First phase leaderboard score versus final submission score as AUC for the classification sub-challenge, and correlation for the quantitative sub-challenge with linear regression fit and 95% confidence region (shaded). Figure from (Sieberts et al., 2016)

## 2.5 Conclusions and acknowledgements

As a first application of machine learning in bioinformatics, I had the chance to participated with our highly motivated team "Outliers" in the RA responder DREAM challenge. This open challenge framework enabled the comparative evaluation of predictions developed by 73 research groups using the most comprehensive available genotype and clinical data of patients and their response to Rheumatoid Arthritis anti-TNF drug therapies.

This project was a very nice opportunity to apply a variety of machine learning techniques on a large exclusive dataset within the supervised learning framework (both classification and regression tasks). It was also an appropriate introduction to research areas in the field and particularly to my further thesis work, since it emphasised the problem of overfitting and the difficulty of going around it in case of low signal to noise ratio or a weak contribution of a part of the dataset in the prediction of the target variable. One of the main findings of the challenge was indeed that SNPs did not meaningfully contribute to the prediction of treatment response above the available clinical predictors. However, the different teams were able to leverage the small set of available clinical features to develop predictions that performed significantly better than random. These results suggest that future research efforts might focus on the incorporation of a richer set of clinical information including seropositivity, treatment compliance and disease duration.

Our team enjoyed the best results in both sub-challenges in the final test phase, with an approximate correlation of 0.4 for the continuous response to treatment regression task, and around 0.62 as AUC for the classification sub-challenge. We value our approach and its relative robustness to overfitting for several reasons:

- The prior selection of SNPs based on biological and medical literature which allowed to considerably reduce the dimension of the data.
- The simplicity of the models used that also permitted embedded feature selection through the $\ell_1$-norm and their interpretability. Our models did already show through feature selection that genetic contributions to the predictions was weak, which was confirmed by the challenge and the community assessments.
- The use of scaled residuals which allowed to implicitly account for some constraints of the response variable.
- A careful 5-fold cross-validation before relying on the leaderboard feedback.

We think that all these steps are crucial in a such a challenge with millions of features from different sources and only a few thousands patients. Finally this challenge helped bringing up different research questions. For instance, how to robustly evaluate individual or collective features contribution to a target prediction in a supervised learning task. How to evaluate and potentially improve the stability of feature selection when the number of observations is small with respect to the data dimension. And finally, how to deal with such overfitting difficulties when little or no prior knowledge is at hand? Further investigations carried in the next chapters will, we believe, partially address this last issue as it was briefly presented in the manuscript Introduction.

I would like to thank all of the "Outliers" team members: Daniel Hidru, Mohsen Hajiloo, Bo Wang, Aziz Mezlini, Cheng Zhao, Rae Yeung for this collaborative work and especially our team leader Anna Goldenberg for having worked hard to advice the team but also actively participating in the model building, for her welcome in Toronto and her guidance throughout the project. I also would like to thank Solveig K. Sieberts and Lara M. Mangravite for their motivation and perseverance in the challenge organisation and the publishing of these results.The data was collected and cured by the Rheumatoid Arthritis Challenge Consortium and the CORRONA network.

# Chapter 3

# Noise injection in the input data

*"If you so choose, each day can be filled with even more joy than the one before. If you so choose, even the most seemingly random events can work in your favour."*

Ralph Marston

Contents

**Abstract**

In this chapter we revisit the history of noise injection in machine learning, providing a taxonomy of the different forms that it has taken in the literature. We then formally redefine in its generality input noise injection INI in the ERM setting. We review the main methods solving the new INI optimisation problem. We provide intuitions, theoretical and empirical studies about the regularisation benefit of INI for general noising functions focusing on dropout for linear model not only as the most popular and studied example of noise injection in the recent machine learning literature but also as an inspiration for our future work in the next chapters. We then provide a new approximation of the INI empirical risk in the case of dropout and show how it provides new theoretical insights of dropout for linear and non-linear models. We finally illustrate the effects of INI in linear models by experiments from simulations and real data examples in order to provide other empirical insights about INI in general and dropout in particular.

**Résumé**

Dans ce chapitre nous revenons sur l'histoire de l'injection de bruit dans le cadre de l'apprentissage automatique en proposant une taxonomie des différentes formes que cette technique a prises dans la littérature. Nous redéfinissons ensuite formellement dans sa généralité l'injection de bruit dans les données INI dans le cadre de la minimisation du risque empirique. Nous passons en revue les principales méthodes permettant de résoudre le nouveau problème d'optimisation résultant. Nous fournissons ensuite des intuitions, une étude théorique et empirique sur les avantages de cette méthode en tant que technique de régularisation en mettant l'accent sur les modèles linéaires et le *dropout*, non seulement en tant que l'exemple le plus populaire et étudié d'injection de bruit dans la littérature récente mais également comme base et inspiration pour nos travaux dans les prochains chapitres. Finalement, une nouvelle approximation du risque empirique est proposée dans le cas du *dropout* fournissant une nouvelle interprétation qui s'étend au cas des modèles non-linéaires et qui présente une implémentation alternative de la même méthode avec certains avantages.

## 3.1 Introduction

### 3.1.1 History and use

Noise is an all-encompassing term that usually describes undesirable disturbances or fluctuations. In biology and other natural science fields, noise typically refers to variability in measured data when identical experiments are repeated or when bio-signals cannot be measured without background fluctuations distorting the desired measurement (Tsimring, 2014; Raser and O'shea, 2005). Noise has therefore been historically a fundamental enemy for communications engineers whose main goal is known to ensure that messages can be transmitted error-free and efficiently from one place to another at the fastest possible rate (Kieffer, 1994). Random noise in the form of electronic fluctuations or electromagnetic interference corrupts transmitted messages. This places limits on the rate at which error-free communication can be achieved. In statistics, statistical noise means unexplained variability within a data sample (Davenport and Root, 1958). The term noise in this context came from signal processing where it was used to refer to unwanted electrical or electromagnetic energy that degrades the quality of signals and data. The presence of noise means that the results of sampling might not be duplicated if the process were repeated. Therefore, the problem of separating out the noise from the signal has long been a focus in statistics as well and is still now in many cases.

However, in 1981, an interesting new phenomenon involving the Brownian motion process was analysed by Benzi *et. al* in the particular case of stochastic dynamical systems (Benzi et al., 1981) and termed as stochastic resonance. It stated that the output signal-to-noise ratio (SNR) can be greater when an appropriate amount of noise is added to weak periodic input signals. This phenomenon can first be thought of as an exotic mathematical curiosity. Stochastic resonance, however, has been observed since then in a large variety of systems including: bistable ring lasers (Vemuri and Roy, 1989), semiconductor devices (Iannelli et al., 1994), chemical reactions(Leonard and Reichl, 1994), human tactile sensing (Beceren et al., 2012), neural networks of mammalian brains (Maass, 2014; McDonnell and Ward, 2011), the blood pressure control system in the human brain (Hidaka et al., 2000) and more. It has also been applied in many engineered systems such as bistable circuits (Harmer et al., 2002). In his book: *Noise* (Kosko, 2006), Bart Kosko, a well-known electrical engineering professor and an early populariser of fuzzy logic and neural networks, terms stochastic resonance as *noise benefit* and introduces the concept of adaptive stochastic resonance in neural networks learning algorithms to find the optimal level of noise to add to many nonlinear systems to improve their performance. He also proves many versions of the so-called *forbidden interval theorem* which guarantees that noise will benefit a system if the average level of noise does not fall in an interval of values (Kosko et al., 2009). He also shows that noise can speed up the convergence of Markov chains to equilibrium and even ventures to suggest that Stochastic Resonance might be a triggering ingredient in the origins of life from inert matter to biological entities.

Stochastic resonance can also occur in static threshold nonlinearities and is used in particular in a signal-processing technique known as dithering (Wannamaker et al., 2000). Dithering involves deliberately adding a random or pseudorandom signal to another signal prior to its digitisation (the process of converting information from a physical format into a digital one) or quantisation (the process of mapping input values from a large set (often a continuous set) to output values in a (countable) smaller set such as rounding and truncation). This randomisation, although increasing the total power of the noise at the output, reduces undesirable harmonic distortion effects introduced by quantisation (Schuchman, 1964). This counterintuitive approach to enhance signal-to-noise ratio, although seemingly distant from supervised machine learning, has been used recently in what is called neural network quantisation (Jacob et al., 2017) where uniform noise is added to the activations of convolutional networks with quantised weights in order to improve the test accuracy (Baskin et al., 2018).

Noise injection is also historically and conceptually related to randomised algorithms.

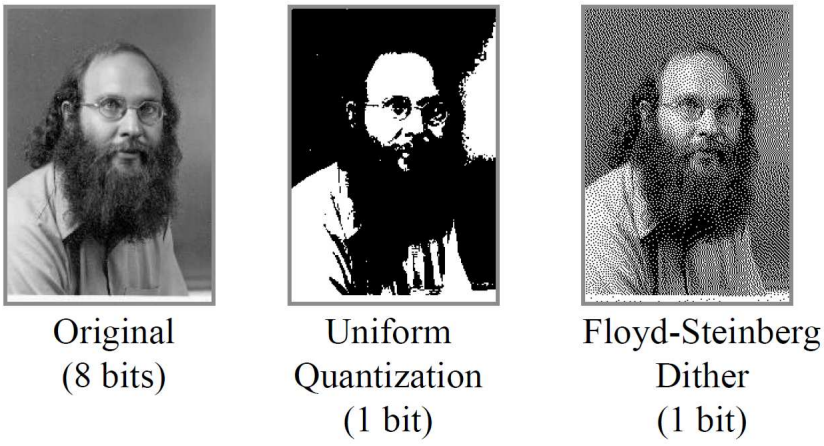



Figure 3.1: Example of dithering.

Randomised algorithms have already been extensively used in programming and have naturally also gained interest in machine learning. Either to design new learning algorithms (Quinlan, 1986; Huang et al., 2011a), to derive faster algorithms (Osoba et al., 2011; Jordan and Mitchell, 2015; Woodruff, 2014), to ensure privacy (Geng and Viswanath, 2016) or to improve performance results of the deterministic counterpart (Mitzenmacher and Upfal, 2005; Gallicchio et al., 2017; Oneto et al., 2017). Even if these numerous benefits have been proven empirically and even theoretically in many cases, we still do not know generally whether the randomisation part truly *helps*: in the sense of whether $P \neq BPP$ or not (that is if there exists a deterministic counterpart to each randomised algorithm that does the same thing).

In all cases, randomised algorithms are nowadays embedded in many computational and probabilistic methods including machine learning and its success raised the interest of research and was the object of many conferences and machine learning workshops (see for instance the NIPS 2013 workshop: *Randomised Methods for Machine Learning)*. Noise injection has today pervaded all the fields of machine learning: in reinforcement learning, research about the benefits of noise in boosting the performance by extending the exploration through noise injection in the action space or to the agent's parameters space is still very active (Hutter and Poland, 2004), encouraged by positive results in other related fields such as cognitive science and deep learning. In unsupervised learning, the idea of training with noise is rooted in many methods: denoising autoencoders (DAE) (Vincent et al., 2010) are trained to reconstruct their clean inputs with noise injected at the input level, while variational auto-encoders (VAE) (Doersch, 2016) are trained with noise injected in their stochastic hidden layer. This line of work has been theoretically justified (Im et al., 2015; Vera et al., 2018) and empirically successful in learning good representations.

As mentioned in the introduction, we will focus in this manuscript on noise injection in supervised learning and its use as a regulariser, that is to prevent overfitting. We focus therefore in the next section on the possible supervised learning frameworks where noise injection can have a meaning and possibly help improve generalisation. Although we will focus in this manuscript on the ERM framework where the aim as stated in the introduction is to find a model that has a small average error over the data distribution, it is also possible to use other different frameworks depending on the setting. These frameworks essentially differ in the way they aggregate the noise being added. They convey as well a different facet of noise injection as a regularisation that is not only important for a global overview of noise injection regularisation in supervised learning but that will also be useful in guiding the intuition, interpreting and expanding noise injection in the ERM framework itself.

### 3.1.2 A taxonomy of noise injection in supervised learning

Different optimisation frameworks have considered the presence of uncertainties, by introducing this same uncertainty in the problem to be optimised, in a way that is in line with their own mathematical frameworks. We can indeed distinguish, depending on the optimisation framework used, clear differences in:

- The mathematical representation of the noise injected
- The aggregation procedure of the injected noise in the optimisation problem
- The goal of the noise injection procedure

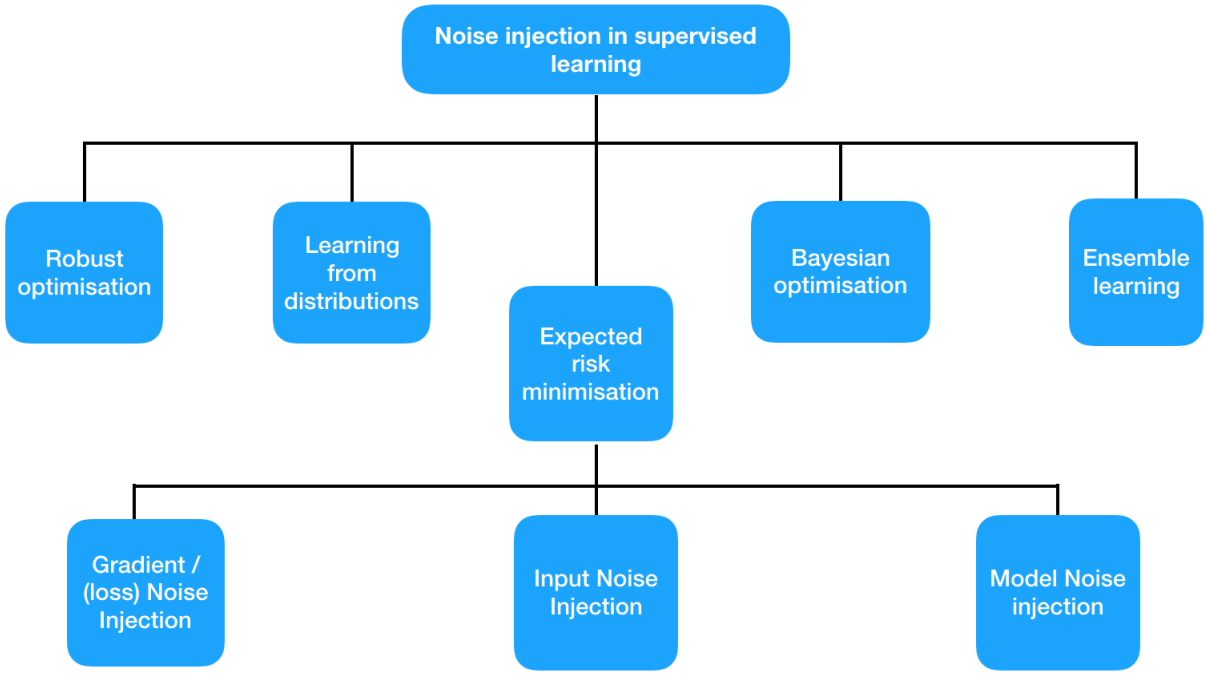



Figure 3.2: Noise injection in different supervised learning frameworks

Following these criteria, one can distinguish different concerned frameworks that have been used in the literature of supervised learning and that have considered additional uncertainties in the data by incorporating them in the optimisation process: Bayesian optimisation, learning from distributions, robust optimisation, ensemble learning and the ERM framework that was presented in the introduction and that will be the considered framework of this manuscript. One main reason to focus on the ERM framework is that most of machine learning in the supervised setting and consequently most of the noise injection literature follow this framework. Nevertheless we provide here, for completeness, a brief overview of how the other frameworks do provide alternative ways to inject noise in the learning process and how they deal and aggregate this added noise.

Bayesian strategies naturally treat optimisation objects and parameters as random variables to be inferred via a posterior distribution, for which we assign prior distributions

(Chakraborty, 2005). These priors captures our beliefs about the model and the data. Therefore, in Bayesian Optimisation (BO) it is straightforward to consider the added uncertainty either as a new prior distribution over the desired object to be noised or as a new random variable with its own prior distribution that can be plugged or learnt at the inference time as a latent variable (Graves, 2011). The main difference with the classical framework considered in most of the machine learning noise injection literature, that is the ERM framework presented in the introduction, is that the marginalisation over the noise latent variable will be in the Bayesian setting over the likelihood and not the risk (that is usually the negative log-likelihood). This marginalisation is usually computationally intractable (or with a very high cost of computation) (Shahriari et al., 2016) and requires approximations such as variational inference (Blei et al., 2017) or special algorithms such as Gibbs sampling or Expectation Maximisation (Casella and George, 1992; Moon, 1996) that we will not describe here. Interestingly, a strong connection exists between these approximations and noise injection in the classical framework of ERM, as it is the case between Bayesian optimisation and regularisation. Bayesian interpretations of noise injection in the ERM framework has also allowed to build adaptive forms of noise in a principled way (Maeda, 2014; Gal and Ghahramani, 2016a), and apply the same noise injection techniques in other models such as recurrent neural networks (Gal and Ghahramani, 2016b). This connection will be more detailed later in this chapter.

Another framework that was only recently developed deserves mentioning, as it sets a beautiful theory for introducing uncertainty in kernel methods in particular (a popular class of methods for supervised and unsupervised learning) and that allows for introducing input uncertainty in a principled way yet different from the Bayesian way, is called *learning from distributions* (Muandet et al., 2012). This framework is relatively new and represents a convergence of isolated efforts to adapt kernel methods to learning from distributional objects. Learning from distributions is not be mixed with distribution learning which is concerned with learning the probability distribution of a random variable given a set of its realisations. This framework proposes to learn a mapping from distributions to labels given $n$ pairs of labeled distributions. This framework is mainly different from the ERM framework in that fact that the risk functional to be minimised is expressed in terms of the loss of the expected model parameters over the data distribution in contrast with the expected loss of the model (Szabó et al., 2016). To see the link between learning from distributions and noise injection in the ERM setting, noise injection can exactly be formalised by this setting as an uncertainty around the observations' values which transforms each of these values to a known distribution with that realisation as a mean value. Therefore the noise injected can be considered as a set of random variable of the size of the data samples, where each random variable affects a given realisation to a distribution. An equivalence is actually shown between the new optimisation problem over the resulting noisy distribution and learning with mean embeddings (expectation of the original kernel taken over the introduced distributions) in the traditional ERM framework, which first provides a keystone in the consistency of the new optimisation problem and potential ways to solve it but also opens up for the theory of mean embeddings over probability distributions (Muandet et al., 2017). Empirical studies show that this noise injection can reduce overfitting in many cases, although a complete theoretical link between regularisation and learning with mean embeddings for other kernels is still missing (Zhu et al., 2017).

Robust Optimisation (RO) is another framework developed among others by Soyster (1973) as a paradigm solving convex problems with *hard* constraints created by bounded uncertainty in the parameters. Instead of being interested in expectation, mean or mode such as in stochastic optimisation, in RO one is interested in the worst case analysis and thus the extrema of the risk under given constraints. For a detailed overview of the RO framework,

we refer the reader to (Gorissen et al., 2015; Norton et al., 2017a; Ben-Tal and Nemirovski, 2000; Ben-Tal et al., 2009). Robust optimisation has naturally found applications in machine learning and will probably witness more interest with fields such as adversarial machine learning, which aims to immunise machine decisions against potential malicious attacks (Shaham et al., 2015). A considerable amount of theoretical work has been developed to support the framework such as properties of robust solutions and many connections have already been studied between existing machine learning algorithms and RO. Support Vector Machines (SVMs) for example, a supervised learning algorithm that builds a hyperplanes maximising distances between data points of different classes (Hearst et al., 1998), naturally corresponds to a robust optimisation problem (Xu et al., 2009b), and popular regularisations such as Tikhonov regularisation and the LASSO (presented in the introduction) are shown also to correspond to robust formulations (Xu et al., 2009a). These formulations allowed on one hand to understand better the generalisations, consistency and structural properties of the corresponding models and on the other hand to generalise these families of regularisers to more broad methods (Xu and Mannor, 2012). Although the view of robust optimisation can be found overly pessimistic from a statistical view and lacking assumptions about the distribution or the nature of the noise introduced (which is not the case for a user-added noise that we explore in this manuscript), more recent variations explore more optimistic versions of RO (Norton et al., 2017b) or by bridging it with stochastic optimisation such as distributionally-robust stochastic optimisation (Goh and Sim, 2010) and is likely to find more interest and development in the future.

Ensemble methods are learning algorithms that use a set of models that are learnt using the data and one or different optimisation problems in order to construct a final model which will be used to classify the new data (see introduction for a brief description). In ensemble learning, the second step which is an aggregation procedure of the sub-models in a final model takes different forms in the literature such as averaging, weighted averaging and majority voting (Dietterich, 2002; Zhang and Ma, 2012). The first step of training individual learners has as a role to construct different models that are at the same time accurate (that is better than random) and diverse (that is having different errors on new data points) also takes different forms and can be briefly classified as:

- Using different learning algorithms
- Injecting randomness in the learning algorithm,e.g. random initialisation
- Manipulating the training examples, e.g. the bootstrap.

We can therefore see that injecting noise in the data is embedded in many ensemble methods as a part of the training process. Adding even more noise when manipulating the examples is shown also to be more effective in terms of performance of the final model (Raviv and Intrator, 1996). It is indeed obvious to see that adding noise to the data is the main cause of diversity when using the learning algorithm especially if there is no randomness involved during training and in the absence of other randomness such as random initialisations. As shown in the introduction, such diversity can reduce the variance of the ensemble method and improve its generalisation performance (Kuncheva and Whitaker, 2003). However, mixture of experts methods requires reasonable time for training data for classifiers that is proportional to the number of individual classifiers and their individual computational complexity, hence resulting in high computational cost (Dietterich, 2000).

The term *noise Injection* has been first used in an explicit way in supervised learning in the case of neural networks in the late 80's under the stochastic optimisation scheme which falls under the ERM framework (Plaut et al., 1986; Alspector et al., 1988). Other terms

such as *training with noise*, *adding noise* and *jitter* were also used (Matsuoka, 1992). Indeed, inspired by the robustness of biological neural networks to noisy environment and/or the loss of neurons, a large body of work has tried to study and emulate this property for artificial neural networks that was termed as *fault tolerance*, although also the terms *robustness* and *reliability* were equally used. In (Chiu et al., 1994), the authors list different ways that have been used to introduce uncertainties during training, such as flipping the labels, adding Gaussian noise to the the input data or in the synaptic weights.

It is hard to assess which machine learning method came first with the idea of training with noise but the idea already appeared in the late 80's, interestingly in one of the works of G.E.Hinton among others Plaut et al. (1986), who was also the one suggesting the dropout method that recently revived interest in noise injection techniques in deep learning and that we will describe later. Other papers appeared indeed in the same period and seem to suggest independently training neural networks while adding noise in the data such as in (Gardner et al., 1989; Peeling et al., 1986). All these experimental studies used the same trick of stochastically adding samples of noise from the same distribution to either weights or the input of the neural network layers. In terms of the used noise distribution, most of the methods focus on Gaussian noise addition in the input of the neural network, known as *jitter* which have then received a lot of attention in those years (Holmstrom and Koistinen, 1992). Noise injection in neural networks was then clearly inspired from biological systems, and statistical studies about benefits of noise injection in terms of generalisation have already started to be empirically analysed such as in (Sietsma and Dow, 1988, 1991; Edwards and Murray, 1996; Murray and Edwards, 1994; An, 1996). Convergence and consistency properties of noise injection in the ERM setting were studied such as links between noise injection and learning invariances (Leen, 1995) or regularisation properties such as connection with weight decay that were revealed for instance in (Bishop, 1995a; Matsuoka, 1992). Three decades before the re-emergence of deep learning, regularisation by noise was already known as a common technique to improve generalisation performance of neural networks but that witnessed a decline in use with the decrease of popularity of neural networks methods, even though it has been adapted by kernel methods (Decoste and Schölkopf, 2002) and ensemble learning cited previously. With the emergence of complex deep networks, regularisation with noise injection regained attention and led to the development of successful techniques, most notably *dropout*, which received lately a lot of attention in the machine learning and deep learning literature and that we will detail in the next sections and further develop in the other chapters.

Let us recall that in the ERM framework, we usually have three objects that one uses for the optimisation problem: a loss function, the model (usually represented by a set of weights if it is parametric) and the data (observations and labels if it is supervised, observations without labels if it is unsupervised, both if it is semi-supervised). Sometimes the model or the optimisation problem have hyper-parameters that are not included in the optimisation problem. Such hyper-parameters include for instance regularisation trade-offs or the number of the layers and the number of units per layer for neural network architectures. These can be fixed by cross-validation or a set of heuristics.

Following the three objects mentioned, we can distinguish three instances or families of noise injection:

- Adding noise to the loss or its gradient such as in (Raginsky et al., 2017; Welling and Teh, 2011)
- Adding noise to the model parameters such as (Murray, 1992; Murray and Edwards, 1994; Wan et al., 2013) for noise injection in the weights of multilayer neural networks and (Srivastava et al., 2014) for noise injection in the activations
- Adding noise to the input data such as (Rifai et al., 2011)

This separation is first conceptual and does not mean that these categories are independent. Even if studying these different schemes and the relationship between them is beyond

the scope of this manuscript, we will still hint to obvious links between these schemes. We will notice for instance in the case of linear models, which we will focus on in a large part of this manuscript, that these schemes are closely related and sometimes result in the same procedure (this will be developed in the next section about dropout). In this manuscript we will be focusing on noise injection in the input data which we will define in the next section.

**Literature comparison of the different frameworks:** It is of course hard to assess a whole framework as better methods can appear, leveraging newer and more efficient algorithms and sets of tricks. However, one can make a relative, modest and up-to-present literature comparison of noise injection development between the different frameworks according to several criteria.

- Flexibility: in terms of diversity of forms that the noise can take as a mathematical object and if the framework can be applied on the variety of real world examples.
- Understandability: in terms of consistency of the theory and how much we understand of the effects of noise injection in the corresponding framework
- Efficiency: in terms of computational cost
- Efficacy: in terms of generalisation results, as relative improvement with noise injection as opposed to without.

Qualitative assessment of noise injection techniques under the different presented frameworks and following these criteria is summarised in 3.1.

Table 3.1: Literature comparison of noise Injection under different supervised learning optimisation frameworks

| ***Framework / Criterion*** | *Flexibility* | *Understandability* | *Efficiency* | *Efficacy* |
|---|---|---|---|---|
| *Learning from distributions* | *Medium* | *Medium* | *Medium* | *Low* |
| *Robust optimisation* | *Low* | *High* | *Medium* | *Low* |
| *Ensemble learning* | *High* | *Low* | *Low* | *High* |
| *Bayesian optimisation* | *High* | *Medium* | *Low* | *Medium* |
| *ERM* | *High* | *Medium* | *High* | *Medium* |

## 3.2 Formulation

In the supervised learning setting, we are given a series of $n$ pairs of the form $(x_i, y_i)_{i=1,\dots,n}$ that are assumed to be $n$ realisations of independent and identically distributed pairs of random variables $(X_1, Y_1), .., (X_n, Y_n)$ from a joint distribution $\mathcal{P}$ on $\mathcal{X} \times \mathcal{Y}$, and one primary goal is to find or construct a mapping $f : \mathcal{X} \mapsto \mathcal{Y}$ that minimises the empirical risk

$$R_{emp}(f) = \sum_{i=1}^{n} \left[ L\left(f(x_i), y_i\right) \right] .$$

We formulate noise injection in the input as a transformation of the input parametrised by a random variable with a known distribution.

More precisely, noise injection consists in learning the same problem but with data drawn from new noisy random variables $(\widetilde{X_i}, Y_i)$ $i = 1, \dots, n$, following a new distribution $\widetilde{\mathcal{P}}$. The new random variables $\widetilde{X_i}$ are defined as a transformation of the variables $X_i$, $i = 1, \dots, n$ using the *noise* variable $\delta$ from a **known** distribution $\Delta$.

$$\widetilde{X_i} = \nu(X_i, \delta), i = 1, \dots, n \, .$$

where $\nu$ is a deterministic noising function. We require the noise injection to be unbiasing, i.e. :

$$\mathop{\mathbb{E}}_{\delta\sim\Delta}(\nu(X,\delta)) = X\,.$$

In this manuscript we will focus on two important noising functions:

- **Additive noise** We take the noising function as a simple addition

$$\nu(X,\delta) = X + \delta\,.$$

  In this case, the unbiasing condition becomes

$$\mathop{\mathbb{E}}_{\Delta}(\delta) = 0_{\mathcal{X}}\,.$$

- **Multiplicative noise** We will also consider the noising function as component-wise multiplication:

$$\nu(X,\delta) = \delta \odot X\,.$$

  Where $\odot$ is the element-wise multiplication of the two vectors. In this case, the unbiasing condition becomes

$$\mathop{\mathbb{E}}_{\Delta}(\delta) = \mathbf{1}_{\mathcal{X}}\,.$$

**Remark 2.** *Here we have considered to focus on adding noise only to the variables $X_i$. One can imagine applying noise on both the data and the labels, either by considering another label noise variable and another noising function for the label, or by considering the same noise and function over the whole space. This generalisation of the proposed noising scheme can be particularly impactful in domains where labeled data is scarce or inherently noisy (Thiel, 2008). We do not develop here this direction as it requires alone another line of research (which noising function to choose and what effect on the learning problem). One intuition following data augmentation that we we presented in introduction and will link later with INI indicates possible positive results on generalisation performance, as also indicated by (Thiel, 2008). Research in noising both labels and observations simultaneously is however almost inexistant (if we exclude noise label modelling) although it has been shown lately to be useful for studying learning methods properties such as multilayer neural networks (Zhang et al., 2016).*

**Remark 3.** *It is possible to parametrise the distribution $\Delta$ as a function of the input data distribution $\mathcal{P}$ (or if $\mathcal{P}$ is unknown on the $n$ observations). Related work will be presented in* Adaptive Structured Noise Injection.

**Remark 4.** *Since the definition here does not include injecting noise in the labels, the same formulation can in fact be used in unsupervised learning. The authors of the original study on dropout, a recent and popular noise injection in the input layers of neural networks, apply for example this noise scheme in a type of unsupervised models called Botlzmann Machines and even show improvement in some aspects (Srivastava et al., 2014).*

The risk to be minimised is now :

$$\widetilde{R}(f) = \mathop{\mathbb{E}}_{(X,Y)\sim\widetilde{\mathcal{P}}}[L\left(f(X),Y\right)] = \mathop{\mathbb{E}}_{(X,Y)\sim\mathcal{P}}\mathop{\mathbb{E}}_{\delta\sim\Delta}\left[L\left(f(\nu(X,\delta)),Y\right)\right]\,. \tag{3.1}$$

Using the data observations as an approximation, the INI empirical risk that will be minimised is :

$$\widetilde{R}_{emp}(f) = \frac{1}{n}\sum_{i=1}^{n}\mathop{\mathbb{E}}_{\delta\sim\Delta}\left[L\left(f(\nu(x_i,\delta)),y_i\right)\right]\,. \tag{3.2}$$

## 3.3 INI as a regulariser

As detailed in the introduction, the ultimate aim of a supervised learning model is to have a null or at least a very small error on unseen data given training data, that is, to minimise the expected risk given its empirical estimate. Since its introduction in the machine learning literature, input noise injection methods have been motivated by this aim and more precisely by helping to *prevent overfitting*. As hinted in our brief history of noise injection, this motivation might seem counterintuitive at first. Several intuitions however can justify the use of such a technique. We briefly review and discuss these intuitions, then present some of the regularisation properties of INI in inear models.

### 3.3.1 Intuitions

Several intuitions have justified the empirical effectiveness of Noise Injection.

- **Data Augmentation intuition:** Maybe one of the most intuitive ways to see why INI can be related to regularisation techniques, is to interpret noise injection as implicit data augmentation. Indeed, if we consider a finite sample estimation of the new INI empirical risk defined in 3.2, by considering a set of noise samples $\delta_k, k = 1, \ldots, m$ drawn randomly from $\Delta$ and defining an empirical discrete distribution $\Delta_m$, the empirical risk to minimise becomes:

$$\widehat{\widetilde{R}_{emp,\Delta_m}}(f) = \frac{1}{m.n} \sum_{k=1}^{m} \sum_{i=1}^{n} \left[ L\left( f(\nu(x_i, \delta_k)), y_i \right) \right] . \tag{3.3}$$

  It is straightforward to notice, first that

$$\mathbb{E}\left[ \widehat{\widetilde{R}_{emp,\Delta_m}}(f) \right] = \widetilde{R}_{emp}(f) .$$

  In other terms, the finite sample estimate of the noise injection empirical risk is unbiased, and second that :

$$\lim_{m \to \infty} \left[ \widehat{\widetilde{R}_{emp,\Delta_m}}(f) \right] = \widetilde{R}_{emp}(f) , \ (\textit{by the law of large numbers}) \quad .$$

  Therefore, INI can be seen as a way to train models on an *infinitely augmented* training set.

  As presented in the introduction, data augmentation is an intuitive regularisation technique that has shown to be effective from its beginnings to the present day (Witten et al., 2016). Indeed, since the generalisation ability of the model depends on the number of samples at hand, the intuition of creating new samples to enlarge the training set seems natural. However data augmentation has almost always been performed with a highly structured corruption process based on much prior knowledge (Van Dyk and Meng, 2001). Besides the different forms of data augmentation that were discussed previously, a close idea that gave significant improvement in the accuracy in benchmark vision recognition tasks is considering elastic distortions as transformations that use random displacement fields, which can be seen as closely linked to the framework of noise injection. In (Simard et al., 2003), authors use random scaling and elasticity parameters introduced in the transformations along with convolutional neural networks and define the elasticity exactly as the variance of the Gaussian noise convolved with input values. In (Konda et al., 2015), authors show that noise injection at different layers of a neural network can also be viewed approximately as augmenting the training data but in a particular way that depends on the network parameters and chosen level of representation.

- **Robustness intuition:** Another intuition that has been mentioned as one of the main motivations for noise injection is that the resulting trained model would be robust to the addition of such noise, that is that its performance will not drastically degrade if the model is trained again on a noisy version of the same dataset. Interestingly, one of the inspirations of the modern reuse of noise injection in deep neural networks that comes from the family of denoising auto-encoders, a type of artificial neural network used to learn efficient data codings in an unsupervised manner, is also one of the main carriers of this intuition. In (Vincent et al., 2008), the authors consider that a good model and representation should be robust under corruptions of the input which follows the set of intuitions of artificial neural networks emulating biological neural networks. These auto-encoders take as input a noisy version of the training data whilst training to recover the original undistorted input and have empirically shown significant improvement on benchmark datasets (Vincent et al., 2010). (van der Maaten et al., 2013) also interestingly show empirically that the use of dropout improves generalisation performance of the resulting linear model in the case of what is called *nightmare at test time* (Globerson and Roweis, 2006) in which some of the features are deleted during testing (due to sensor failures or because the feature computation exceeds a time budget) and outperforms models trained with the robust optimisation view of the *worst-case scenario*. It is remarkable however that little work has focused on theoretically proving or analysing this intuition in spite of its appearant simplicity. One of the main reasons is of course the difficulty in analysing the stochastic noise injection scheme which we will hint to later in this chapter, but also the fact that often the use of terms such as *robustness* and *stability* is too wide and is not backed up by a proper definition. One can mention the work of (Hardt et al., 2016) which show that noise injection can improve stochastic approximation bounds on models stability in the case of Lipschitz loss. The work around stochastic robustness has only started to flourish lately with the discovery of the sensitivity of neural networks to chosen noise (Szegedy et al., 2013) and the emergence of the field of adversarial machine learning.

- **Invariance intuition:** Another intuition that is relevant in the case where the noise affects the observation but not its label (as in our formulation) is that for each example of the training dataset, we are constructing a set of pseudo-examples (generated by the noising function), that do share the same label. We are thus encouraging label-invariance through the noise injection. For many datasets indeed, this intuition makes sense : Images that have a certain label do usually have the same label after some pixels removal, as it is the case for documents where some words are deleted or changed. (Leen, 1995) list different methods for encouraging invariance in machine learning and lists data augmentation with examples transformed under the desired invariance property while maintaining the label as one method to achieve the invariance. Note that the particular noise injection strategy is not forcing the model to be invariant but rather encouraging the invariance through the error function. Lately, some connections between the *dropout* noise injection scheme and learning invariant representations have been theoretically and empirically drawn in (Achille and Soatto, 2018b) for instance. There seems however to be much more room for further work around the subject which is gaining popularity in the special case of deep learning (Achille and Soatto, 2018a).

### 3.3.2 General properties for linear models

We start with simple and general properties about the new INI empirical risk :

**Property 1.** *For linear models and convex loss functions with respect to the weights (respectively strictly convex),* $\widetilde{R}_{emp}$ *is convex (respectively strictly convex).*

**Proof** The original empirical risk is convex as the average of convex function. The convexity of the INI empirical risk follows directly from the linearity of the expectation □ . Therefore, and for convex loss functions, convergence to a global minimum is assured, even though this minimum is not necessarily unique.

**Property 2.** *For linear models and convex loss functions (with respect to the weights), the INI empirical risk can be written as:*

$$\widetilde{R}_{emp} = R_{emp} + \Omega_{\nu,\Delta}(\mathcal{D}, w)\,.$$

*where $\Omega_{\nu,\Delta}(\mathcal{D}, w)$ is a non-negative term that potentially depends on the data and the weights.*

**Proof** Let us denote :

$$\Omega_{\nu,\Delta}(\mathcal{D}, w) = \frac{1}{n}\sum_{i=1}^{n}\left[\mathop{\mathbb{E}}_{\delta\sim\Delta}\left(L\left(w\cdot(\nu(x_i,\delta)), y_i\right)\right) - L\left(w\cdot x_i, y_i\right)\right].$$

By substracting and adding the original empirical risk, we have

$$\begin{aligned}\widetilde{R}_{emp} &= R_{emp} + \frac{1}{n}\sum_{i=1}^{n}\left[\mathop{\mathbb{E}}_{\delta\sim\Delta}\left(L\left(w\cdot(\nu(x_i,\delta)), y_i\right)\right) - L\left(w\cdot x_i, y_i\right)\right] \\ &= R_{emp} + \Omega_{\nu,\Delta}(\mathcal{D}, w)\,.\end{aligned} \tag{3.4}$$

$\Omega_{\nu,\Delta}(\mathcal{D}, w)$ is non-negative by Jensen inequality since the loss function is convex □ . For linear models and convex loss functions, input noise injection acts indeed as a regularisation where the regulariser is data-dependent. The last property shows therefore how minimising the new INI empirical risk introduces a tradeoff between the original empirical risk and what can be seen as a bias introduced by noise injection or a regularisation penalty, that is a non-negative function of the model weights, just like $\ell_1$ or $\ell_2$ penalties that were presented in the introduction. Here however, the penalty depends also on the data and the loss function which makes it potentially interesting but harder to analyse.

### 3.3.3 Exact marginalisation

Another way to minimise the noise injection objective would be to directly compute the expectation of the empirical risk over the noise variable. In some cases, as for instance when the model $f$ is linear and the loss is quadratic, we can perform a full marginalisation on the noise:

$$\begin{aligned}\widetilde{R}_{emp} =& \frac{1}{n}\sum_{i=1}^{n}\mathop{\mathbb{E}}_{\delta\sim\Delta} L(w\cdot\nu(x_i,\delta), y_i) \\ =& \frac{1}{n}\sum_{i=1}^{n}\mathop{\mathbb{E}}_{\delta\sim\Delta}\left(y_i - w\cdot\nu(x_i,\delta)\right)^2 \\ =& \frac{1}{N}\left(\sum_{i=1}^{n}(y_i - w\cdot x_{i,})^2 + \mathbb{E}\left[(w\cdot\nu(x_i,\delta))^2 - (w\cdot x_i)^2\right]\right) \\ =& R_{emp} + \frac{1}{n}\sum_{i=1}^{n}\mathop{\mathrm{Var}}_{\delta\sim\Delta}\left[w\cdot\nu(x_i,\delta)\right] \\ =& R_{emp} + \frac{1}{n}\sum_{i=1}^{n} w^\top \mathop{\mathrm{Var}}_{\delta\sim\Delta}\left[\nu(x_i,\delta)\right] w\,.\end{aligned}$$

This decomposition of the new empirical risk in two terms, where the first is the empirical risk on the original observations and the second is a quadratic term in $w$, has in fact already been written in (Richard and Lippmann, 1991; Haykin, 1994; Bishop, 1995a) for the case

of additive noise and then later by (Wager et al., 2013; van der Maaten et al., 2013) in the case of multiplicative noising function, and we rewrite it here for general noising functions. This particular analysis is indeed an important tool, not only as a way to minimise the new empirical risk without having to explicitly add noisy observations before or during the training, but it will also be a way to understand the effect of noise injection on the optimisation problem and thus of its solution and the statistical properties of this solution.

One can see indeed that for the two examples of additive and multiplicative noise we can easily compute the variance over the noise:

- For additive noise with variance $\lambda$: $\mathrm{Var}\left[\nu(x_i, \delta)\right] = \lambda$ and thus the INI empirical risk can be rewritten as :
$$\widetilde{R}_{emp} = R_{emp} + \lambda \left\|w\right\|_2^2 . \tag{3.5}$$
One can see that in this case INI is exactly equivalent to $\ell_2$-norm regularisation. This type of regularisation and its benefits on generalisation have been extensively studied in inverse problems optimisation, statistics and machine learning literature (see introduction). This marginalisation is therefore the first simple yet rigorous proof of why INI can prevent overfitting.

- For multiplicative noise with induced variance $\mathrm{Var}\left[\nu(x_i, \delta)\right] = \lambda \left\|x_i\right\|_2^2$ and in this case the INI empirical risk can be rewritten as :
$$\widetilde{R}_{emp} = R_{emp} + \lambda \frac{\sum_{i=1}^n \left\|x_i\right\|_2^2}{n} \left\|w\right\|_2^2 . \tag{3.6}$$
One can see that in this case INI is again exactly equivalent to an $\ell_2$-norm regularisation. The regularisation hyper-parameter here is data-dependent but since *cross-validation* is usually used to find an optimal value for $\lambda$, there is in the case of quadratic loss and linear models virtually no difference between additive and multiplicative INI.

### 3.3.4 Approximated marginalisation

It is interesting to look at what happens when the loss is non-quadratic but still smooth (such as the usual logistic or cross-entropy loss functions used for classification). For a general form of smooth loss and in the linear model setting, we can write the Taylor approximation of the loss up to the second order with respect to $w \cdot x_i$ :

$$\begin{aligned} L(w \cdot \nu(x_i, \delta), y_i) \simeq\ & L(w \cdot x_{i,}, y_i) \\ & + \frac{\partial L(w \cdot x_i, y)}{\partial w.x_i} \left[w \cdot (\nu(x_i, \delta) - x_i)\right] \\ & + \frac{1}{2} \frac{\partial^2 L(w.x_i, y)}{\partial^2 w.x_i} \left[w \cdot (\nu(x_i, \delta) - x_i)\right]^2 . \end{aligned}$$

Taking the expectation with respect to the noise $\delta$, the first order term cancels out since

$$\mathop{\mathbb{E}}_{\delta \sim \Delta} (\nu(x_i, \delta)) = x_i .$$

We get:

$$\begin{aligned} \widetilde{R}_{emp} &= \frac{1}{n} \sum_{i=1}^n \mathop{\mathbb{E}}_{\delta \sim \Delta} L(w \cdot \nu(x_i, \delta), y_i) \\ &\simeq R_{emp} + \frac{1}{2} \sum_{i=1}^n \frac{\partial^2 L(w \cdot x_i, y)}{w.x_i} \left(\mathrm{Var}\left[w \cdot \nu(x_i, \delta)\right]\right) . \end{aligned} \tag{3.7}$$

We still find the second order term with respect to the noising function, because we locally perform a quadratic approximation of the loss function and which will lead again to

an $\ell_2$-norm regularisation. However, a set of new terms appear which represent the second derivative of the loss with respect to the model parameters at the training samples, or in other terms the *Hessian* of the loss. The Hessian plays an important role in many aspects of optimisation and learning as in for instance:

- Several nonlinear optimisation algorithms used for training neural networks are based on considerations of the second-order properties of the error surface, which are controlled by the Hessian matrix (Battiti, 1992). These algorithms are known for faster convergence and stability properties (Meyer, 1970; Ruder, 2016)
- The inverse of the Hessian has been used to identify the least significant weights in a network as part of network "pruning" algorithms (Hassibi and Stork, 1993)
- The Hessian plays a central role in the Laplace approximation for a Bayesian neural network. Its inverse is used to determine the predictive distribution for a trained network, its eigenvalues determine the values of hyper-parameters, and its determinant is used to evaluate the model evidence (Nasrabadi, 2007).

Again, the second term in the marginalisation is a quadratic term in the model weights. For the particular case of the logistic regression one obtain:

- For additive noise with variance $\lambda$

$$\begin{aligned}\widetilde{R}_{emp} &\simeq R_{emp} + \frac{\lambda}{2}\sum_{j=1}^{d}\left[\sum_{i=1}^{n} P(Y=1 \mid X=x_i,w)P(Y=0 \mid X=x_i,w)\right] w_j^2 \\ &= R_{emp} + \frac{\lambda}{2}\left[\sum_{i=1}^{n}\alpha_i\right]\|w\|_2^2 \,.\end{aligned} \tag{3.8}$$

  where $\alpha_i = P(Y=1 \mid X=x_i,w)P(Y=0 \mid X=x_i,w)$ is the derived term from the loss second order derivative following equation 3.7. It is worth noticing that $\alpha_i$ is maximal if $P(Y=1 \mid X=x_i,w) = P(Y=0 \mid X=x_i,w) = 1/2$ , that is if the model does not give confident predictions for the given observation $x_i$.

- For multiplicative noise with variance $\lambda$:

$$\begin{aligned}\widetilde{R}_{emp} &\simeq R_{emp} + \frac{\lambda}{2}\sum_{j=1}^{d}\sum_{i=1}^{n}\left[P(Y=1 \mid X=x_i,w)P(Y=0 \mid X=x_i,w)x_{i,j}^2\right] w_j^2 \\ &= R_{emp} + \frac{\lambda}{2}\sum_{j=1}^{d}\sum_{i=1}^{n}\alpha_i x_{i,j}^2 w_j^2 \,.\end{aligned} \tag{3.9}$$

  One can see that because of the new presence of $x_{i,j}^2$, each feature weight is now penalised individually for the first time. This also couples a penalisation by the feature norm and by the confidence of the model (translated by the presence of the $\alpha_i$ )

Table 3.2 summarises the previously derived regulariser forms (that is the additional term appearing in the new INI empirical risk) for the square and logistic loss functions (as an example of non quadratic loss) respectively for additive and multiplicative INI in the in the case of a linear model. One can see from 3.2 that if the normalisation factor of the regulariser that trades off the empirical risk objective and the regularisation objective (usually chosen by cross validation, see introduction) is not taken in account, then the only case corresponding to an interesting data dependent regularisation that differs from the $\ell_2$-norm regularisation is the case of multiplicative INI for non quadratic loss (such as the case of dropout or multiplicative Gaussian INI for logistic regression). This is one of the main reasons we later focus on such a scheme for this chapter and the following ones.

**Remark 5.** *These two particular instances of the INI approximation are not new. It has has been first derived over three decades ago in the context of additive Gaussian INI (Bishop, 1995a; Reed et al., 1995). It was however recently revisited in the case of multiplicative noise by several authors and leveraged to develop more understanding in particular of the dropout algorithm and its application in semi-supervised learning (Wager et al., 2013; van der Maaten et al., 2013).*

**Remark 6.** *However, this Taylor approximation has a few limitations on the theoretical level that are important to notice:*

- *This approximation is only valid when the quantity $w.(\nu(x_i, \delta) - x_i)$ is small. Even though this quantity is smaller when the noise variance is small, and even for a linear noising function such as additive or multiplicative noise, this quantity depends on the data and on the model weights $w$ that are not bound in principle. Helmbold and Long (2015)show for instance that for multiplicative INI, in particular* dropout*, and for logistic regression, the true regulariser can remain bounded when the weights go to infinity, and that it can be non-monotonic as individual weights increase from 0. We clearly do not have these properties with the Taylor approximation regulariser.*

- *Even if the approximation can be close to the expected INI risk, the structure of the function can change drastically for non-quadratic losses which makes the minimum of the true INI risk and the minimum of the Taylor approximation risk far from each others. We will see in the experiments that this gap will translate in a gap in generalisation performance between the Taylor and the stochastic approximation.*

- *This approximation will be the same for different multiplicative noise injection schemes with the same variance induced by the noising function, since it only depends on the second moment of the noisy data with respect to the noise variable. It will not therefore show some special properties, for example for* dropout *over other multiplicative noise schemes. It has been empirically shown indeed that multiplicative noise injection schemes do not result in the same accuracy (Srivastava et al., 2014; van der Maaten et al., 2013).*

*These reasons lead to the importance of searching for other approximations in the case of a particular INI such as* dropout*. We build and analyse a novel approximation in the section about dropout.*

Table 3.2: Form of different regularisers resulting from different loss functions and two instances of noising (additive and multiplicative) using previous notations. The particular and most interesting data-dependent regularisation is highlighted with a blue colour.

| **Loss** | **Additive INI** | **Multiplicative INI** |
|---|---|---|
| Square loss | $\lambda \left\|w\right\|_2^2$ | $\lambda \left[\sum_{i=1}^n \left\|x_i\right\|_2^2 / n\right] \left\|w\right\|_2^2$ |
| Logistic loss | $\frac{\lambda}{2} \left[\sum_{i=1}^n \alpha_i\right] \left\|w\right\|_2^2$ | $\frac{\lambda}{2} \sum_{j=1}^d \sum_{i=1}^n \alpha_i x_{i,j}^2 w_j^2$ |

## 3.4 Algorithms

In this section we will review proposed methods to minimise the noise injection empirical risk. The first method relies on a finite sampling of noisy examples (that is of the noise random variable), the second on a stochastic sampling of the noise during the gradient descent and the last method on the marginalisation derived in the previous section.

### 3.4.1 Finite sample approximation

As previously stated, if we consider a finite sample estimation of the new INI empirical risk defined in 3.2, by considering a set of noise samples $\delta_k, k = 1, \ldots, m$ drawn randomly from $\Delta$ and defining an empirical discrete distribution $\Delta_m$, the empirical risk to minimise becomes:

$$\widehat{\widetilde{R}_{emp,\Delta_m}(f)} = \frac{1}{m.n} \sum_{k=1}^{m} \sum_{i=1}^{n} \left[ L\left( f(\nu(x_i, \delta_k)), y_i \right) \right] . \tag{3.10}$$

This approximation also can be seen as sampling observations from $\widetilde{\mathcal{P}}$. Indeed, we can apply the weak law of large numbers to see that the new INI empirical risk converges to the expected INI risk just as the empirical risk converges to the expected risk. One can also leverage knowledge about the loss function (such as its bounds) and the noise distributions to study more precisely the gap between the empirical and the expected INI risks. Grandvalet et al. (1997) for instance propose a point-wise and a global control of the empirical INI risk fluctuations in many cases with assumptions on the loss function and the hypothesis search space. If one uses gradient descent for the minimisation of the empirical risk, the algorithm 1 below describes the finite sample approximation of the INI objective.

**Algorithm 1** INI Finite sample approximation

**Require:** Training set $(x_i, y_i)_{i=1,\ldots,n}$, initialisation $w_0 \in \mathbb{R}^d$, learning rate $\gamma > 0$, number of passes $n_{passes} \in \mathbb{N}$, noising function $\nu$, noise distribution $\Delta$, finite noise sample size $m$.

**procedure** BUILD AUGMENTED DATASET
  **for** $k = 1$ `to` $m$ **do**
    Sample $\delta \sim \Delta$
    **for** $i = 1$ `to` $n$ **do**
      $\widetilde{x_i}^{(k)} \leftarrow \nu(x_i, \delta)$
      $\widetilde{y_i}^{(k)} \leftarrow y_i$
    **end for**
  **end for**
**return** $(\widetilde{x_i}^{(k)}, \widetilde{y_i}^{(k)})_{i=1,\ldots,n;k=1,\ldots,m}$,
**end procedure**
**procedure** GD ON THE NEW DATA
  $w^0 \leftarrow w_0$
  **for** $t = 1$ `to` $n_{passes}$ **do**
    $w \leftarrow w - \frac{1}{m.n}\gamma \sum_{i=1}^{n} \sum_{k=1}^{m} \nabla_w L(w.\widetilde{x_i}^{(k)}, \widetilde{y_i}^{(k)})$
  **end for**
**return** $w$
**end procedure**

Although approaches that explicitly corrupt the training data are simple (in that they just replace training on the original dataset by training on an augmented dataset) and effective (in that control guarantees exist as $m$ grows), they lack elegance and the resulting model can show some instability for the same noise distribution $\Delta$ when its discrete equivalent $\Delta_m$ varies. One can indeed use different discrete estimates for the same noise distribution and Grandvalet et al. (1997) for instance shows that one can have a uniform control (with respect to the model and discrete distribution) over the difference between the empirical and the expected INI risks in very simple settings but that it does not seems to be obvious for larger model families (such as neural networks for instance). More importantly, this method comes with high computational and memory costs since the minimisation of the new empirical risk (the evaluation of the risk values and of its gradients) scales linearly in the number of corrupted observations, i.e. it scales as $\mathcal{O}(n.m)$. Moreover, data storage will be problematic for large values of $m$ which is the regime one is interested in, especially for large datasets which

are becoming more ubiquitous in machine learning. This makes stochastic approximation a more interesting and practical approach.

### 3.4.2 Stochastic approximation

We have seen in the introduction that stochastic approximation and in particular the stochastic gradient descent can be used in the case when computing the full gradient is computationally expensive (in the case of complex loss functions, complex models or a very large dataset). One can make again use of the stochastic gradient descent since we have again a stochastic estimate of the loss gradient and that we are interested in minimising the expectation of this loss. Since we know the distribution of noise to be injected, it is sufficient, in addition to picking an input sample $x_t$ with label $y_t$ from the training set, to sample a noise example $\delta^{(t)}$ from the chosen distribution $\Delta$ and take a gradient step. We will then have for each step $t$ :

$$w^{(t+1)} = w^{(t)} - \gamma_t \nabla_w L(f_{w^{(t)}}(\nu(x_t, \delta^{(t)})), y_t)\,.$$

Under the same conditions mentioned by Bottou (2010) for instance, and that mainly depend on the regularity of the loss function and the growth learning rates $\gamma_t$, the Robbin-Siegmund theorem (Robbins and Siegmund, 1971) again, ensures convergence towards $\widetilde{R}_{emp}$. Algorithm 2 below shows the exact procedure when using stochastic gradient descent along with INI along with a shuffling step for the training dataset (introducing the permutation $\pi$, see (Bottou, 2010)).

**Algorithm 2** INI stochastic approximation algorithm

**Require:** Training set $(x_i, y_i)_{i=1,\dots,n}$, initialisation $w_0 \in \mathbb{R}^d$, initial learning rate $\gamma_0 > 0$, learning rate decay $\beta > 0$, number of passes $n_{passes} \in \mathbb{N}$, noise distribution $\Delta$.

**procedure** INI SGD
  $w \leftarrow w_0$
  $t \leftarrow 0$
  **for** $iter = 1$ `to` $n_{passes}$ **do**
    $\pi \leftarrow$ random permutation of $[1, n]$ ▷ Shuffle training set
    **for** $i = 1$ `to` $n$ **do** ▷ (Mini-)batch also possible
      $\gamma_t \leftarrow \gamma_0/(1 + \beta t)$
      Sample $\delta \sim \Delta$
      $\widetilde{x_{\pi(i)}} \leftarrow \nu(x_{\pi(i)}, \delta)$
      $\widetilde{y_{\pi(i)}} \leftarrow y_{\pi(i)}$
      $w \leftarrow w - \frac{\gamma_t}{n} \nabla_w L(w.\widetilde{x_\pi(i)}, \widetilde{y_\pi(i)})$
      $t \leftarrow t + 1$
    **end for**
  **end for**
**return** $w$
**end procedure**

### 3.4.3 Marginalisation for quadratic loss

Another way to minimise the noise injection objective would be to directly compute the expectation of the empirical risk over the noise variable. In some cases, as for instance when the model $f$ is linear, we can perform a full marginalisation on the noise and thus directly minimise through algorithms such as gradient descent without having to sample from the noise distributions. We saw that this is possible for the quadratic loss :

Recall that for quadratic loss and linear models with weights $w$, noise $\delta \sim \Delta$ and noising function $\nu$ :

$$\widetilde{R}_{emp} = R_{emp} + \frac{1}{n}\sum_{i=1}^{n} w^\top \underset{\delta\sim\Delta}{\text{Var}}\left[\nu(x_i,\delta)\right] w\,.$$

The first term does not depend on the noise variable. So if one can marginalise the second term, the new empirical risk can be expressed deterministically and can thus be minimised by classical (or stochastic) gradient methods or directly solved if an analytical solution is known. Algorithm 3 shows a gradient descent approach of the obtained marginalised loss when the loss is quadratic. When the loss is not quadratic either one can rely on the Taylor approximation, as shown in 3.3.4, or the other minimisation approaches.

**Algorithm 3** INI with marginalised square loss for linear models

**Require:** Training set $(x_i, y_i)_{i=1,\dots,n}$, initialisation $w_0 \in \mathbb{R}^d$, learning rate $\gamma > 0$, number of passes $n_{passes} \in \mathbb{N}$, noising function $\nu$, noise distribution $\Delta$, finite noise sample size $m$.

**procedure** GD ON THE MARGINALISED LOSS
  $w^0 \leftarrow w_0$
  $\Omega \leftarrow \sum_{i=1}^{n}\sum_{i=1}^{n} \text{Var}_\Delta\left[\nu(x_i,\delta)\right]$ ▷ or estimate if not directly computable
  **for** $t = 1$ `to` $n_{passes}$ **do**
    $w \leftarrow w - \frac{\gamma}{n}\sum_{i=1}^{n} \nabla_w L_{\text{square}}(w.x_i, y_i) - 2\frac{\gamma}{n}\Omega w$
  **end for**
**return** $w$
**end procedure**

**Remark 7.** *For noising functions such as polynomial functions with degree $n$, the noise variance term can be easily expressed in terms of moments of the distribution $\Delta$ up to order $2n$. For more complicated functions one can perform a Taylor approximation on $\nu$ around the mean of $\delta$.*

## 3.5 The special case of dropout

In 2012, the ImageNet Large Scale Visual Recognition challenge was won by the University of Toronto team by a surprisingly large margin. In an invited talk at NIPS, Hinton (2012) credited the dropout training technique for much of their success. In this section, we review the dropout algorithm that was proposed by Hinton and Srivastava (Hinton et al., 2012b; Srivastava et al., 2014) as one of the most popular noise injection technique in deep learning today. Even though dropout was proposed as a noise injection method for all layers of a neural network, we justify why dropout for linear networks is a form of multiplicative Input Noise Injection. We then briefly review the literature related to the method and discuss its implications. More details about the special cases of dropout in linear models and multilayer neural networks will follow in the two following chapters.

### 3.5.1 The dropout algorithm

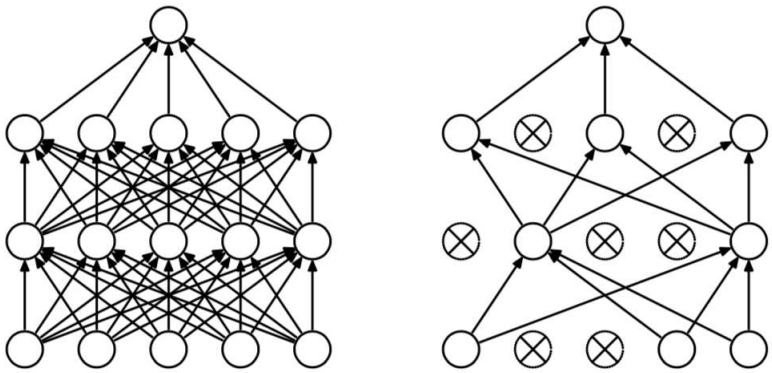
Consider a neural network with $L$ hidden layers. Let $l \in \{1,\dots,L\}$ index the hidden layers of the network. Let $z^{(l)}$ denote the vector of inputs into layer $l$, $y^{(l)}$ denote the vector of outputs from layer $l$ ($y^{(0)} = x$ is the input). $W^{(l)}$ and $b^{(l)}$ are the weights and biases at layer $l$. The feed-forward operation of a standard neural network can be described as (for $l \in \{1,\dots,L\}$ and any hidden unit $i$, and $f$ being any activation function) as,

$$\begin{aligned} z_i^{(l+1)} &= w_i^{(l+1)} y^{(l)} + b_i^{(l+1)} \\ y_i^{(l+1)} &= f(z_i^{(l+1)})\,. \end{aligned}$$

With classical dropout, the feed-forward operation becomes

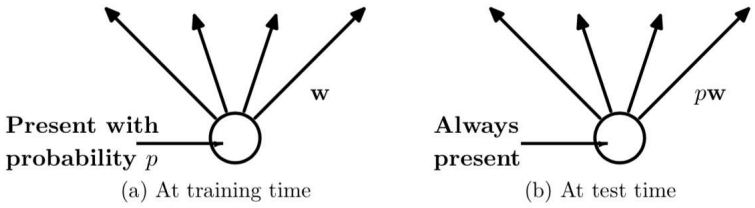
$$
\begin{aligned}
r_j^{(l)} &\sim \text{Bernoulli}(p) \\
\widetilde{y}^{(l+1)} &= r^l \odot y^{(l)} \\
z_i^{(l+1)} &= w_i^{(l+1)} \widetilde{y}^{(l+1)} + b_i^{(l+1)} \\
y_i^{(l+1)} &= f(z_i^{(l+1)}) \,.
\end{aligned}
$$

Here $\odot$ denotes an element-wise product. For classical dropout, $\mathbf{r}$ is a vector of independent Bernoulli random variables, each of which has probability $p$ of being 1. This process is applied at each layer. For learning, the derivatives of the loss function are back-propagated through the sub-network. At test time, the weights are scaled as $W_{test}^{(l)} = pW^{(l)}$ and no dropout is applied.



(a) Standard Neural Net (b) After applying dropout.

Figure 3.3: Dropout Neural Net Model: a standard neural net with 2 hidden layers (left figure) and an example of a thinned net produced by applying dropout to the original network (right figure). Figure from (Srivastava et al., 2014)




(a) At training time (b) At test time

Figure 3.4: Dropout at the unit level: a unit at training time that is present with probability p and is connected to units in the next layer with weights w (right figure). At test time, the unit is always present and the weights are multiplied by p (right figure). Figure from (Srivastava et al., 2014).

**Property 3.** *For linear models, dropout with probability p is equivalent to multiplicative INI where the noise $\delta$ is such that :* $\delta = \begin{cases} 1/p & \text{with probability } p \\ 0 & \text{otherwise} \end{cases}$

**Proof** For a linear network, dropout is applied on the linear activation $w.x$ through the stochastic gradient descent, and thus by the Stochastic approximation theorem (Robbins and

Siegmund, 1971), solves:

$$\begin{aligned}&\min_w \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta\sim\text{Bernoulli}(p)^d}[L(\delta \odot (w \cdot x_i)), y_i] \\ =&\min_w \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta\sim\text{Bernoulli}(p)^d}[L(w \cdot (\delta \odot x_i)), y_i] \ .\end{aligned}$$

By setting $\widetilde{w} = pw$ as the estimated used in the dropout trick at the test time, and absorbing the scaling factor in the Bernoulli variable, dropout solves:

$$\min_{\widetilde{w}} \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta\sim\text{Bernoulli}(p)^d}\left[L(\widetilde{w}.(\frac{\delta}{p} \odot x_i)), y_i\right] \qquad \square \ .$$

This property allows to see apply the regularisation intuitions and marginalisation techniques previously present for INI on dropout performed that is performed on activations in the case of linear models.

### 3.5.2 Related work

As presented previously, dropout follows the stream of ideas that appeared already three decades ago in the literature of neural networks, and that consists in adding noise in the input or the weights of multilayer networks (see introduction). The main novelties that dropout presents are that :

- The noise is injected at the unit activations, that is on the input of each layer of the network, instead of only the input layer or the weights.
- The noise takes the form of a multiplicative Bernoulli random variable instead of additive Gaussian noise.
- Instead of having a non biased noising procedure (as defined in the first section of this chapter), weights are instead multiplied by the Bernoulli distribution parameter at test time (presented in (Srivastava et al., 2014) as an approximate averaging procedure).

Since its introduction by Hinton et al. (2012b) and Srivastava et al. (2014), dropout has raised a lot of attention in the machine learning and especially deep learning community. The two original publications have been cited more than $10,000$ times and dropout became a classical technique in deep neural networks to improve their generalisation performance. One of the main reasons for this success is the set of extensive experiments in the original publications showing that dropout can help obtain the state-of-the-art performance on a range of benchmark data sets. Another reason is the simplicity of the trick that does not require changing the loss function, to add external prior knowledge nor to constrain the weights in a user dependent way. We can argue that a third reason is the mystery that surrounded and still partially surrounds dropout despite its practical simplicity and the many intuitions it has been justified with.

One can classify the literature around dropout into three categories :

- Theoretical understanding of the dropout properties.
- Design of variants of dropout.
- Applications of dropout or its variants to new problems or new models.

Although dropout has been introduced as an effective technique to prevent overfitting, Hinton et al. (2012b) only presented empirical results for this effectiveness, supplemented by two main intuitions. The first is that dropout can prevent complex *co-adaptations* on the

training data. The authors did not define the term *co-adaptations* and only argue that by the stochastic absence of other units, each hidden unit in the multilayer network will rely less on the other hidden units. The second intuition is that dropout is an efficient way of performing model averaging with neural networks, by stochastically sampling a thinned network at each weight update and then approximating the average prediction produced by all these networks (by scaling the weights at test time). In the second and more complete publication, these intuitions remain without theoretical justifications, the authors even venture to a comparative intuition that comes from the role of sex in evolution, mainly comparing units to genes that co-adapt less with sexual reproduction. This argument can be seen to be a particular case of encouraging diversity in ensemble learning, but was not backed up by theory or related work in the original publications.

This effectiveness of dropout in reducing overfitting raised many questions about the dynamics of its training, convergence and averaging properties at test time, that have been for instance listed by Baldi and Sadowski (2014) and that the community tried to answer in the following years. Baldi and Sadowski (2014) confirm for instance the ensemble averaging intuition in linear networks, and show how dropout can be approximated by normalised geometric means of subnetworks in the nonlinear case.

A lot of work has then focused on dropout for linear models, trying to understand dropout as noise injection in a simple case: van der Maaten et al. (2013) and Wager et al. (2013) use the second order Taylor approximation, that has been used previously in the case of Gaussian additive noise (Bishop, 1995a), in order to marginalise the loss over the multiplicative Bernoulli noise and conclude that dropout performs a kind of data-dependent regularisation. Wager et al. (2014) also show improved results when training with dropout in the case of document classification and later provides statistical learning bounds favouring dropout regularisation over $\ell_2$-norm regularisation for generative Poisson topic linear models. This finds generalisation by the same authors (Wager et al., 2016) to a particular noise injection where data is generatively drawn from Levy processes and noise from a conjugate distribution.

However, Helmbold and Long (2015) shows that the adaptive $\ell_2$-norm approximation of dropout for linear models is fundamentally false, and that for the logistic loss and linear models for instance, the dropout induced penalty can remain bounded even when a subset of the weight norms goes to infinity, that it can be non-monotonic as individual weights increase from 0, and that the penalty may not be convex. When extending their study for multilayer networks, the same authors surprisingly show that dropout training can lead to negative weights even when the output is a positive multiple of the inputs(Helmbold and Long, 2017). They also show that unlike weight decay and other $L_p$-norm regularisers, dropout training is insensitive to the rescaling of input features, and largely insensitive to rescaling of the outputs. It is also shown in the same paper that in the case of multilayer networks the dropout penalty can be negative, and that it depends on the labels, in contrast with the generalised linear model setting studied in (Wager et al., 2013). Other facets concerning the effect of dropout on the generalisation performance have been explored: McAllester (2013) present generalisation bounds of models trained with dropout using PAC-Bayesian theory. Gao and Zhou (2015) use the Rademacher complexity, a measure of the richness of the models class with respect to a probability distribution, in order to show that dropout can improve generalisation by an exponential reduction of this complexity (in terms of the number of layers) in contrast with the polynomial reduction of dropout in linear models or $\ell_2$ regularisation. Another line of work has focused on the interpretation of dropout as a Bayesian approximation, by seeing the multiplicative Bernoulli or Gaussian noise injected as a measure of the weight's uncertainty, which allowed to view another facet of dropout as a regularisation, but also to provide new principled variants and applications of the dropout method (Gal and Ghahramani, 2016a; Maeda, 2014).

The empirical success of dropout and the emergence of some answers about its properties has led to the development of many variants that try to boost the effect of dropout or improve some of its regularising aspects. The first variant termed *Dropconnect* appeared in (Wan et al., 2013), and proposed to apply dropout, that is multiplication by Bernoulli variables on the weights in contrast with the activations for the original method. The method was backed by a theoretical analysis of the Rademacher Complexity and showed improved results on benchmark vision recognition tasks. Another aspect of dropout that has been questioned was the choice the probability distribution of the Bernoulli variables, called *dropout rate*, empirically chosen to be around 0.5 for hidden layers in the original publications. A first set of methods propose theoretical heuristics to choose a nearly-optimal rate such as in (Zhai and Wang, 2018) which use the Rademacher complexity bounds to adaptively optimise the dropout rate during training or which introduce a binary belief network which is overlaid on a neural network to optimise the rate for each hidden unit through a similar back-propagation (Ba and Frey, 2013). Another line of work relies on the Bayesian interpretations of dropout in order to propose EM-like algorithms where dropout rates are optimised along with the model weights in alternate fashion (Kingma et al., 2015; Maeda, 2014).

Dropout was proposed in multilayer networks but its success have encouraged its adaptation to other types of models and in other settings. In (Chen et al., 2015), the authors adapt the dropout method the non-smooth hinge loss used in SVM methods. Gal and Ghahramani (2016b) use their Bayesian interpretation of dropout to provide a justified straightforward application in the case of recurrent neural networks. Variants for convolutional networks recently appeared and obtained improved results (Tompson et al., 2015; DeVries and Taylor, 2017). Another line of work tried to make sparse versions of dropout in order to perform simultaneously desirable feature selection (Li and Liu, 2016; Molchanov et al., 2017; Kang et al., 2018)

Of course, it is hard to keep pace of all the work that is related to dropout and that was applied in many subfields of machine learning. One can notice interestingly that a lot of mystery is still underlying the exact effect of the noise injection regularisation in neural networks in particular and machine learning methods in general (Rudi and Rosasco, 2017). Even with the emergence of other simple and effective regularisation methods that empirically reduced the need for dropout such as *maxout networks* (Goodfellow et al., 2013) and *batch normalisation* (Ioffe and Szegedy, 2015), dropout did not lose interest and is still widely used and inspiring competitive variants, and even inspiring more understanding of regularisation (Elisseeff et al., 2005) and deep learning optimisation (Hardt et al., 2016; Scardapane and Wang, 2017), which are two areas that lack a complete theory and are still full of open questions.

In the remainder of this chapter, we try to provide additional theoretical and empirical understanding of dropout: in the next section we indeed develop a simple yet insightful novel approximation of dropout, for general models, in order to illustrate original intuitions of the dropout method and motivate other variants and a generalisation of dropout. In the final section we provide experimental results, focusing on linear models (complementary experiments will follow in next chapters for multilayer and convolutional networks).

## 3.6 Another approximation for Dropout

We have seen that dropout, just as other multiplicative INI schemes, can be approximated in the case of linear models as an adaptive $\ell_2$-norm regularisation, that has its limitations. Recall that with dropout, we minimise:

$$\widetilde{R}_{emp} = \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta\sim\text{Bernoulli}(p)^d} \left[ L(f(\frac{\delta}{p} \odot x_i), y_i) \right] .$$

For simplicity of notations and of analysis, we will omit the scaling (one can just consider scaling the weights by $p$ as mentioned earlier). We can rewrite the empirical risk as:

$$\begin{aligned}\widetilde{R}_{emp} &= \frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{d}\left[\sum_{\delta^{(k)}\in\{0,1\}^d,\left\|\delta^{(k)}\right\|_0=d-k} L\left(f(\delta^{(k)}\odot x_i), y_i\right)\right](1-p)^k p^{d-k} \\ &= \frac{1}{n}\sum_{i=1}^{n}\sum_{k=0}^{d}\left[\sum L\left(f(\underbrace{x_{i,1},0,0,..,x_{i,j},0,\ldots,x_{i,d}}_{\text{"k"zeros}}), y_i\right)\right](1-p)^k p^{d-k}\,.\end{aligned}$$

When $p$ is close to 1, and if we approximate the loss up to order 1 around $(1-p)$, only the two first terms of the marginalisation remain. With the fact that $p^\alpha \simeq 1-\alpha(1-p)$, we obtain

$$\begin{aligned}R_{emp} = \frac{1}{n}(1-p)\sum_{i=1}^{n}\left[\sum_{j=1}^{d} L\left(f(x_{i,1},..,x_{i,j-1},0,x_{i,j+1},\ldots,x_{i,d}), y_i\right)\right] \\ + \sum_{i=1}^{n}(1-d(1-p))L(f(x_i),y_i) + o(1-p)\,.\end{aligned} \tag{3.11}$$

We have therefore obtained a new empirical risk $\widetilde{R}_{emp}$, that we can rewrite differently, replacing by the empirical risk:

$$\widetilde{R}_{emp} \simeq (1-d\cdot(1-p))R_{emp} + d\cdot(1-p)\left[\frac{\sum_{j=1}^{d} R_{emp\setminus j}}{d}\right] = \widetilde{R}_{emp}^{\text{approx}}\,, \tag{3.12}$$

where

$$R_{emp\setminus j} = \frac{1}{n}\sum_{i=1}^{n}\left[L\left(f(x_{i,1},..,x_{i,j-1},0,x_{i,j+1},\ldots,x_{i,d}), y_i\right)\right]\,.$$

Interestingly, $R_{emp\setminus j}$ appears as a new cost, representing the error made by the same model but with the deletion of one feature.

This approximation gives a new insight on the effect of dropout for general loss functions and models, as the new derived risk establishes a tradeoff between :

- The empirical risk on the original dataset
- The average of empirical risks on the dataset with the deletion of one feature (and all the others untouched)

The new empirical risk encourages therefore the model not to rely only on one feature for prediction. We can recover the intuition of the first paper about dropout as preventing feature co-adaptations (Hinton et al., 2012b). Even if there was no definition of co-adaptations in the original paper, one can define two features to be co-adapted if their mutual presence produces a small risk but the presence of one of them for the same model can incur a high risk. In other terms, one can understand dropout at the input level through this approximation as making the model accurate with respect to the original risk while being more robust to individual input feature deletions (with respect to the risk as well).

It is interesting to bound the difference between the new approximative risk and the original risk. In contrast with the quadratic approximation, the bound will be of first order in terms of the units activations:

**Property 4.** *With the previous notations and if the loss function is Lipschitz with constant $l$, the following inequality holds in the case of linear models: $f(x) = w \cdot x$ :*

$$\left|R_{emp} - \widetilde{R}_{emp}^{approx}\right| \leq l.(1-p)\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}|w_j x_{i,j}| \ . \tag{3.13}$$

**Proof** From equation 3.12, we get:

$$\begin{aligned}\left|R_{emp} - \widetilde{R}_{emp}^{\text{approx}}\right| &= (1-p)\left|\sum_{j=1}^{d}(R_{emp} - R_{emp\setminus j})\right| \\ &\leq (1-p)\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}|L\left(w \cdot x_i, y_i\right) - L\left(w \cdot x_i - w_j x_{i,j}, y_i\right)| \\ &\leq l.(1-p)\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d}|w_j x_{i,j}| \quad \square \quad .\end{aligned} \tag{3.14}$$

**Remark 8.** *As for the quadratic Taylor approximation, this new approximation provides another approach to minimise the expected INI risk using for example gradient descent on the approximated risk. One should keep in mind however that even the first approximation requires computing $d+1$ gradients computations at each iteration for each example, and thus might become prohibitive in high dimension. The bound in the last property also becomes weaker as the dimension grows. This approximation is in fact more a way to justify the use and understand dropout in a more general setting as an alternative to the Taylor approximation. However and as it will be shown in the simulation experiments, it may be a more promising approach also as a practical regulariser when compared to the Taylor approximation.*

**Remark 9.** *One can pursue the development of higher order terms in the expected dropout empirical risk which will actually contain terms of multiple features subsets. The next higher order term around $(1-p)$ of the expected loss function with respect to the i-th observation will indeed contain the terms :*

$$\sum_{j,k\in\{1,..,d\}} L\left(f(x_{i,1},..,x_{i,j-1},0,x_{i,j+1},\ldots,x_{i,k-1},0,x_{i,k+1},..,x_{i,d}),y_i\right) \ .$$

*This means that the new approximation of the dropout expected risk at the second order will take into account not only the contribution of individual features as the cost of removing each from the model, but also the cost of removing pairs of these features, in a new three-part tradeoff. If the presence of both features is much more important for the classification than the presence of each, as some synergy in the model between two features can take place, this second order approximation will better take in account this effect. We can see that dropout actually accounts for an exponential number of these interactions. And even if the way it accounts for these interactions through the loss function is somehow complex (depending heavily on the loss function, the model and the data structure), this sheds light to one of the reasons why dropout might be an important and popular trick in the recent literature.*

**Remark 10.** *When $p$ is close to 0, that is in the case of aggressive dropout, a new tradeoff will take place, between the risks of the model with one feature active only:*

$$\widetilde{R}_{emp} = \frac{1}{n}\sum_{i=1}^{n}\left[p\sum_{j=1}^{d}L\left(f(0,\ldots,x_{i,j},\ldots,0),y_i\right) + (1-d\cdot p)L\left(f(0_d),y_i\right)\right] \ . \tag{3.15}$$

*It seems that dropout will have again also a preference for models where all the features are important for the prediction, but now in a more direct way. That is not that the absence feature will incur an increase in the error, but that the presence of the feature will incur a decrease in the error.*

*To understand more what happens in this case case, one can look at the least squares with linear models after scaling the weights:*

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}\left[p\sum_{j=1}^{d}L\left(f(0,\dots,x_{i,j},\dots,0),y_i\right)+(1-d\cdot p)L\left(f(0,\dots,0),y_i\right)\right]\\
&=\frac{1}{n}\sum_{i=1}^{n}\left[p\sum_{j=1}^{d}\left(y_i-\frac{w_j x_{i,j}}{p}\right)^2+(1-d\cdot p)y_i^2\right]\\
&=\frac{1}{n}\sum_{i=1}^{n}\left[(y_i-w\cdot x_i)^2\right]+w^\top\left[\frac{diag(X^\top X)-2pX^\top X}{p}\right]w\,.
\end{aligned}\tag{3.16}$$

*For sufficiently small p, the matrix $diag(X^tX)-2pX^tX$ will be positive semi-definite (by continuity of the eigenvalues), we find the same kind of quadratic approximation in terms of the weights as in the Taylor approximation previously presented. However here if two features are correlated negatively, the weights are encouraged to be closer and if two features are correlated positively, the weights are encouraged to be different. This follows the intuition of making the model robust to feature deletion since it diversifies the weights if we have redundant features.*

## 3.7 Experiments and empirical insights

We focus on linear models. We perform a series of experiments on simulated and real data in order to assess the ability of noise injection in preventing overfitting as opposed to without noise injection or to other regularisations (such as $\ell_2$-norm regularisation).

### 3.7.1 Data with rare and useful features benefits from dropout

In order to test the intuition given by the second order Taylor approximation given in equation 3.9, that is that dropout logistic regression should perform well with rare but useful features, we build on the simulation setting by Wager et al. (2013), who however only compared the quadratic Taylor approximation of dropout and $\ell_2$-norm regularisation. In this study, we will also add to the comparison the stochastic approximation of dropout, the stochastic approximation of multiplicative noise injection and our new approximation to dropout. Wager et al. (2013) leverage the same quadratic approximation of multiplicative noise injection with logistic function in 3.9 to design a simulation where this penalty is small for the model weights that generated the data (see (Wager et al., 2013) for more details).

**Setting:** the simulation has 1050 features. The first 50 discriminative features form 5 groups of 10. The last 1000 features are purely noisy variable. Each example $(x_i, y_i)$ is independently generated as follows :

- Pick a group number $g \in 1,\dots,25$, and a sign: $sgn = \pm 1$.
- If $g \leqslant 5$, draw the entries of $x_i$ with index between $10(g-1)+1$ and $10(g-1)+10$ uniformly from $sgn \times e^C$, where C is selected such that $\mathbb{E}[(x_{,j})^2] = 1$ for all j, that is the features are scaled in terms of variance. Set all the other discriminative features to 0. If g > 5, set all the discriminative features to 0.
- Draw the last 1000 entries of $x_i$ independently from $\mathcal{N}(0,1)$

- Generate $y_i$ from the Bernoulli distribution with parameter $\frac{1}{1+e^{-x_i w}}$, where the first 50 coordinates of $w$ are 0.057 and the remaining 1000 coordinates are 0. The value 0.057 was selected to make the average value of $|x_i.w|$ in the presence of signal be 2.

**Training and evaluation:** For each simulation run, a training set of size n=75 is generated (cycling over the group number g deterministically). We train the different methods on 100 training datasets and test them on one test dataset with 10,000 examples (drawn using the same setting). As in the original simulation training, the penalisation parameters were set to roughly optimal values. For dropout, we used $p = 0.1$ while for $\ell_2$-norm regularisation we used $\lambda = 32$. After each training run of each method, we report the test classification performance on the test set as a measure of the AUC, that is the Area Under the ROC (Receiver Operating Characteristic) curve.

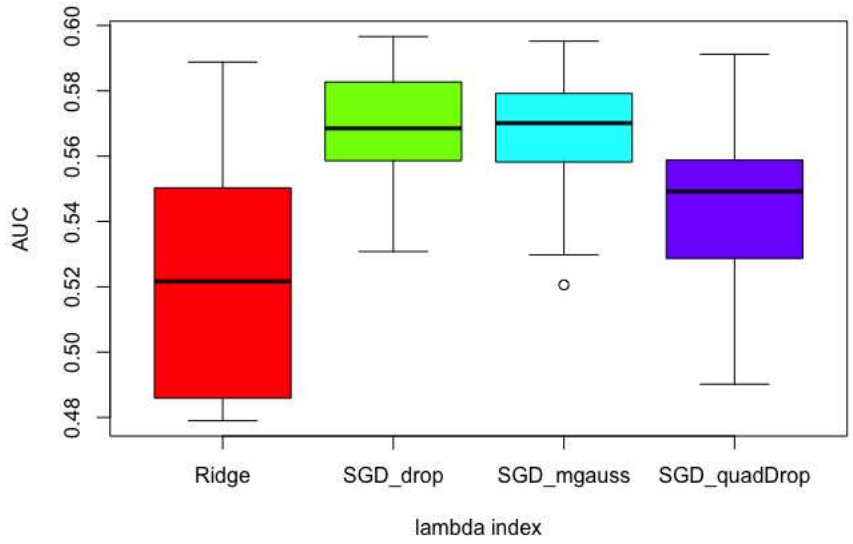



Figure 3.5: Box plots of test AUC on Wager simulation, with ridge (red), with dropout (green), multiplicative Gaussian (light blue) and the quadratic approximation of dropout (purple), over the 100 runs on the folds, taken with the best regularisation parameter.

The results are illustrated by figure 3.5 in terms of box plots of the test performance on the same test dataset of the 100 runs on each training set, for each linear model trained without noise injection, with ridge regularisation or with input noise injection. The figure first shows that for this setting quadratic approximation of dropout and the stochastic approximation of dropout are more effective than $\ell_2$ regularisation in the classification performance. We therefore verify these findings.

However, we can also see that multiplicative Gaussian noise provides the same improvement, which can be explained by the fact that the quadratic approximation for which this setting is favorable (Wager et al., 2013) leads to the same loss for all multiplicative INI methods as previously remarked.

Interestingly, the argument behind the success of dropout (and multiplicative Gaussian) in this simulation setting is that the regulariser will have small values for the best model. However, when we compare the quadratic approximation of dropout to stochastic dropout we see an additional gap in terms of accuracy. One reason for this difference could be the fact that the optimal dropout probability used is not close to 1 as required in the approximation, but this does not explain the better performance of the stochastic approximation. Another reason for this difference (and that was also previously hinted to) is that the quadratic approximation

for logistic loss is not faithful to the true dynamics of dropout even when the dropout rate is close to 1.

### 3.7.2 Our derived approximation is more effective than Taylor approximation

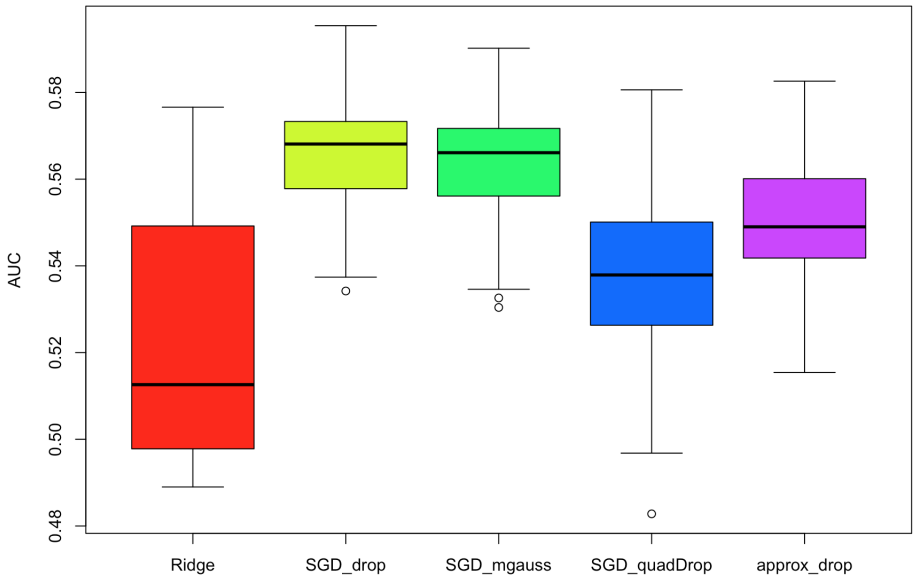



Figure 3.6: Box plots of test AUC on Wager simulation, with ridge (red), with dropout (yellow), multiplicative Gaussian (green), the quadratic approximation of dropout (blue) and our new approximation (purple), over the 100 runs on the folds, taken with the best regularisation parameter.

We repeat the same simulation experiments but adding now our new and simple approximation of dropout, in order to compare it with the stochastic and the Taylor approximations of dropout on a setting where these are effective. Figure 3.6 shows the test accuracy in terms of AUC evaluated on the same generated test set across the different methods. First, we can notice that our novel approximation performs better than the quadratic approximation for logistic loss and the same optimal parameter. Second, that it therefore leads to a smaller gap with respect to the stochastic approximation in terms of test classification, which can be explained in particular by the fact that it does preserve the dynamics of the loss function.

### 3.7.3 Data with redundant features benefits from dropout

Our approximation of dropout in the general setting shows that dropout prefers models that diversify their weights in the case of redundant features. We use the classical simulation setting proposed by Guyon et al. (2007) for the MADELON dataset to study the effect of feature redundancy on different INI schemes and also compare additive and multiplicative noise injection on a basic level. The simulation is built as follow

- Each one of the 2 classes to be predicted is composed of two Gaussian clusters. $\mathcal{N}(0, 1)$ is used to draw for each cluster useful examples of independent features.
- Some covariance is added by multiplying by a random matrix $A$, with uniformly distributed random numbers between -1 and 1.
- The two clusters are then placed at random on the vertices of a hypercube in a $d_{use}$ useful features dimensional space.

- $n_{red}$ redundant features are added by multiplying the useful features by a random matrix $B$, with uniformly distributed random numbers between $-1$ and 1.
- Useless features (random probes) are added using $\mathcal{N}(0, 1)$.
- All features are then shifted and rescaled randomly to span 3 orders of magnitude.
- Random noise is then added to the features according to $\mathcal{N}(0, 0.1)$
- A fraction flip $y$ of labels are randomly exchanged.

As for the training we generate, for each run, 100 samples for each run of training and 10,000 validation samples with balanced labels. We train a linear model using the logistic loss on the same training set again using (1) without INI, (2) additive Gaussian INI with different values of $\lambda$, (3) Multiplicative Gaussian INI with different values of $\lambda$ (10 values regularly spaced after log transform between $\lambda_{\min} = 10^{-8}$ and $\lambda_{\max} = 1$) and (4) dropout INI with different values of $p$ that corresponds to $\frac{\lambda}{\lambda+1}$, in order to make the comparison fair, since with the quadratic loss this corresponds to the same regularisation intensity (since it corresponds to the same noise variance) . We fix the total number features to $1,000$ and useful features to 100. We vary the number of redundant features $n_{red}$, in order to study the performance of the different INI methods with respect to this parameter.

Table 3.3: Average classification accuracy of linear models without noise injection without INI and with different INI schemes (with the best regularisation hyper-parameter) on the MADELON simulation with 10% useful features, varying the number of redundant features:

| **MADELON** | **Without INI** | **Additive Gaussian** | **Multiplicative Gaussian** | **Dropout** |
|---|---|---|---|---|
| $n_{red} = 0$ | 80.2 ±1.7 % | **80.9** ±1.9 % | 77.5 ±4.1 % | 77.55 ±4.2 % |
| $n_{red} = 400$ | 79.4 ±1.3 % | 80.3 ±1.5 % | **81.0** ±1.4 % | 80.2 ±1.4 % |
| $n_{red} = 800$ | 80.2 ±3.7 % | 80.5 ±3.0 % | 81.7 ±3.2 % | **82.2** ±3.3 % |

Table 3.3 shows classification accuracy averaged over 10 training runs. The test accuracies for the INI regularised methods are compared for the best regularisation parameter. A first observation that comes from 3.3 is that in the setting without redundant features, additive Gaussian performs best. In this setting, adding even a small amount of multiplicative noise hurts the performance and thus the best multiplicative INI would be with $\lambda = 0$. When adding redundant features however, the performance of a linear model without regularisation and with additive Gaussian INI is approximately the same but the performance of a linear model with multiplicative INI increases which can be justified by the intuition of dropout reducing co-adaptations between the features, and thus probably succeeding in extracting more information from the added redundant features.

### 3.7.4 Dropout is not always an effective regularisation

With most of dropout and noise injection experiments being shown in the deep learning setting, we want to see if INI is already efficient in simpler settings as for linear models.

#### Document classification tasks

We also study the performance of linear models with additive and multiplicative INI on document classification tasks. In this case, we use 2 datasets publicly available: IMDB and Reuters reviews (see supplementary table A.1). The labeled and already processed IMDB Movie Reviews dataset consists of 50,000 IMDB movie reviews, specially selected for sentiment analysis. The sentiment of the reviews is binary, meaning an IMDB rating less than 5 results in a sentiment score of 0, and a rating greater than 7 has a sentiment score of 1. No individual

movie has more than 30 reviews (Maas et al., 2011). The data is equally partitioned in a training and a test dataset. The Reuters newswire topics dataset consists of 11,228 newswires from Reuters, labeled over 46 topics. As with the IMDB dataset, the data is processed such that each wire is encoded as a sequence of word indexes (same conventions as Reuters). The data is partitioned into 8,982 training samples and 2,246 test samples. As for the MADELON simulation, we train a linear model using the logistic loss on the same training set, either (1) without INI, or (2) additive Gaussian INI with different values of $\lambda$, or (3) multiplicative Gaussian INI with different values of $\lambda$.

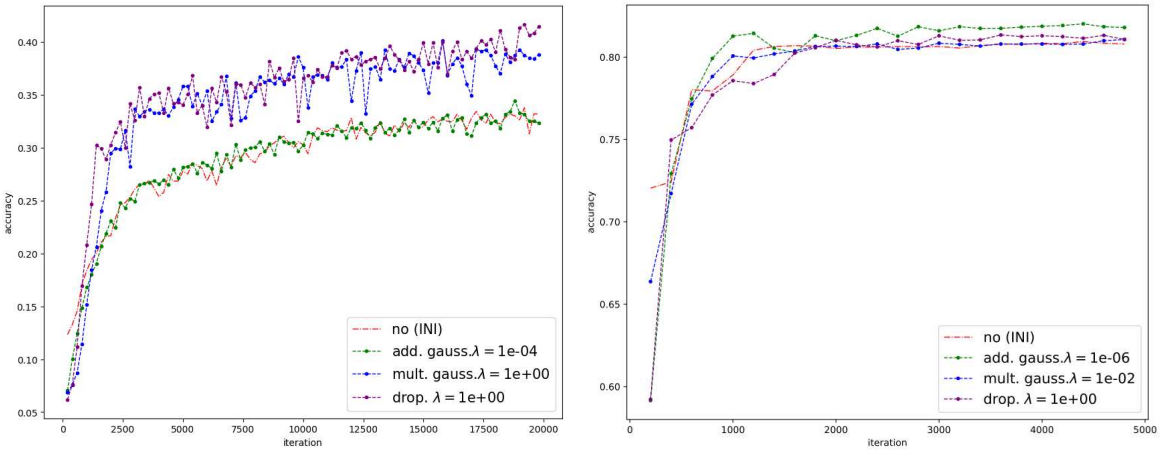



Figure 3.7: Test document classification accuracy on Reuters (left) and IMDB (right) during training, without INI and with different INI schemes (with the best regularisation hyper-parameter)

Table 3.4: Test accuracy on document classification, without INI and with different INI schemes (with the best regularisation hyper-parameter)

| **Dataset** | **without INI** | **Additive Gaussian** | **Multiplicative Gaussian** | **Dropout** |
|---|---|---|---|---|
| IMDB | 81.1 ± 0.5 % | **82.2 ± 0.8 %** | 81.2± 0.5 % | 81.7± 0.5 % |
| Reuters | 31.9 ± 1.0% | 32.1 ± 1.5 % | 37.5± 1.0% | **37.6 ± 1.0 %** |

Table 3.4 and figure 3.7 show the evolution and final value of test accuracy results of linear models with multiplicative, additive and without INI in IMDB and Reuters datasets. For the IMDB dataset, it is interesting to see that additive gaussian INI has the best test accuracy which first emphasises that there is some overfitting even in the case of a linear model, and that additive INI is in this case more effective than multiplicative INI, although we didn't try different distributions of noise. For Reuters however, it is clear that multiplicative INI has an advantage and this improvement is seen both for Gaussian and Bernoulli distributions again, which might hint to the fact that in most settings, the noising function is more important than the distribution of the noise in terms of generalisation performance. We leave this for further investigations.

Although both datasets were processed and indexed in a similar manner (encoded as a sequence of word indexes), INI effectiveness varies considerably between these 2 datasets which indicate empirically that the regularisation properties are more intricate than the nature of the task or loss function, even for a linear model. In fact, previous work on dropout and document classification (Wager et al., 2013; van der Maaten et al., 2013) only compared the addition of dropout versus no INI showing that dropout in linear models was more performant in most cases than no regularisation. It is not clear however if multiplicative noise injection is the best noising scheme for document classification.

### Image classification

We now turn to experiments on three main image classification benchmark datasets (Krizhevsky, 2009):

- **MNIST** handwritten digits dataset, which consists of $28 \times 28$ tiny grey digit images from 10 classes. It has a training set of $60,000$ examples, and a test set of $10,000$ example with balanced classes.
- **CIFAR-10**: consists of $32 \times 32$ tiny colour images from 10 classes . The dataset is divided into $60,000$ training samples with $6,000$ images per class and $10,000$ test samples. The test dataset contains exactly $1,000$ randomly-selected images from each class. Data is flattened such that the dimension of a sample is $32 \times 32 \times 3 = 3072$
- **CIFAR-100**: this dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. There are 500 training images and 100 testing images per class.

Although dropout has been studied in the linear model regime, studies have focused on document classification. van der Maaten et al. (2013) perform experiments with linear models on CIFAR10, however the data is transformed using bag-of-visual-words features (that is basically treating image features as words by creating a vector of occurrence counts of a vocabulary of local image features) following (Coates et al., 2011). We use here the classical pixels format of the image datasets (Krizhevsky, 2009). We perform the same testing experiment with the different already mentioned methods.

Table 3.5: Test accuracy on document classification, without INI and with different INI schemes (with the best regularisation hyper-parameter)

| **Dataset** | **without INI** | **Additive Gaussian** | **Multiplicative Gaussian** | **Dropout** |
|---|---|---|---|---|
| MNIST | 92.7 ± 0.1 % | 92.7± 0.1 % | 92.8 ± 0.1 % | 92.7 ± 0.1 % |
| CIFAR10 | 40.0± 0.1 % | 39.9 ± 0.1 % | **40.3** ± 0.1 % | **40.3** ± 0.1 % |
| CIFAR100 | 16.11 ± 0.1 % | 16.1 ± 0.5 % | 16.0 ± 0.2 % | 15.9 ± 0.2 % |

Table 3.5 and figure 3.8 show the evolution and final value of test accuracy results of linear models with multiplicative, additive and without INI on the different datasets. The main observation that seems at first surprising is that INI here doest not seem to improve generalisation performance in the case of these benchmark image datasets, where dropout and multiplicative gaussian noise have showed better results than without regularisation in the case of multilayer networks (Srivastava et al., 2014).

To investigate more why input noise injection does not help generalisation in the case of image classification with linear models, that is if INI methods are not effective here or that regularisation is not needed here in the case of linear models, one can both either include other regularisations methods (which might be ineffective too), or subsample the training data. We do both by retraining a randomly subsampled training set from MNIST with 100 and $1,000$ observations and comparing the average performance of different INI schemes with no INI and with $\ell_2$ regularisation. We perform 100 runs over randomly subsampled MNIST with balanced labels. The different methods are then evaluated on the same test dataset of $10,000$ images.

The average classification accuracy results are reported in table 3.6. Figure 3.9 shows the evolution of the same classification accuracy through the training iterations for the different methods, taken with the best regularisation parameter.

The first observation that stems from table 3.6 is that classification accuracy decreases for all methods as the number of training samples decreases. As previously noticed, the training accuracy for the different methods, including $\ell_2$-norm regularisation are not significantly

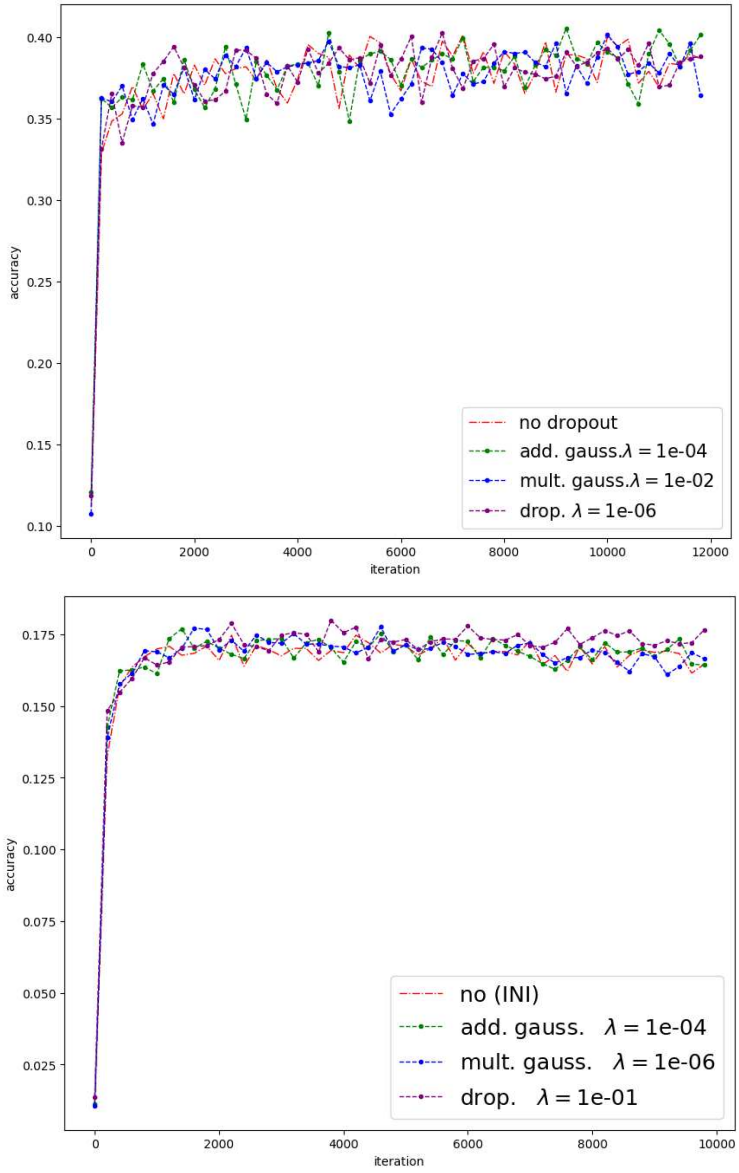



Figure 3.8: Test accuracy on CIFAR10 (top) and CIFAR100 (bottom) during training, without INI and with different INI schemes (with the best regularisation hyper-parameter).

Table 3.6: Best average test classification score for a linear model without any regularisation, with $\ell_2$ regularisation, additive Gaussian INI, multiplicative Gaussian INI and dropout (with the best regularisation parameters) on MNIST original and subsampled dataset.

| **Dataset** | **without INI** | **$\ell_2$-norm (ridge)** | **Add. Gauss.** | **Mult. Gauss.** | **Dropout** |
|---|---|---|---|---|---|
| MNIST: all samples | 92.72 ± 0.05 % | **92.75** ± 0.06 % | 92.74± 0.10 % | **92.75** ± 0.06 % | 92.74 ± 0.06 % |
| MNIST: 1,000 samples | 86.99 ± 0.50 % | 86.89 ± 0.30 % | 87.02± 0.08 % | 87.14 ± 0.09 % | **87.37** ± 0.09 % |
| MNIST: 100 samples | 68.67 ± 0.10 % | 68.75 ± 0.50 % | 69.02± 0.50 % | 70.92 ± 0.52 % | **70.93** ± 0.65 % |

better than the linear model without regularisation. This strongly hints that the linear model does not overfit on the MNIST dataset. Indeed the number of parameters, equal to the data dimension: $28 \times 28 = 784$ is 2 orders smaller than the number of samples: $50,000$.

As the number of samples is reduced, multiplicative Gaussian and Bernoulli INI start to perform better than additive Gaussian INI, $\ell_2$-norm regularisation and the linear model without regularisation, in terms of classification accuracy. This difference of performance grows as the number of training samples decreases, which indicates that the linear model already overfits with $1,000$ samples and this overfitting is more effectively prevented by multiplicative INI schemes. This is confirmed by figure 3.9 as we see, for the last figure corresponding to training on 100 samples of the dataset, that the test accuracy of multiplicative INI still growing slightly whereas it reaches a top and decreases for additive INI, $\ell_2$-norm and no INI models.

### Cancer prognosis

Dropout and other INI variants have been used and empirically studied mainly on document classification tasks. We further study here the performance of dropout on a type of bioinformatics data, namely gene microarrays.

Microarray technology has become one of the indispensable tools that many biologists use to monitor genome wide expression levels of genes in a given organism. A microarray is typically a glass slide on to which DNA molecules are fixed in an orderly manner at specific locations called spots (or features). Microarrays are used to measure gene expression in patients or cell DNA. We use here tumour gene microarrays in order to evaluate the predictive performance of linear models, without or with INI or other regularisation methods as $\ell_2$-norm and $\ell_1$-norm regularisation, in classifying patients with and without metastasis.

We use the Van't Veer breast cancer (VANT) data set from (Van De Vijver et al., 2002), and on the WANG dataset from (Wang et al., 2005), both restricted to $8,141$ genes, by Chuang et al. (2007). The Van't Veer set contains 295 tumors, split into 78 metastatic and 217 non-metastatic ones, while the Wang dataset contains 286 tumors among which 106 are metastatic (see supplementary table A.1 for details). The observations (or data rows) are the patient samples and the covariates (or data columns) will be the selected genes. For both datasets, we perform a 5-fold cross-validation and average the results of the test set partition over the 10 runs for each method. Figure 3.10 shows the cross-validation classification performance in terms of AUC box plots of the different models on each of the datasets.

We first see that $\ell_2$-norm, $\ell_1$-norm regularisations and INI perform better than the linear model without regularisation for both datasets, which means that even a linear model is overfitting in this case, which is understandable given the ratio between the dimension and the number of samples for these datasets ($\approx 30$). We can see that for VANT dataset, $\ell_2$-norm regularisation (ridge) performs best. In WANG dataset, dropout, $\ell_2$-norm and $\ell_1$-norm regularisation perform equally well, with a slight advantage for the $\ell_2$-norm regularisation. Notice the higher variance of the $\ell_1$-norm regularisation (LASSO) as a consequence for the model instability, leading also to unstable feature selection, has been already identified for instance in (Haury and Vert, 2010).

One intuition why dropout is not more effective than $\ell_2$ regularisation here, if one relies on the data augmentation intuition previously discussed, might be the complexity of the hybridisation noise associated with gene expression microarrays data (Tu et al., 2002). From an invariance point of view, if a subset of gene expression values are set to 0 with other genes having the same expression this might change the phenotype or the class of the observation (here metastatic or not) as the deregulation and mutations in only some driver genes are already known to present genetic risk factors and facilitate metastasis (Minn et al., 2005; Bos et al., 2009).

In the next chapter, we develop a predictive and variable selection model for another more recent type gene expression analysis data, where an inherent sequencing noise can justify a

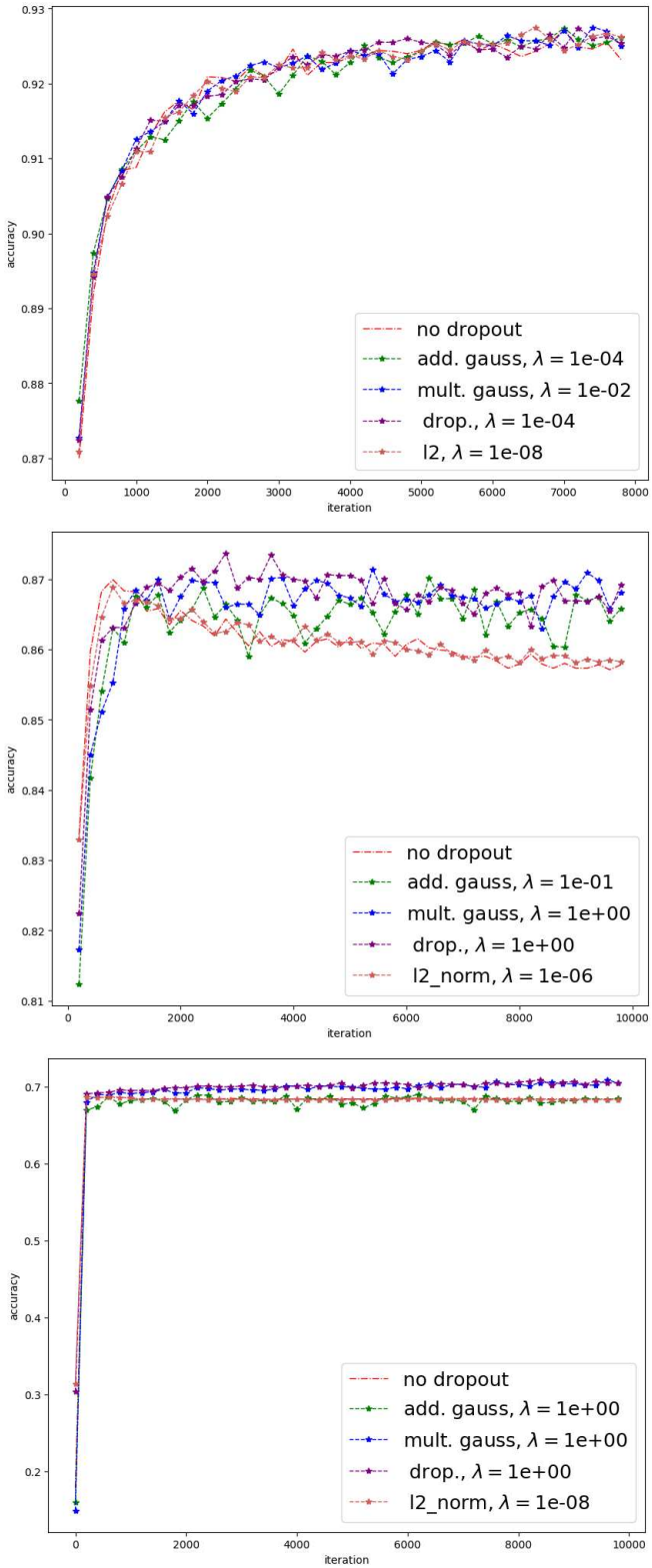



Figure 3.9: Average test accuracy on the originall (top) and subsampled MNIST dataset (middle: to 1,000 samples and bottom: to 100 samples) during training, without INI and with different INI schemes (shown for the best regularisation hyper-parameter)

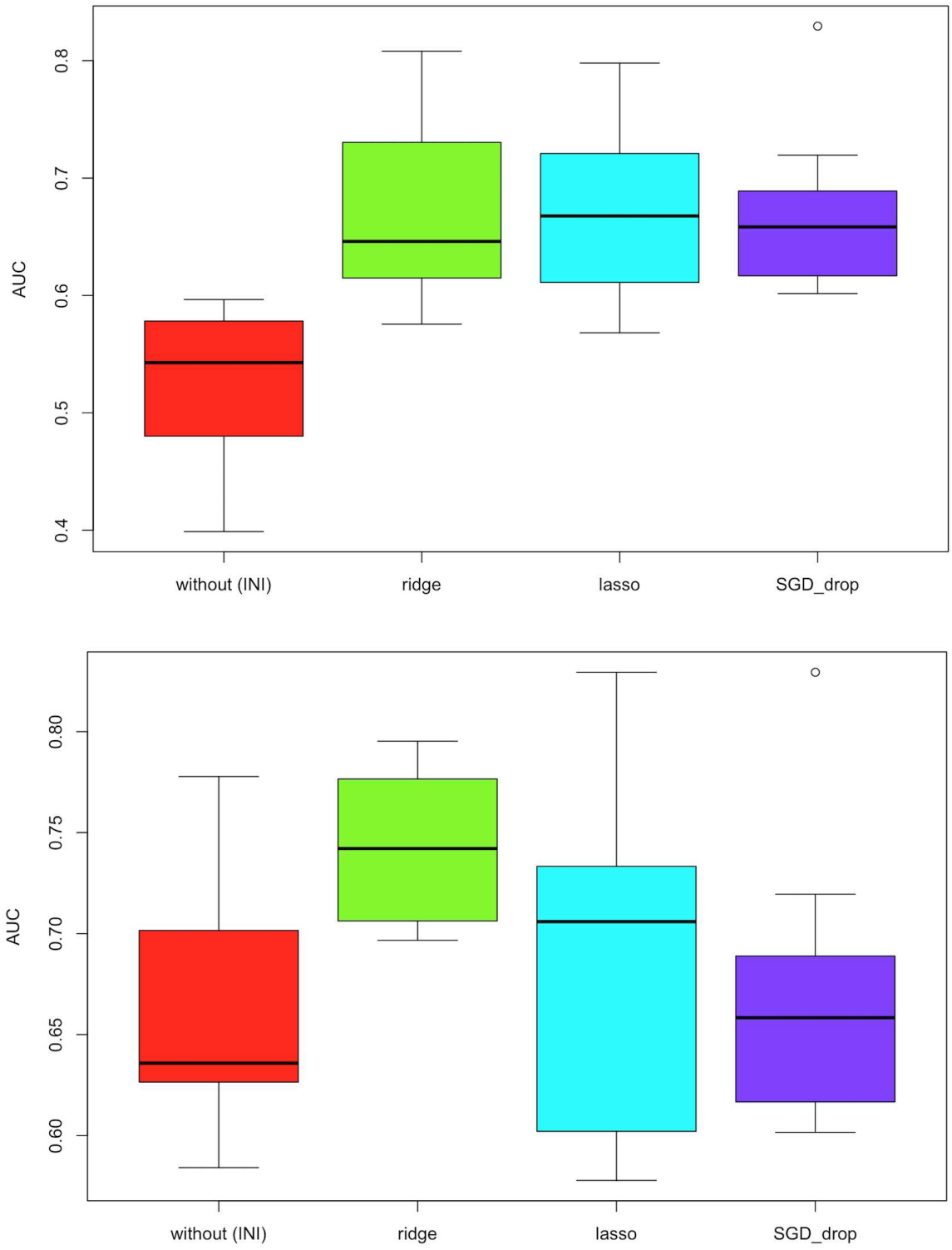



Figure 3.10: Cross-validation AUC box plots without INI and with different INI schemes with the best regularisation hyper-parameters, on WANG (above) and VANT (below) datasets.

data augmentation with multiplicative Bernoulli noise.

### 3.7.5 Dropout does not produce sparse models

As presented in the introduction, producing a sparse model such as $\ell_1$ regularisation is a nice feature since it allows for feature selection along with generalisation improvement. Dropout has been reported to sparsify the activations of units in case of nonlinear activations (Srivastava et al., 2014), as we will also confirm in the last chapter of this thesis.

Here, we want to know if a linear model regularised with additive or multiplicative INI can provide feature selection in the case of a classification task where the true underlying model is sparse. Indeed, we know that for a regression task with least squares at least following equations 3.5 and 3.6, INI can be exactly translated to an $\ell_2$-norm regularisation which, as previously presented in the introduction, does not perform feature selection in general. We look back here at the MADELON simulation experiments, with the particular case where we have 10% informative features and 40% redundant features, and thus half of the features are nuisance. As the experiments of training the different INI schemes was already performed, we visualise the distribution of the average weights at the end of training with or without INI in figure 3.11. For additive and multiplicative INI the distribution is visualised for different regularisation parameters $\lambda \in \{10^{-2}, 10^{-1}, 1\}$

No method provided sparse weights. On the contrary, a first observation that stems from figure 3.11 is that the distributions of weights resulting from the training of a linear model with multiplicative INI have a higher variance than the distribution of the weights without INI for all visualised values of noise variance $\lambda$. This comes to contradict the logistic regression and general loss INI approximations 3.8 and 3.9 that hint to a weight shrinkage in the case of a small noise variance and linear model. Interestingly, the distribution has a higher variance for multiplicative INI as the noise variance increases. For additive Gaussian INI however, the distribution of the weights is relatively stable and quite close to that without INI. We have seen that in this setting through table 3.3, multiplicative INI methods perform slightly better than additive Gaussian INI, which indicates that multiplicative INI does not work better by shrinking the weights (or at least not only), which was also shown in (Helmbold and Long, 2015).

This study also confirms the difference between the dropout mechanism and its Taylor approximation version as already noticed since multiplicative INI does not generally reduce the weights intensity, but also confirms that the stochastic version of dropout does not generally perform variable selection. The result shown in (Srivastava et al., 2014) where dropout led to sparser activations was actually reported for ReLU activations of hidden units, which become zero as the scalar product between the weights and activations of the previous hidden layer units is negative. The fact that no similar results are shown for linear or sigmoid functions, and having in the mind the equivalence between ridge and multiplicative INI the least squares setting, might hint to the fact that our observation about absence of variable selection for multiplicative INI is more general. This will be a motivation for next chapters and will be looked at in these.

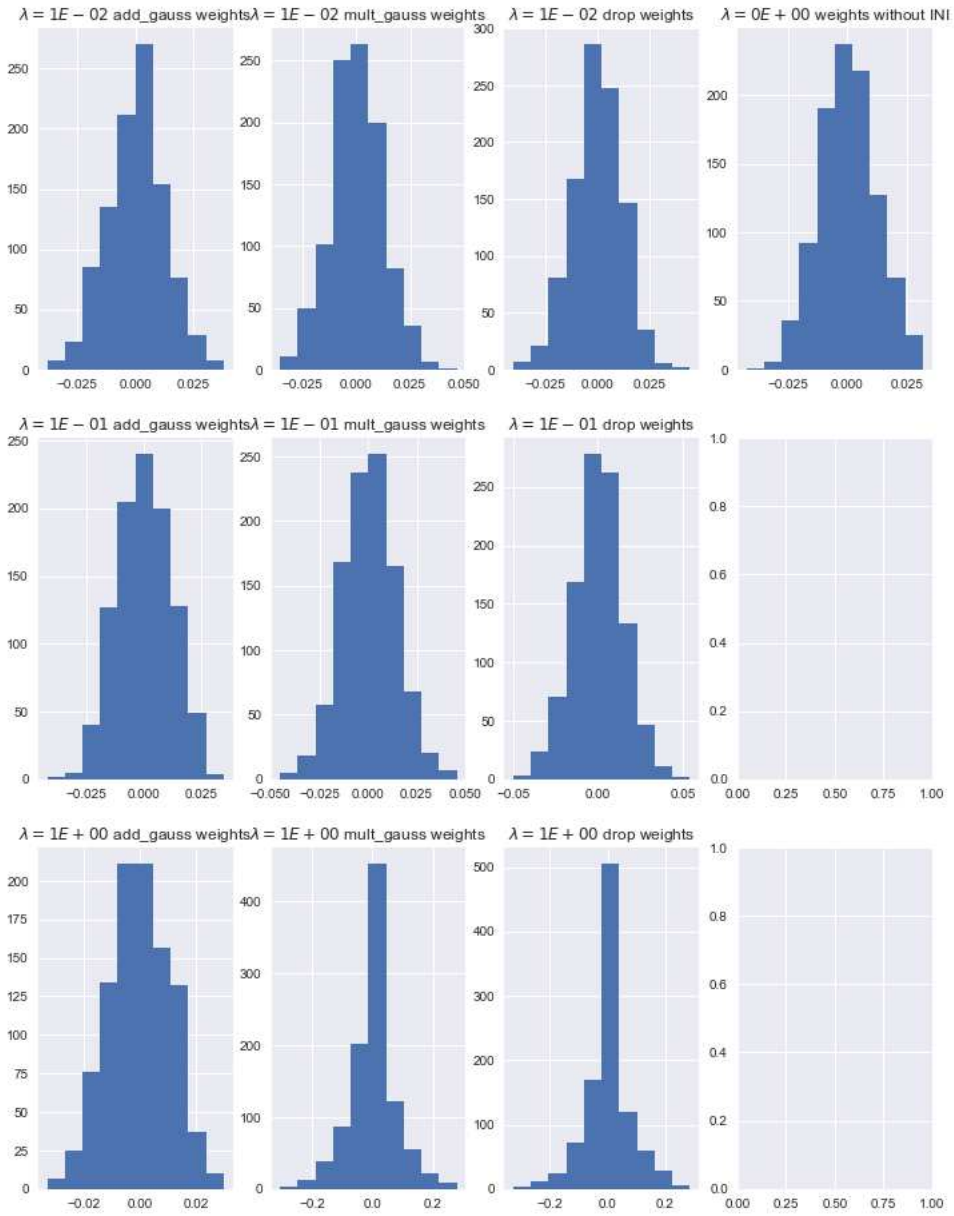



Figure 3.11: Histogram of the linear model weights learnt on MADELON after convergence (last training iteration) without INI and with different INI schemes for different values of $\lambda$.

## 3.8 Discussion and remarks

Adding stochastic noise to the input data, the model or the optimisation problem in the case of supervised learning has been widely used in different forms to improve generalisation performance, and has been adapted and interpreted in different frameworks in the last decades. With the growing complexity of datasets dimension and models complexity such as the number of parameters of neural networks, it led to a variety of ubiquitous tools that are constantly used in the field and even re-emerged as an active field of research through new variants of noise injection depending on different models and applications. However, a lack of a unified view and a clear comparison of the different proliferating techniques might be confusing and could even hinder understanding and development of the field. This chapter is only the beginning of an effort in this sense of a general theoretical and empirical study of Input noise injection INI as a general regularisation technique. After a brief overview on the subject, we have focused here on the ERM framework where we re-formalise INI as the minimisation of the expectation over the added noise of the empirical risk over the noisy samples. Although the framework covers general noise distributions and noising functions, we focused on additive and multiplicative Gaussian and Bernoulli INI. The exact and approximate second order marginalisation of the new empirical risk revealed important insights about regularisation properties of INI, but also some theoretical differences between additive and multiplicative INI. The studied second order approximation for general loss functions is a classical result - in the case of additive noise injection and Input dropout (that we briefly re-describe in this chapter)- that is however limited to a narrow setting (linear models) and can be sometimes even fundamentally misleading and does not lead to the same efficiency, especially for multiplicative INI. We derived another simple deterministic approximation for dropout that does not assume local quadrature of the loss and that is not limited only to linear models. Finally, we perform a set of simulations and real data experiments which have mainly indicated that :

- Multiplicative noise injection can be beneficial in the setting of rare and useful features and in the setting of many redundant features.
- Our novel approximation improves on the second order Taylor approximation in the case of linear logistic loss in terms of test accuracy in the previous setting.
- Dropout and multiplicative Gaussian INI are equally effective in improving the generalisation performance, in nearly all simulation and real data experiments
- INI is not always more effective than $\ell_2$-norm regularisation for preventing overfitting, and multiplicative INI is not always more effective than additive INI.
- No INI scheme provides sparse models even in a linear model setting where the true model is sparse.

An interesting direction for future work is first of all to understand new properties of the INI from a regularisation point of view and more broadly from a statistical point of view. Computational complexity and the design of faster methods have received some attention but deserves more analysis (Wang and Manning, 2013).

One way to strengthen our understanding of this inherently stochastic INI trick is to develop and study deterministic counterparts. An important future work will be to investigate more theoretically and empirically our proposed approximation to *dropout* as a deterministic alternative to the quadratic approximation. It will be first important to fully compare the theoretical properties of the different INI estimators from the different approximations in different special cases if the general case is quite complicated. In particular it will be interesting to see if our novel approximation inherits some of the main properties of dropout in linear and

multilayer networks such as non-convexity of the induced bias (Helmbold and Long, 2015, 2017).

Another direction opened by our approximation is the generalisation of the idea of robustness by modification of the loss function. Our approximation interprets dropout as emphasising robustness to feature deletion, but other cases can be considered such as the robustness by any kind of transformation of the model. This can contribute to design new generic risks that are tailored to specific datasets, as an alternative to data augmentation. An interesting recent work in this direction in the semi-supervised setting is proposed in (Sajjadi et al., 2016). The corresponding experimental datasets used in this chapter are listed in table A.1, and the corresponding code is available via `https://github.com/BeyremKh/INI-Experiments`.

# Chapter 4

# DropLasso: A robust variant of Lasso for single cell RNA-seq data

Contents

**Abstract**

Single-cell RNA sequencing is a fast growing approach to measure the genome-wide transcriptome of many individual cells in parallel, but results in noisy data with many dropout events. Existing methods to learn molecular signatures from bulk transcriptomic data may therefore not be adapted to scRNA-seq data, in order to automatically classify individual cells into predefined classes.

In this chapter, we propose a new method called DropLasso to learn a molecular signature from scRNA-seq data. DropLasso extends the dropout regularisation technique, popular in neural network training, to estimate sparse linear models. It is well adapted to data corrupted by dropout noise, such as scRNA-seq data, and we clarify how it relates to elastic net regularisation. We provide promising results on simulated and real scRNA-seq data, suggesting that DropLasso may be better adapted than standard regularisations to infer molecular signatures from scRNA-seq data.
DropLasso is freely available as an R package at https://github.com/jpvert/droplasso

**Résumé**

Le séquençage d'ARN monocellulaire est une approche d'une popularité grandissante permettant de mesurer le transcriptome du génome de nombreuses cellules en parallèle, mais qui aboutit à des données très bruitées avec de nombreux "dropout" ou expressions non détectées. Les méthodes existantes pour apprendre les signatures moléculaires à partir de données transcriptomiques plus classiques risquent donc de ne pas être adaptées aux données scRNA-seq, afin de classer automatiquement les cellules individuelles dans des classes prédéfinies.

Dans ce chapitre, une nouvelle méthode appelée DropLasso pour apprendre une signature moléculaire à partir de données scRNA-seq. DropLasso est une généralisation de Dropout, une technique de régularisation très utilisée dans les réseaux de neurones, dans le cadre de modèles linéaires parcimonieux. Nous justifions que cette régularisation est bien adaptée aux données corrompues par le bruit "dropout", telles que les données scRNA-seq, et nous clarifions son lien avec la régularisation elastic net. . Nous fournissons des résultats prometteurs sur des données scRNA-seq simulées et réelles, suggérant que DropLasso serait peut-être mieux adapté que les régularisations standard pour déduire des signatures moléculaires à partir de données scRNA-seq.
DropLasso est disponible ouvertement en tant que package R ici: https://github.com/jpvert/droplasso

## 4.1 Introduction

### 4.1.1 Single cell RNA-seq

In 1977 Frederick Sanger invented Sanger sequencing, a DNA-sequencing method that uses modified dideoxynucleotides to cause chain-termination (for which he was awarded his second Nobel prize in 1980). For the first time, we could read our genetic code. This method is however expensive and labor-intensive. Thus, it took 13 years and cost three billion dollars to complete the "Human Genome Project" in which essentially the entire human genome was sequenced by 2003 (filling in the gaps of the first draft that was published in 2001) [1]. Sequencing for an individual research project was not economically practical (microarrays were the choice of method) until next generation sequencing (NGS) methods were marketed in 2005, as it was mentioned in the introduction of this manuscript. NGS enables high-throughput sequencing by synthesis (SBS), reduces cost and increases feasibility. The recently marketed Nanopore device, the third generation of sequencers, will further push this limit. With sequencing costs continuing to drop, the first single-cell RNA sequencing method (scRNA-seq) was developed in 2009 (Tang et al., 2009). Single-cell RNA sequencing (scRNA-seq) is now a fast growing approach to measure the genome-wide transcriptome of many individual cells in parallel. Single cell sequencing allows to examine the sequence information from individual cells with optimised NGS technologies and have stood out in the last years as a promising source of biological discoveries (Kolodziejczyk et al., 2015).

Individual cells are the basic building blocks of organisms and each cell is unique. Performing bulk RNA sequencing often masks such uniqueness and fails to reveal latent changes. The fast paced development of massively parallel sequencing technologies and protocols has made it possible to measure gene expression with more precision and less cost in recent years.SscRNA-seq, in particular, is a fast growing approach to measure the genome-wide transcriptome of many individual cells in parallel (Kolodziejczyk et al., 2015). By giving access to cell-to-cell variability, it represents a major advance compared to standard "bulk" RNA sequencing to investigate complex heterogeneous tissues and reveal new cell types (Macosko et al., 2015; Tasic et al., 2016; Zeisel et al., 2015; Villani et al., 2017), study dynamic biological processes such as embryo development (Deng et al., 2014) and cancer (Patel et al., 2014), identify gene regulatory mechanisms (Jaitin et al., 2016; Xue et al., 2013) and reveal random patterns in allelic gene expression (Chen et al., 2016; Deng et al., 2014).

### 4.1.2 Motivation

Besides exploratory analysis and gene-per-gene differential analysis, a promising use of scRNA-seq technology is to automatically classify individual cells into pre-specified classes. In fact, as of today, the computational analysis of scRNA-seq data is dominated by unsupervised approaches in order to identify and molecularly profile hitherto unknown cell types. On the other hand, supervised approaches are likely to become more and more important, too, as they allow for a comparison of tissue compositions in terms of a priori known cell types. With more and more cell types being discovered and more and more tissues being profiled, this setting, where we aim at classifying a cell into one out of several known classes given its expression profile, is likely to become more and more relevant, in particular for small-scale studies. such as particular cell types in a cancer tissue. This can also allow to establish cell type specific "molecular signatures" that could be shared and used consistently across laboratories, just like standard molecular signatures are commonly used to classify tumour samples into subtypes from bulk transcriptomic data (Ramaswamy et al., 2001; Sørlie et al., 2001). From a methodological point of view, molecular signatures are based on a supervised analysis, where a model is trained to associate each genome-wide transcriptomic profile to a particular class, using a set of profiles with class annotation to select the genes in the signa-

[1] https://www.genome.gov/human-genome-project

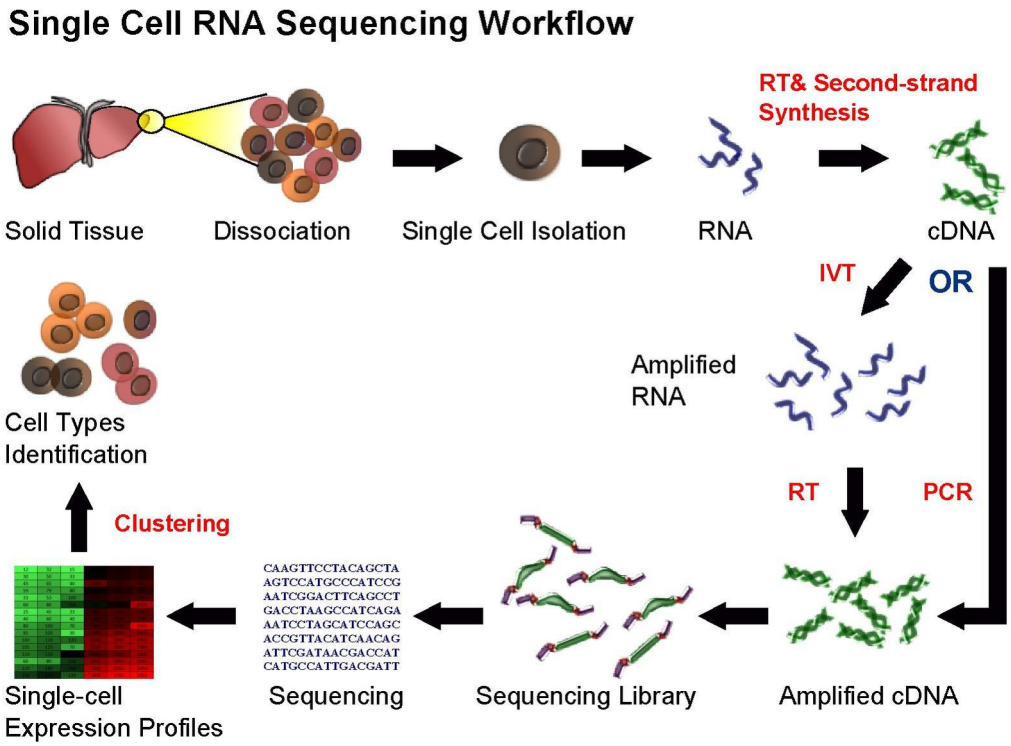



Figure 4.1: single cell RNA-seq workflow.

ture and fit the parameters of the models. While the classes themselves may be the result of an unsupervised analysis, just like breast cancer subtypes which were initially defined from a first unsupervised clustering analysis of a set of tumours (Perou et al., 2000), the development of a signature to classify any new sample into one of the classes is generally based on a method for supervised classification or regression.

The analysis of scRNA-seq data is however challenging and raises a number of specific modelling and computational issues (Ozsolak and Milos, 2011; Bacher and Kendziorski, 2016). In particular, since a tiny amount of RNA is present in each cell, a large fraction of polyadenylated RNA can be stochastically lost during sample preparation steps including cell lysis, reverse transcription or amplification (see figure 4.1 for the experimental workflow) . As a result, many genes fail to be detected even though they are expressed, a type of errors usually referred to as *dropouts*. In a standard scRNA-seq experiment it is common to observe more than 80% of genes with no apparent expression in each single cell (see figure 4.2 which represents the histogram of 3 gene expressions across sample cells in a later described real dataset), an important proportion of which are in fact dropout errors (Kharchenko et al., 2014). The presence of so many zeros in the raw data can have significant impact on the downstream analysis and biological conclusions, and has given rise to new statistical models for data normalisation and visualisation (Pierson and Yau, 2015; Risso et al., 2018) or gene differential analysis (Kharchenko et al., 2014). Imputation methods developed for bulk RNA-seq data may not be directly applicable to scRNA-seq data. First, much larger cell-level variability exists in scRNA-seq, because scRNA-seq has cell-level records for gene expression; on the other hand, bulk RNA-seq data have the averaged gene expression of the population of cells. Second, dropout events in scRNA-seq are not exactly missing values; dropout events have zero expression, and they are mixed with real zeros. In addition, the proportion of missing values in bulk RNA-seq data is much smaller.

Interestingly and independently, the term ”dropout” has also gained popularity in the machine learning community in recent years, as a powerful technique to regularise deep neu-

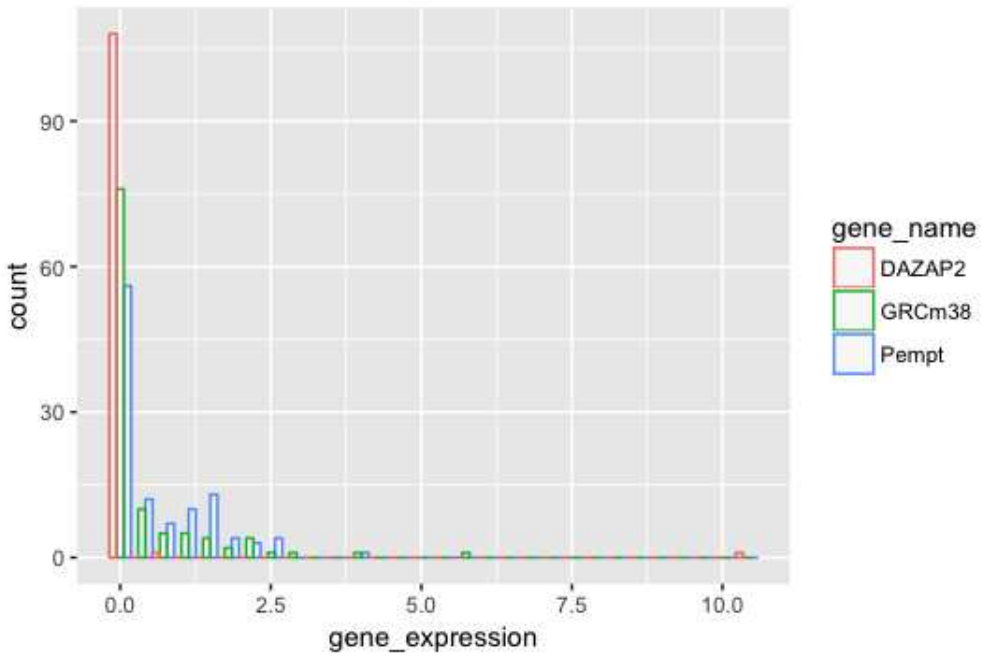



Figure 4.2: Histogram of 3 randomly sampled gene expressions in read counts across all cells (gene names in the legend) from a typical scRNA-seq dataset (GSE45719).

ral networks (Srivastava et al., 2014). Dropout regularisation works by randomly removing connexions or nodes during parameter optimisation of a neural network. On a simple linear model (a.k.a. single-layer neural network), this is equivalent to randomly creating some dropout noise to the training examples, i.e., to randomly set some features to zeros in the training examples (Wager et al., 2013; Baldi and Sadowski, 2013). Several explanations have been proposed for the empirical success of dropout regularisation. Srivastava et al. (2014) motivated the technique as a way to perform an ensemble average of many neural networks, likely to reduce the generalisation error by reducing the variance of the estimator, similar to other ensemble averaging techniques like bagging (Breiman, 1996a) or random forests (Breiman, 2001). Another justification for the relevance of dropout regularisation, particularly in the linear model case, is that it performs an intrinsic data-dependent regularisation of the estimator (Wager et al., 2013; Baldi and Sadowski, 2013) which is particularly interesting in the presence of rare but important features. Yet another justification for dropout regularisation, particularly relevant for us, is that it can be interpreted as a *data augmentation* technique, a general method that amounts to adding virtual training examples by applying some transformation to the actual training examples, such as rotations of images or corruption by some Gaussian noise; the hypothesis being that the class should not change after transformation. Data augmentation has a long history in machine learning (e.g., Schölkopf et al., 1996), and is a key ingredient of many modern successful applications of machine learning such as image classification (Krizhevsky et al., 2012). As shown by van der Maaten et al. (2013), dropout regularisation in the linear model case can be interpreted as a data augmentation technique, where corruption by dropout noise enforces the model to be robust to dropout events in the test data, e.g., to blanking of some pixels on images or to removal of some words in a document. Wager et al. (2014) show that in some cases, data augmentation with dropout noise allows to train model that should be insensitive to such noise more efficiently than without.

Since scRNA-seq data are inherently corrupted by dropout noise, we therefore propose that dropout regularisation may be a sound approach to make the predictive model robust to this form of noise, and consequently to improve their feature selection performance in scRNA-seq supervised classification for instance. Since plain dropout regularisation does not lead to feature selection and to the identification of a limited number of genes to form a

molecular signature, we furthermore propose an extension of dropout regularisation, which we call DropLasso regularisation, obtained by adding a sparsity-inducing l1 regularisation to the objective function of the dropout regularisation, just like lasso regression adds an $\ell_1$ penalty to a mean squared error criterion in order to estimate a sparse model (Tibshirani, 1996). We show that the l1 penalty can be integrated in the standard stochastic gradient algorithm used to implement dropout regularisation, resulting in a scalable stochastic proximal gradient descent formulation of DropLasso. We also clarify the regularisation property of DropLasso, and show that it is to elastic net regularisation what plain dropout regularisation is to the plain ridge regularisation. Finally, we provide promising results on simulated and real scRNA-seq data, suggesting that specific regularisations like DropLasso may be better adapted than standard regularisations to infer molecular signatures from scRNA-seq data.

**R** code for reproducing the experiments in this chapter can be found in https://github.com/BeyremKh/Droplasso-experiments.

## 4.2 Methods

### 4.2.1 Setting and notations

We consider the supervised machine learning setting, where we observe a series of $n$ pairs of the form $(x_i, y_i)_{i=1,\dots,n}$. For each $i \in [1, n]$, $x_i \in \mathbb{R}^d$ represents the gene expression levels for $d$ genes measured in the $i$-th cell by scRNA-seq, and $y_i \in \mathbb{R}$ or $\{-1, 1\}$ is a label to represent a discrete category or a real number associated to the $i$-th cell, e.g., a phenotype of interest such as normal vs tumour cell, or an index of progression in the cell cycle. For $i \in [1, n]$ and $j \in [1, d]$, we denote by $x_{i,j} \in \mathbb{R}$ the expression level of gene $j$ in cell $i$. From this training set of $n$ annotated cells, the goal of supervised learning is to estimate a function to predict the label of any new, unseen cell from its transcriptomic profile. We restrict ourselves to linear models $f_w : \mathbb{R}^d \to \mathbb{R}$, for any $w \in \mathbb{R}^d$, of the form

$$\forall u \in \mathbb{R}^d, \quad f_w(u) = \sum_{i=1}^{d} w_i u_i \,.$$

To estimate a model on the training set, a popular approach is to follow a penalised maximum likelihood or empirical risk minimisation principle and to solve an objective function of the form

$$\min_{w \in \mathbb{R}^d} \left\{ \frac{1}{n} \sum_{i=1}^{n} L(w, x_i, y_i) + \lambda \Omega(w) \right\}, \tag{4.1}$$

where $L(w, x_i, y_i)$ is a loss function to assess how well $f_w$ predicts $y_i$ from $x_i$, $\Omega$ is an (optional) penalty to control overfitting in high dimensions, and $\lambda > 0$ is a regularisation parameter to control the balance between under- and overfitting. Examples of classical loss functions include the square loss:

$$L_{\text{square}}(w, x_i, y_i) = \left( y_i - \sum_{j=1}^{d} w_j x_{i,j} \right)^2 ,$$

and the logistic loss:

$$L_{\text{logistic}}(w, x_i, y_i) = \log \left( 1 + \exp(-y_i \sum_{j=1}^{d} w_j x_{i,j}) \right) ,$$

which are popular losses when $y_i$ is respectively a continuous ($y_i \in \mathbb{R}$) or discrete ($y_i \in \{-1, 1\}$) label. As for the regularisation term $\Omega(w)$ in (4.1), popular choices include the ridge penalty (Hoerl and Kennard, 1970):

$$\Omega_{\text{ridge}}(w) = \|w\|_2^2 = \sum_{i=1}^{d} w_i^2 \,,$$

and the lasso penalty (Tibshirani, 1996):

$$\Omega_{\text{lasso}}(w) = \|w\|_1 = \sum_{i=1}^{d} |w_i| \,.$$

The properties, advantages and drawbacks of ridge and lasso penalties have been theoretically studied under different assumptions and regimes. The lasso penalty additionally allows feature selection by producing sparse solutions, i.e., vectors $w$ with many zeros; this is useful to in many bioinformatics applications to select "molecular signatures", i.e., predictive models based on the expression of a limited number of genes only. It is known however that lasso can be unstable in particular when there are several highly correlated features in the data. It also cannot select more features than the number of observations and its accuracy is often dominated by that of ridge. For these reasons, another popular penalty is elastic net, which encompasses the advantages of both penalties Zou and Hastie (2005) :

$$\Omega_{\text{elastic net}}(w) = \alpha \|w\|_2^2 + (1-\alpha)\|w\|_1 \,,$$

where $\alpha \in [0,1]$ allows to interpolate between the lasso ($\alpha = 0$) and the ridge ($\alpha = 1$) penalties.

### 4.2.2 DropLasso

For scRNA-seq data subject to dropout noise, we propose a new model to train a sparse linear model robust to the noise by artificially augmenting the training set with new examples corrupted by dropout. Formally, given a vector $u \in \mathbb{R}^d$ and a dropout mask $\delta \in \{0,1\}^d$, we consider the corrupted pattern $\delta \odot u \in \mathbb{R}^d$ obtained by entry-wise multiplication $(\delta \odot u)_i = \delta_i u_i$. In order to consider all possible dropout masks, we make $\delta$ a random variable with independent entries following a Bernoulli distribution of parameter $p \in [0,1]$, i.e., $P(\delta_i = 1) = p$, and consider the following DropLasso regularisation for any $\lambda > 0$, $p \in [0,1]$ and loss function $L$:

$$\min_{w \in \mathbb{R}^d} \left( \frac{1}{n} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta_i \sim B(p)^d} L(w, \delta_i \odot \frac{x_{i,}}{p}, y_i) + \lambda \|w\|_1 \right) . \tag{4.2}$$

In this equation, the expectation over the dropout mask corresponds to an average of $2^d$ terms. The division by $p$ in the term $x_i/p$ is here to ensure that, on average, the inner product between $w$ and $\delta_i \odot \frac{x_{i,}}{p}$ is independent of $p$, because:

$$\begin{aligned} \mathop{\mathbb{E}}_{\delta_i \sim B(p)^d} \sum_{j=1}^{d} w_j \left( \delta_i \odot \frac{x_{i,}}{p} \right)_j &= \sum_{j=1}^{d} \mathop{\mathbb{E}}_{\delta_{i,j} \sim B(p)} w_j \delta_{i,j} \frac{x_{i,j}}{p} \\ &= \sum_{j=1}^{d} w_j x_{i,j} \,. \end{aligned}$$

When $p = 1$ and $\lambda > 0$, the only mask with positive probability is the constant mask with all entries equal to 1, which performs no dropout corruption. In that case, DropLasso (4.2) therefore boils down to standard lasso. When $\lambda = 0$ and $p < 1$, on the other hand, DropLasso boils down to the standard dropout regularisation proposed by Srivastava et al. (2014) and studied, among others, by Wager et al. (2013); Baldi and Sadowski (2013); van der Maaten et al. (2013). In general, DropLasso interpolates between lasso and dropout. For $\lambda > 0$, it inherits from lasso regularisation the ability to select features associated with $\ell_1$ regularisation (Bach et al., 2011). We therefore propose DropLasso as a good candidate to select molecular signatures (thanks to the sparsity-inducing $\ell_1$ regularisation) for data corrupted with dropout noise, in particular scRNA-seq data (thanks to the dropout data augmentation).

### 4.2.3 Algorithm

For any convex loss function $L$ such as the square or logistic losses, DropLasso (4.2) is a non-smooth convex optimisation problem whose global minimum can be found by generic solvers for convex programs. Due to the dropout corruption, the total number of terms in the sum in (4.2) is $n \times 2^d$. This is usually prohibitive as soon as $d$ is more than a few, e.g., in practical applications when $d$ is easily of order $10^4$ (number of genes). Hence the objective function (4.2) can simply not be computed exactly for a single candidate model $w$, and even less optimised by methods like gradient descent.

To solve (4.2), we instead propose to follow a stochastic gradient approach to exploit the particular structure of the model, in particular the fact that it is fast and easy to generate a sample randomly corrupted by dropout noise. A similar approach is used for standard dropout regularisation when $L$ is differentiable w.r.t. $w$ (Srivastava et al., 2014), however in our case we additionally need to take care of the non-differentiable $\ell_1$ norm; this can be handled by a forward-backward algorithm which, plugged in the stochastic gradient loop, leads to the proximal stochastic gradient descent algorithm presented in Algorithm 4. The fact that Algorithm 4 is correct, i.e., converges to the solution of (4.2), follows under weak conditions from general results on stochastic approximations and proximal stochastic gradient descent algorithms (Robbins and Siegmund, 1971; Atchadé et al., 2017).

**Algorithm 4** Solving DropLasso

**Require:** Training set $(x_i, y_i)_{i=1,\dots,n}$, initialisation $w_0 \in \mathbb{R}^d$, initial learning rate $\gamma_0 > 0$, learning rate decay $\beta > 0$, number of passes $n_{passes} \in \mathbb{N}$, $\lambda \geq 0$, $p \in [0, 1]$

**procedure** DropLasso
  $w^0 \leftarrow w_0$
  $t \leftarrow 0$
  **for** $iter = 1$ `to` $n_{passes}$ **do**
    $\pi \leftarrow$ random permutation of $[1, n]$ ▷ Shuffle training set
    **for** $i = 1$ `to` $n$ **do** ▷ (Mini-)batch also possible
      $\gamma_t \leftarrow \gamma_0/(1 + \beta t)$
      Sample $\delta \sim Bernoulli(p)^d$
      $z \leftarrow \delta \odot x_{\pi(i)}/p$
      $w^{t+1} \leftarrow S_{\gamma_t \lambda}(w^t - \gamma_t \nabla_w L(w^t, z, y_{\pi(i)}))$ ▷ $S_{\gamma_t\lambda}$is the soft-thresholding operator
      $t \leftarrow t + 1$
    **end for**
  **end for**
**return** $w^t$
**end procedure**

We can easily see that for $p = 1$ , our algorithm becomes a classical stochastic proximal descent algorithm. On the other hand when $\lambda = 0$, the soft threshholding operator becomes the identity and we turn back to the stochastic gradient descent with the dropout trick.

When $p = 1$ (no dropout), it is known that the solution of (eq:droplasso) is sparse, and is even 0 when $\lambda$ is larger than a value $\lambda_{max}$ than can be used as initial value when one wants to compute the set of solutions over a decreasing grid of values for $\lambda$. Interestingly, this property also holds when $p < 1$, with the same $\lambda_{max}$ value which therefore does not depend on $p$:

**Theorem 1.** *For a loss function of the form $L(w, x, y) = \ell_y(w^\top x)$ where $\ell_y$ is convex and differentiable at 0 for all $y$, $w = 0$ is solution of (4.2) if and only if $\lambda \geq \lambda_{\max}$ with*

$$\lambda_{\max} = \left\| \frac{1}{n} \sum_{i=1}^{n} \ell'_{y_i}(0) x_i \right\|_\infty . \tag{4.3}$$

*Proof.* Under the assumptions of the theorem, the function $w \to F(w)$ with

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta_i \sim B(p)^d} L(w, \delta_i \odot \frac{x_{i,}}{p}, y_i) = \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta_i \sim B(p)^d} \ell_{y_i}\left(w^\top(\delta_i \odot \frac{x_{i,}}{p})\right)$$

is convex and its subdifferential is

$$\partial F(w) = \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta_i \sim B(p)^d} \partial\ell_{y_i}\left(w^\top(\delta_i \odot \frac{x_{i,}}{p})\right) \delta_i \odot \frac{x_{i,}}{p}\,. \tag{4.4}$$

At $w = 0$, this simplifies to

$$\partial F(0) = \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta_i \sim B(p)^d} \ell'_{y_i}(0)\, \delta_i \odot \frac{x_{i,}}{p} = \frac{1}{n}\sum_{i=1}^{n} \ell'_{y_i}(0)\, x_i\,.$$

Besides, the subdifferential of $w \to \|w\|_1$ at $w = 0$ is $\partial\|\cdot\|_1(0) = \{u : \|u\|_\infty \le 1\}$. Using the standard characterization that $w$ is solution of the convex problem (4.2) if and only if $0 \in \partial\left(F + \lambda\|\cdot\|_1\right)(w)$, we get that $w = 0$ is a solution of (4.2) if and only if $-\partial F(0) \in \lambda\partial\|\cdot\|_1(0)$, or equivalently $\|\partial F(0)\|_\infty \le \lambda$. The theorem follows by using (4.4). □

In practice, for the square loss $\ell_y(u) = (u-y)^2$, we get $\ell'_y(0) = -2y$; and for the logistic loss $\ell_y(u) = \ln(1 + e^{-yu})$, we get $\ell'_y(0) = -y/2$. Taking

$$S = \frac{1}{n}\sum_{i=1}^{n} y_i x_i\,,$$

we therefore have the following $\lambda_{\max}$ values for respectively the square and logistic losses:

$$\lambda_{\max}^{\text{square}} = 2\|S\|_\infty\,, \quad \lambda_{\max}^{\text{logistic}} = \frac{\|S\|_\infty}{2}\,.$$

In order to get the regularization path of DropLasso, i.e., the set solutions (4.2) when $\lambda$ varies for a fixed $p$, we therefore first fix a grid of values to test for $\lambda$ in an interval $[\lambda_{\min}, \lambda_{\max}]$ where $\lambda_{\max}$ is given by (4.3) and, for example $\lambda_{\min} = \lambda_{\max}/100$. We then iteratively solve (4.2) using Algorithm 4 for decreasing values of $\lambda$ using warm restart, i.e., taking the solution for the previous $\lambda$ as initialization for the next $\lambda$. Since 0 is the solution for $\lambda = \lambda_{\max}$, we initialize the first optimization with $w_0 = 0$.

### 4.2.4 DropLasso and elastic net

As we already mentioned, DropLasso interpolates between lasso ($p = 1, \lambda > 0$) and dropout ($p \in [0,1]$, $\lambda = 0$). On the other hand, dropout regularisation is known to be related to ridge regularisation (Wager et al., 2013; Baldi and Sadowski, 2013); in particular, for the square loss, dropout regularisation boils down to ridge regression after proper normalisation of the data, while for more general losses it can be approximated by reweighted version of ridge regression. Here we show that DropLasso largely inherits these properties, and in a sense is to elastic net what dropout is to ridge.

Let us start with the square loss. In that case we have the following:

**Theorem 2.** *If the data are scaled so that*

$$\forall j \in [1,d]\,, \quad \frac{1}{n}\sum_{i=1}^{n} x_{i,j}^2 = 1\,,$$

*then solving the DropLasso problem (4.2) with parameters $\lambda$ and $p$ and the square loss $L_{square}$ is equivalent to solving the elastic net problem*

$$\min_{w\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} L_{square}(w, x_i, y_i) + \lambda_{enet}\left(\alpha_{enet}\|w\|_2^2 + (1-\alpha_{enet})\|w\|_1\right)\,,$$

*with*

$$\lambda_{enet} = \lambda + \frac{1-p}{p} \quad \textit{and} \quad \alpha_{enet} = \frac{1-p}{1-p+\lambda p}\,.$$

*Proof.* By developing the error function and marginalising over the Bernoulli variables, we can rewrite the objective function of (4.2) as follows:

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta_i \sim B(p)^d} L_{\text{square}}(w, \delta_i \odot \frac{x_{i,}}{p}, y_i) + \lambda \left\|w\right\|_1 \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta_i \sim B(p)^d} \left(y_i - \sum_{j=1}^{d} w_j \delta_{i,j} \frac{x_{i,j}}{p}\right)^2 + \lambda \left\|w\right\|_1 \\
&= \frac{1}{n}\sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{d} w_j x_{i,j}\right)^2 + \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{d} w_j^2 x_{i,j}^2 \text{Var}\left(\frac{\delta_{i,j}}{p}\right) + \lambda \left\|w\right\|_1 \\
&= \frac{1}{n}\sum_{i=1}^{n} L_{\text{square}}(w, x_i, y_i) + \frac{1-p}{p}\sum_{j=1}^{d}\left(\frac{1}{n}\sum_{i=1}^{n} x_{i,j}^2\right) w_j^2 + \lambda \left\|w\right\|_1 \\
&= \frac{1}{n}\sum_{i=1}^{n} L_{\text{square}}(w, x_i, y_i) + \frac{1-p}{p}\|w\|_2^2 + \lambda \left\|w\right\|_1\,,
\end{aligned}$$

and Theorem 2 easily follows by identifying $\lambda_{\text{enet}}$ and $\alpha_{\text{enet}}$ from this equation. □

We note that conversely, in order to solve an elastic net problem with parameters $\lambda_{\text{enet}}$ and $\alpha_{\text{enet}}$, one can equivalently solve a DropLasso problem with parameters

$$\lambda = \lambda_{\text{enet}}\,(1 - \alpha_{\text{enet}}) \quad \text{and} \quad p = \frac{1}{1 + \lambda_{\text{enet}}\alpha_{\text{enet}}}\,.$$

When the data are not scaled as in Theorem 2, then instead of a standard elastic net penalty the DropLasso problem with square loss is equivalent to a modified elastic net problem where the $\ell_2$ norm is weighted by the vector of mean squared norm of each column in the data matrix.

In the case of the logistic loss, we can also adapt a result of Wager et al. (2013) which relates dropout to an adaptive version of ridge regression:

**Property 5.** *: For the logistic loss, DropLasso can be approximated when the dropout probability $p$ is close to 1 by an adaptive version of elastic net that automatically scales the data but also that encourages more confident predictions.*

*Proof.* Writing the Taylor expansion for the logistic loss up to the second order when the dropout is small ($p$ close to 1), we obtain the following quadratic approximation to the dropout loss on a point:

$$\begin{aligned}
L(w, \delta_{\mathbf{i}} \odot \frac{x_{i,}}{p}, y_i) \simeq\; & L(w, x_{i,}, y_i) \\
& + \sum_{j=1}^{d} \frac{\partial L(w, x_{i,}, y)}{\partial x_{i,j}} \left(\frac{\delta_{i,j}}{p} - 1\right) x_{i,j} \\
& + \frac{1}{2}\sum_{j=1}^{d}\sum_{k=1}^{d} \frac{\partial^2 L(w, x_{i,}, y)}{\partial x_{i,j}\partial x_{i,k}} \left(\frac{\delta_{i,j}}{p} - 1\right)\left(\frac{\delta_{i,k}}{p} - 1\right) x_{i,j} x_{i,k}\,.
\end{aligned}$$

Taking the expectation with respect to $\delta_i \sim B(p)^d$, the first order term cancels out since $\mathbb{E}\delta_{i,j} = p$ for all $j \in [1, d]$. The off-diagonal second-order term also disappear because $\delta_{i,j}$ and $\delta_{i,k}$ are independent for $j \neq k$. Noting that for $\delta \sim B(p)$ it holds that

$$\mathbb{E}\left(\frac{\delta}{p} - 1\right)^2 = \frac{1-p}{p}\,,$$

and that for the logistic loss,

$$\frac{\partial^2 L_{\text{logistic}}(w, x_{i,}, y)}{\partial x_{i,j}^2} = \pi_i(1-\pi_i)w_j^2 \,,$$

where $\pi = e^{w^\top x_i} / \left(1 + e^{w^\top x_i}\right) = P_w(Y = 1 \,|\, X = x_i)$ under the logistic model parametrized by $w$, we finally get the following quadratic approximation:

$$L_{\text{logistic}}(w, \delta_{\mathbf{i}} \odot \frac{x_{i,}}{p}, y_i) \simeq L_{\text{logistic}}(w, x_{i,}, y_i) + \frac{1-p}{2p} \sum_{j=1}^{d} \pi_i(1-\pi_i) w_j^2 x_{i,j}^2 \,.$$

We finally get the following approximation to the DropLasso objective function:

$$\begin{aligned} &\frac{1}{n} \sum_{i=1}^{n} \mathop{\mathbb{E}}_{\delta_i \sim B(p)^d} L_{\text{logistic}}(w, \delta_i \odot \frac{x_{i,}}{p}, y_i) + \lambda \left\|w\right\|_1 \\ &\simeq \frac{1}{n} \sum_{i=1}^{n} L_{\text{logistic}}(w, x_{i,}, y_i) + \frac{1-p}{2p} \text{ïٔ} \cdot \sum_{j=1}^{d} \gamma_j w_j^2 + \lambda \left\|w\right\|_1 \,, \end{aligned}$$

where for $j \in [1, d]$,

$$\gamma_j = \sum_{i=1}^{n} \frac{\partial^2 L(w, x_{i,}, y)}{\partial^2 w^\top x_{i,j}} . x_{i,j} = \sum_{i=1}^{n} \pi_i(1-\pi_i) x_{i,j}^2 \,.$$

□

This shows that with the logistic loss, that the ridge penalty corresponding to the approximation of the Droplasso is controlled both by the size of the features $x_{i,j}^2$, but also by the fact that the prediction for each sample is confident or not. In fact $\gamma_j$ is maximal when $\pi_i = 0.5$ for all $i \in [1, n]$, which means that the model is not confident about the examples it is learnt with.

## 4.3 Results

### 4.3.1 Simulation results

We first investigate the performance of DropLasso on simulated data, and compare it to standard dropout and elastic net regularisation. We design a toy simulation to illustrate in particular how corruption by dropout noise impacts the performances of the different methods. The simulation goes as follow :

- We set the dimension to $d = 20$.
- Each sample is a random vector $z \in \mathbb{N}^d$ with entries following a Poisson distribution with parameter $\pi = 1$. The data variables are independent.
- The "true" model is a logistic model with sparse weight vector $w \in \mathbb{R}^d$ satisfying $w_i = +10, i = 1 \dots d_1$, $w_i = -10, i = (d_1 + 1) \dots 2d_1$, and $w_i = 0$ for $i = (2d_1 + 1), \dots, d$. $d_1$ here is fixed to 2 and thus we have 4 active predictors (with signal) in this simulation.
- Using $w$ as the true underlying model and $z$ as the true observations, we simulate a label $y \sim \text{Bernoulli}(1/(1 + \exp(-\sum_{j=1}^{d} w_j z_j)))$
- We introduce corruption by dropout events by multiplying entry-wise $z$ with an i.i.d Bernoulli variables $\delta$ with probability $q$.

We simulate $n = 100$ samples to train elastic net and DropLasso models, and evaluate their performance in terms of area under the receiving operator curve (AUC) on $10,000$ independent samples. Both models have two parameters, $\lambda$ and $\alpha$ for elastic net, $\lambda$ and $p$ for DropLasso. We vary each parameter over a grid: $\alpha$ over 11 regularly spaced values between 0 and 1, $p$ over the grid $0.6^n$ for $n = 0, \ldots, 10$, and $\lambda$ over a regular grid of 10 values between $\lambda_{\max}$ and $\lambda_{\max}/100$, where $\lambda_{\max}$ is the smallest value such that the solution of the optimization problem is the null model (see Theorem 1). All model are trained on the training set, and the best parameter set is chosen as the one that maximizes the AUC on an independent validation set of 10,000 samples; only the AUC of the best model is then reported on the test set. We repeat the whole procedure 1,000 times in order to estimate the variability of the performance of each method.

Table 4.1: Test AUC of elastic net and DropLasso regression on simulations with different amount of dropout noise on the training data. The $*$ indicates that a method significantly outperforms the other (i.e., $P < 0.05$ according to a paired $t$-test comparing the AUC over 1,000 repeats).

| **Noise rate** | **Elastic net** | **DropLasso** |
|---|---|---|
| q=1 | $0.974 \pm 0.006^*$ | $0.954 \pm 0.012$ |
| q=0.4 | $0.641 \pm 0.043$ | $0.639 \pm 0.027$ |
| q=0.2 | $0.554 \pm 0.031$ | $0.561 \pm 0.021^*$ |

Table 4.1 shows the classification performance in terms of test AUC of elastic net and DropLasso, when we vary the amount of dropout noise in the training data. We first observe that, for both methods, the performance drastically decreases when dropout noise increases, confirming the difficulty induced by dropout events to learn predictive models. Second, we note that in the absence of noise, elastic net significantly outperforms DropLasso. However, when the amount of noise increases, both methods perform similarly (for $q = 0.4$), and ultimately DropLasso outperforms elastic net in the configuration with large dropout noise ($q = 0.2$). This confirms that DropLasso provides potential benefits in situations where data are corrupted by dropout noise.

### 4.3.2 Classification on Single Cell RNA-seq

We now turn on to real scRNA-seq data. To evaluate the performance of methods for supervised classification, we collected 4 publicly available scRNA-seq datasets amenable to this setting, as summarised in Table 4.2. These datasets were pre-processed by Soneson and Robinson (2017), and we downloaded them from the *conquer* website[2], a collection of consistently processed, analysis-ready and well documented publicly available scRNA-seq data sets. We used the preprocessed length-scaled transcripts per million mapped reads (see Soneson and Robinson, 2017, for details about data processing). These datasets were used by Soneson and Robinson (2017) to assess the performance of methods for gene differential analysis between classes of cells, and we follow the same splits of cells into classes for our experiments of supervised classification.We used the available sample annotations to create binary classification problems, as described in Table 4.2. Note that some datasets have more than two classes, in which case we created several binary classification problems.

On each of the 10 resulting binary classification problems, we compare the performance of 5 regularisation methods for logistic regression: lasso, ridge, elastic net, dropout and DropLasso. We train the different models on 20% of the data chosen in such way that labels are balanced, choose the best hyper-parameter(s) for each on a 20% validation set, and finally evaluate the performance of the resulting models on the 60% remaining data. We search the

[2] http://imlspenticton.uzh.ch:3838/conquer/

best parameters for each method over the same grid as described for the simulation study above (except that lasso, ridge and dropout have a single parameter to tune).

Table 4.2: Description of the scRNA-seq data and the corresponding (binary) classification tasks.

| **Dataset** | **Classification task** | **Variables** | **Samples** |
|---|---|---|---|
| EMTAB2805 | Cell cycle phase: G1 vs G2M | 18,979 | 96 ; 96 |
| EMTAB2805 | Cell cycle phase: S vs G1 | 18,740 | 96 ; 96 |
| EMTAB2805 | Cell cycle phase: S vs G2 | 18,873 | 96 ; 96 |
| GSE45719 | mid blastocyst vs 16-cell stage blastomere | 22,059 | 50 ; 60 |
| GSE45719 | 8-cell stage blastomere vs 16-cell stage blastomere | 21,590 | 50 ; 60 |
| GSE48968 | BMDC 1h LPS vs 4h LPS Stimulation | 16,439 | 95 ; 96 |
| GSE48968 | BMDC 4h LPS vs 6h LPS Stimulation | 15,719 | 95 ; 96 |
| GSE74596 | NKT0 vs NKT17 | 15,642 | 45 ; 44 |
| GSE74596 | NKT0 vs NKT1 | 14,962 | 45 ; 46 |
| GSE74596 | NKT1 vs NKT2 | 16,135 | 46 ; 48 |

Table 4.3: Mean test AUC score for different regularizations schemes, on different binary classification problems.)

| **Dataset** | **dropout** | **DropLasso** | **elastic net** | **ridge** | **lasso** |
|---|---|---|---|---|---|
| EMTAB2805, G1 vs G2M | 0.96 | 0.97 | 0.98 | 0.97 | 0.94 |
| EMTAB2805, G1 vs S | 0.98 | 0.97 | 0.98 | 0.98 | 0.91 |
| EMTAB2805, S vs G2M | 0.99 | 0.98 | 0.99 | 0.99 | 0.95 |
| GSE45719, 16-cell vs Mid blast. | 1.00 | 0.99 | 1.00 | 1.00 | 0.99 |
| GSE45719, 16-cell vs 8-cell | 0.98 | 0.95 | 0.97 | 0.98 | 0.72 |
| GSE48968, 1h vs 4h | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| GSE48968, 4h vs 6h | 0.84 | 0.84 | 0.86 | 0.85 | 0.79 |
| GSE74596, NKT0 vs NKT17 | 1.00 | 1.00 | 0.99 | 0.99 | 1.00 |
| GSE74596, NKT0 vs NKT1 | 1.00 | 1.00 | 1.00 | 1.00 | 0.99 |
| GSE74596, NKT1 vs NKT2 | 0.98 | 0.98 | 0.99 | 0.99 | 0.98 |

The first observation is that the performances reached by all methods on all datasets are generally high, and can reach a perfect AUC score of 1 on some of the datasets. This suggests that the labels chosen in these datasets are sufficiently different in terms of transcriptomic profiles that they can be easily recognised most of the time. We still notice some differences in performance between datasets, with GSE48968 with 1h-4h stimulation labels being the easiest dataset to classify while GSE48968 with 6h-4h stimulation labels is the most challenging, for all methods. This contrast of performance for the same dataset confirms that supervised learning on single cell data can be challenging when the labels are biologically close regardless of the preprocessing step. Soneson and Robinson (2017) also noticed a difference in signal-to-noise ratio between these datasets, in the context of gene differential analysis. Second, we observe that the lasso is clearly the worst performing method in terms of accuracy, while all other methods tend to have similar accuracies. To further analyze the relative performance of different methods, we perform statistical tests for each pair of methods on each dataset, and call a method a "winner" if it is statistically more accurate than the other method ($P < 0.05$ for a $t$-test on the test AUC). Figure 4.3 reports, for each method, the number of times it is a

winner. The plot first confirms that lasso is the least performing method in terms of accuracy, and that elastic net and dropout are the methods that have the largest number of wins. Although it was expected that elastic net improves over the lasso in this high-dimensional data setting, where many genes are correlated through several regulatory networks (Abdelmoez et al., 2018), it is also interesting to see that elastic net slightly outperforms ridge indicating that at least some of these biological labels can be explained by a sparse model. Dropout outperforming ridge indicates that the adaptive regularisation that dropout introduces is relevant to this type of data. Finally, DropLasso only outperforms the lasso method, but in contrast with dropout (and ridge) does allow for feature selection and the discovery of potential biomarkers, which we study next.

Figure 4.3: P-value wins by method: Number of significant wins (best accuracy among all methods with a P-value $< 0.005$) for each method over all datasets.

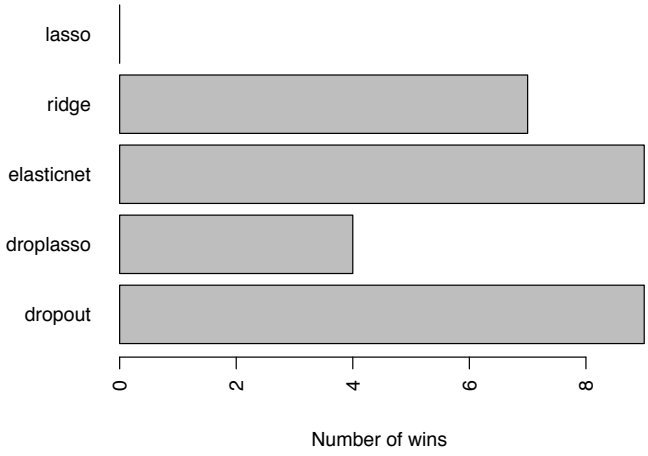



Table 4.4 shows the average number of selected features for each method on each classification problem. Selected features are defined by having nonzero coefficients in the corresponding model after fixing its parameters. We use a sensitivity threshold $\epsilon = 10^{-8}$ to account for potential convergence issues (coefficients below this threshold are considered as null). According to Table 4.4, lasso is the method with the highest values of sparsity (that is selecting the most compact sets of features for the classification task) with an average selected set size of 6.63, coming before DropLasso with an average of 676. It is interesting that elastic net does perform feature selection but with a much bigger average selected size of $11,869$. Ridge and dropout do not perform feature selection if we do not account for coefficients below the threshold.

Providing a compact set of features that can discriminate the task labels with high accuracy is important not only for computational time and memory footprint but more importantly for the interpretability of the model and the identification of a minimal set of features or a *molecular signature* of the observed phenotype. Using the reported results in the previous tables, we compare in Figure 4.4 the trade off presented by the different methods between accuracy, as evaluated by mean AUC for each dataset, and model sparsity that can be defined by the proportion of features not selected for each dataset, where each point is the best validated model for one method on one dataset. Figure 4.4 confirms the fact that most accurate models are not sparse, and presents DropLasso as the method that trades off best sparsity and accuracy, presenting a more sparse alternative to elastic net, and a more accurate alternative to lasso.

**Biological significance of the selected features:**

To conclude this section, we now evaluate the biological relevance of the gene lists or the molecular signatures estimated by the two methods that consistently provided sparse models,

Table 4.4: Average number of selected variables for the different models

| Dataset | Variables | dropout | DropLasso | elastic net | ridge | lasso |
|---|---|---|---|---|---|---|
| EMTAB2805, G1 vs G2M | 18,979 | 18,973 | 274 | 13,117 | 14,597 | 7 |
| EMTAB2805, G1 vs S | 18,740 | 18,733 | 291 | 13,089 | 14,606 | 7 |
| EMTAB2805, S vs G2M | 18,979 | 18,867 | 41 | 8,193 | 13,088 | 6 |
| GSE45719, 16-cell vs Mid blast. | 22,059 | 21,965 | 4 | 19,747 | 19,747 | 3 |
| GSE45719, 16-cell vs 8-cell | 21,590 | 21,413 | 4,892 | 17,133 | 21,393 | 7 |
| GSE48968, 1h vs 4h | 16,439 | 16,431 | 18 | 7,071 | 10,139 | 7 |
| GSE48968, 4h vs 6h | 15,719 | 15,711 | 594 | 8,994 | 12,998 | 14 |
| GSE74596, NKT0 vs NKT17 | 15,642 | 15,416 | 60 | 7,000 | 8,758 | 5 |
| GSE74596, NKT0 vs NKT1 | 14,962 | 14,806 | 33 | 6,364 | 6,364 | 5 |
| GSE74596, NKT1 vs NKT2 | 16,135 | 16,020 | 55 | 7,368 | 9,148 | 5 |

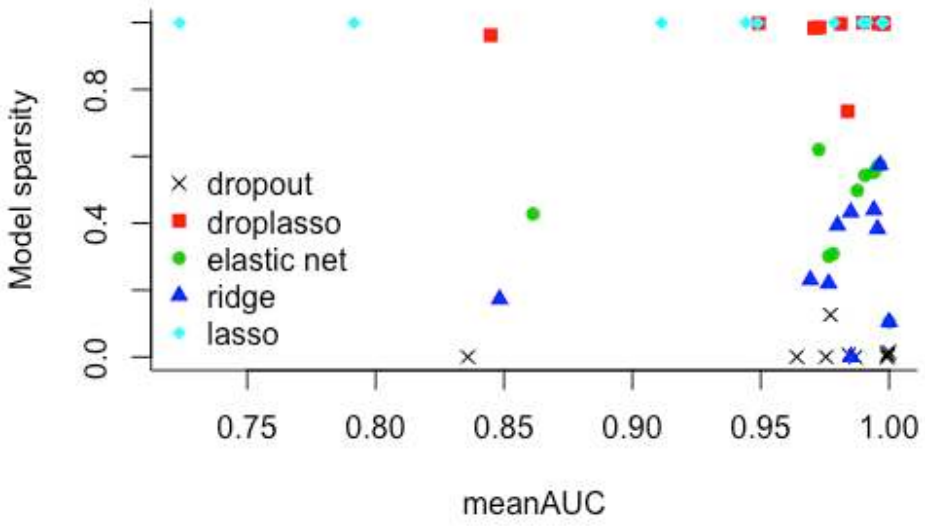
that is the lasso and DropLasso regularisation. We first illustrate this comparison on the first dataset, EMTAB2805, where the goal is to discriminate mice cells at the G1 from the G2M cell cycle stages. To this end, we retrain the different methods with the parameters corresponding to the best accuracy but this time on all the samples, and then we perform a Gene Ontology enrichment analysis using DAVID (Huang et al., 2009) on the subset of genes with non-zero coefficients for each method.

For this dataset and the best tuning parameters, DAVID identifies 24 genes selected by DropLasso and 5 genes selected by lasso. While the analysis of the genes selected by DropLasso shows enrichment in the functional term "positive regulation of mitotic cell cycle", the genes selected by the lasso method do not include the terms "cell division", "cell cycle" or "mitosis". Among the genes selected by DropLasso, 5 genes were related to the functional term "cell cycle" and 2 genes were related to the term "cell division". It is interesting to notice first that 4 out of 5 genes selected by lasso were related to ATP synthesis which underlies the potential importance of the relationship between energy and the cell cycle, as reviewed in (Salazar-Roa and Malumbres, 2017), and second that all the genes selected by lasso were also selected by DropLasso, which shows that DropLasso potentially allows for the discovery of novel biomarkers.

The enrichment analysis on the GSE48968 dataset, where the goal is to discriminate between primary mice dendritic cells exposed to 1 hour LPS stimulation and 4 hours stimulation, identifies 8 genes selected by DropLasso and 4 genes selected by lasso. Although both sets were enriched with the term "response to virus", DropLasso set shows enrichment for "immune response", "inflammatory response" and "cellular response to lipopolysaccharide", as it also interestingly shows enrichment for the terms "defense response to Gram-negative bacterium" and " cellular response to tumor necrosis factor" , as it is known that lipopolysaccharide stimulates the production of tumor necrosis factor (TNF)-$\alpha$ (Barsig et al., 1995; Ogikubo et al., 2004). While the analysis of lasso selected genes does not reveal any enriched functional annotation cluster, one cluster is enriched in the DropLasso genes set and appears to be mainly related to cytokines and chemokine which were previously shown to have very altered profiles by LPS stimulation (Medvedev et al., 2000; Kopydlowski et al., 1999; Johnston et al., 1998). Interestingly, here again all the genes selected by lasso are also selected by DropLasso.

Finally, the enrichment analysis on the GSE74596 with the classification task between natural killer T cell subsets (NKT0 vs NKT1) shows some differences in the selected genes by DropLasso and lasso, where some genes selected by lasso are not selected by DropLasso (3 out of 6 identified genes by lasso). While both methods are mostly enriched with the same terms: "CTL mediated immune response against target cells" and "Ras-Independent

Figure 4.4: Accuracy-sparsity tradeoff for the different methods:Average AUC against average model sparsity of models with best validation parameters across the different methods and datasets. Each point represents the result of one method (see legend) on one dataset.




pathway in NK cell-mediated cytotoxicity", DropLasso set additionally shows enrichment for two terms including the term "Immunoglobulin" and three terms including the term "major histocompatibility complex (MHC) " molecules, that are both related by definition to T-cells.

Overall, this short analysis of the molecular signatures estimated by lasso and DropLasso confirms that a small number of relevant genes tend to be selected by both methods, and the fact that DropLasso significantly outperforms lasso in AUC on most datasets confirms that its list of genes is likely to be more complete than that selected by lasso.

## 4.4 Discussion

ScRNA-seq is changing the way we study cellular heterogeneity and investigate a number of biological process such as differentiation or tumourigenesis. Yet, as the throughput of scRNA-seq technologies increases and allows to process more and more cells simultaneously, it is likely that the amount of information captured in each individual cell will remain limited in the future and that dropout noise will continue to affect scRNA-seq (and other single-cell technologies).

Several techniques have been proposed to handle dropout noise in the context of data normalisation or gene differential expression analysis, and shown to outperform standard techniques widely used for bulk RNA-seq data analysis. In this paper we investigate a new setting which, we believe, will play an important role in the future: supervised classification of cell populations into pre-specified classes, and selection of molecular signatures for that purpose. Molecular signatures for the classification of tissues from bulk RNA-seq data has already had a tremendous impact in cancer research, and as more and more cell types are investigated and discovered with scRNA-seq it is likely that specific molecular signatures will be useful in the future to automatically sort cells into their classes.

DropLasso, the new technique we propose, borrows the recent idea of dropout regularisation from machine learning, and extends it to allow feature selection. While a parallel between dropout regularisation and (data-dependent) ridge regression has already been shown by Wager et al. (2013) and Baldi and Sadowski (2013), it is reassuring that we are able to extend this parallel to DropLasso and elastic net regularisation.

More interesting is the fact that, on both simulated and real data, we obtained promising

results with DropLasso in terms of trade-off between accuracy and feature selection. They suggest that, again, specific models tailored to the data and noise can give an edge over generic models developed under different assumptions.

The intuition behind why dropout (and DropLasso) perform well on scRNA-seq data, however, remains a bit unclear. Our main motivation to use them in this context was to see them as data augmentation techniques, where training data are corrupted according to the noise we assume in the data. While we believe this is fundamentally the reason why we obtained promising results, alternative explanations for the success of dropout have been proposed, and may also play a role in the context of scRNA-seq. They include for example the interpretation of dropout as a regulariser similar to a data-dependent weighted version of ridge regularisation, which works well in the presence of rare but important features (Wager et al., 2013); it would be interesting to clarify if the regularisation induced by DropLasso on scRNA-seq data exploits some fundamental property of these data, and may be replaced by a more direct approach to model this.

Finally, this first study of dropout and DropLasso regularisation on biological data paves the way to many future direction. For example, it is known that the probability of dropout in scRNA-seq data depends on the gene expression level (Kharchenko et al., 2014; Risso et al., 2018). It would therefore be interesting to study both theoretically and empirically if a dropout regularisation following a similar pattern may be useful. Second, instead of independently perturbing the different features one may create a correlation between the dropout events in different genes. Creating a correlation may be a way to create new regularisation by generating a *structured* dropout noise. It may for example be possible to derive a correlation structure for dropout noise from prior knowledge about gene annotations or gene networks in order to enforce a structure in the molecular signature, just like structured ridge and lasso penalties have been used to promote structure molecular signatures with bulk transcriptomes (Rapaport et al., 2007; Jacob et al., 2009).

# Chapter 5

# ASNI: Adaptive Structured Noise Injection for shallow and deep neural networks

> " *Structure is not just a means to a solution.It is also a principle and a passion*"
>
> Marcel Breuer

Contents

**Abstract**

Dropout is a regularisation technique in neural network training where unit activations are randomly set to zero with a given probability *independently*. In this work, we propose a generalisation of dropout and other multiplicative noise injection schemes for shallow and deep neural networks, where the random noise applied to different units is not independent but follows a joint distribution that is either fixed or estimated during training. We provide theoretical insights on why such an adaptive structured noise injection scheme may be relevant, and empirically confirm that it helps boost the accuracy of simple feedforward and convolutional neural networks, disentangles the hidden layer representations, and leads to sparser representations. Our proposed method is a straightforward modification of the classical dropout and does not require additional computational overhead.
**Availability:** All code concerning the real data experiments is available at https://github.com/BeyremKh/ASNI

**Abstract**

*Dropout* est une technique de régularisation dans l'entraînement des réseau de neurones artificiels où les activations d'unités sont fixées aléatoirement à zéro *indépendamment* avec une probabilité donnée. Dans ce chapitre, nous proposons une généralisation du "dropout" classique et d'autres schémas d'injection de bruit multiplicatif pour les réseaux de neurones, où le bruit appliqué à différentes unités n'est pas indépendant mais suit une distribution jointe qui est soit fixe, soit estimée au cours de l'entrainement. Nous fournissons des éclaircissements théoriques sur les raisons pour lesquelles une telle injection de bruit structurée adaptative peut être pertinente, et nous confirmons empiriquement qu'elle contribue à améliorer les performances en généralisation de réseaux de neurones denses et convolutionnels, à dissocier les représentations des couches intermédiaires et à améliorer leur parcimonie. La méthode que nous proposons est une simple modification du "dropout" classique et le temps de calcul supplémentaire en est négligeable.
**Availability:** All code concerning the real data experiments is available at https://github.com/BeyremKh/ASNI

## 5.1 Introduction

The tremendous empirical success of deep neural networks for many machine learning tasks such as image classification and object recognition (Krizhevsky et al., 2017) contrasts with their relatively poor theoretical understanding. One feature commonly attributed to DNN to explain their performance is their ability to build hierarchical *representations* of the data, able to capture relevant information in the data at different scales (Bengio et al., 2013; Tishby and Zaslavsky, 2015; Mallat, 2012). An important idea to create good sets of representations is to reduce redundancy and increase diversity in the representation, an idea that can be traced back to early investigations about learning (Barlow, 1959) and that has been implemented in a variety of methods such as independent component analysis (Hyvärinen, 2013) or feature selection (Peng et al., 2005). Explicitly encouraging diversity has been shown to improve the performance of ensemble learning models (Kuncheva and Whitaker, 2003; Dietterich, 2000), and techniques have been proposed to limit redundancy in DNN by pruning units or connections (Hassibi and Stork, 1993; LeCun et al., 1990; Mariet and Sra, 2016) or by explicitly encouraging diversity between units of each layer during training (Cogswell et al., 2015; Desjardins et al., 2015; Rodríguez et al., 2016; Luo, 2017).

Dropout (Hinton et al., 2012b; Srivastava et al., 2014) is a recent and popular regularisation techniques in deep learning that exploits the idea of creating diversity stochastically while training, by randomly setting some units or connections to zero during stochastic gradient optimisation. Proposed by Hinton et al. (2012b) as a way to prevent co-adaptation of units and to approximately combine exponentially many different DNN architectures efficiently, it has improved DNN performance in many benchmark datasets. Dropout can also be interpreted as a regularisation technique (Baldi and Sadowski, 2013; Wager et al., 2013; Maeda, 2014; Helmbold and Long, 2017), however its impact on learning a good representation of the data remains elusive. Several variants of the original dropout model have been proposed to adapt the algorithm to other models (e.g., Gal and Ghahramani, 2016b), or to modify the distribution of the stochastic noise. Ba and Frey (2013) propose that the dropout rate of a unit should depend on its magnitude and dynamically adapt the dropout rate of each unit activation during training. Another variant is to group units together because of their proximity in a map (Tompson et al., 2014; DeVries and Taylor, 2017) of because they are strongly correlated (Aydore et al., 2019), before applying dropout jointly on the units in a group. This can equivalently be interpreted as applying dropout to individual units, but constraining the stochastic dropout noise in units within a group to be perfectly correlated.

In this work, we extend and generalise the idea to modify the noise distribution, and analyse both theoretically and empirically the effect of different choices. In particular, we study the impact of creating correlations among noise in the units of a given layer, generalising the ideas of Tompson et al. (2014); DeVries and Taylor (2017); Aydore et al. (2019) to a general covariance structure. We depart from binary dropout noise to the more flexible multiplicative Gaussian noise model, which allows to specify any covariance structure without the need to explicitly cluster units into groups, and highlight the role of the noise correlation matrix in the regularisation effect of this structured noise injection procedure. We show in particular that borrowing the covariance of the units to create the covariance of the noise can decrease redundancy among the units, a phenomenon we confirm empirically that leads to better representations and classification accuracy.

## 5.2 Dropout and multiplicative noise

Let us first set notations to describe a standard neural network for inputs in $\mathbb{R}^d$ with $H$ layers of respective dimensions $d_1, \dots, d_H$. For any layer $l \in [1, H]$ let $W^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ and $b^{(l)} \in \mathbb{R}^{d^{(l)}}$ denote respectively the matrix of weights and the vector of biases at layer $l$, with the convention $d^{(0)} = d$, and let $\theta = (W_l, b_l)_{l=1,\dots,H}$ denote the set of parameters

of the network. The network defines a function $f_\theta : \mathbb{R}^d \to \mathbb{R}^{d^{(H)}}$ given for any $x \in \mathbb{R}^d$ by $f_\theta(x) = y^{(H)}$, where $y^{(l)}$ is defined recursively for $l = 0, \ldots, H$ by $y^{(0)} = x$ and, for $l \in [1, H]$:

$$\begin{cases} z^{(l)} &= W^{(l)} y^{(l-1)} + b^{(l)} \,, \\ y^{(l)} &= \sigma^{(l)}(z^{(l)}) \,, \end{cases}$$

where $\sigma^{(l)}$ is an activation function at the $l$-th layer, such as the RELU function $\sigma(t) = t\mathbf{1}(t > 0)$ for $t \in \mathbb{R}$, applied entrywise if its input is a vector.

Given a training set $\mathcal{D}$ of $N$ labelled inputs $(x_1, y_1), \ldots, (x_N, y_N) \in \mathbb{R}^d \times \mathbb{R}^p$, and a loss function $L : \mathbb{R}^{d^{(H)}} \times \mathbb{R}^p \to \mathbb{R}$, training the neural network amounts to fitting the parameters $\theta$ by trying to minimise the average loss over the training set:

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} L\left(f_\theta\left(x_i\right), y_i\right) \,, \tag{5.1}$$

usually by some form of stochastic gradient descent (SGD) using backpropagation to compute gradients. For example, when the label is a scalar ($p = 1$), then one can use $d^{(H)} = 1$ and the squared error $L(u, v) = (u - v)^2$ for $u, v \in \mathbb{R}$.

A popular way to improve the training of neural networks is to use dropout regularisation, where the units of the input and hidden layers are stochastically omitted during training (Srivastava et al., 2014). Dropout is a particular case of multiplicative noise, which we can formalise as follows. Given a sequence of vectors $r = \left(r^{(0)}, \ldots, r^{(H-1)}\right) \in \mathbb{R}^{d^{(0)}} \times \ldots \times \mathbb{R}^{d^{(H-1)}}$, of total dimensions $D = d^{(0)} + \ldots + d^{(H-1)}$, we create the modified function $f_\theta(x, r) = y^{(H)}$ where $y^{(0)} = x$ and, for $l \in [1, H]$:

$$\begin{cases} \widetilde{y}^{(l-1)} &= r^{(l-1)} \odot y^{(l-1)} \,, \\ z^{(l)} &= W^{(l)} \widetilde{y}^{(l-1)} + b^{(l)} \,, \\ y^{(l)} &= \sigma^{(l)}(z^{(l)}) \,, \end{cases}$$

where $\odot$ denotes the Hadamard or element-wise product. We then define a set of independent and identically distributed (i.i.d.) random variables $(R_i)_{i=1,\ldots,N}$ with values in $\mathbb{R}^D$ (the "noise") and train the network by solving

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \mathbb{E} L\left(f_\theta\left(x_i, R_i\right), y_i\right) \,. \tag{5.2}$$

With these notations, the standard dropout approach of (Srivastava et al., 2014) with parameter $p \in [0, 1]$ is obtained by taking a noise distribution with i.i.d. entries across the dimensions taking the value $1/p$ with probability $p$, and 0 with probability $1 - p$.

## 5.3 Structured noise injection (SNI)

We propose to create new learning schemes by learning with multiplicative noise, as described above, where the noise distribution of $R$ is *not* i.i.d across the units (while we keep the noise samples $R_1, \ldots, R_N$ i.i.d. according to the noise distribution). For simplicity, we focus only on Gaussian noise:

$$R \sim \mathcal{N}(\mathbf{1}_D, \lambda\Sigma) \,, \tag{5.3}$$

where $\mathbf{1}_D$ is the constant $D$-dimensional vector with all entries equal to 1, $\lambda \geq 0$ is a regularisation parameter and $\Sigma \in \mathbb{R}^{D \times D}$ is a symmetric positive semidefinite covariance matrix. When $\lambda = 0$, $R$ is almost surely constant equal to $\mathbf{1}$, and we recover the standard learning without noise injection (5.1). When $\lambda > 0$ and $\Sigma = \mathbf{I}$ we learn from i.i.d. multiplicative Gaussian noise, a variant of dropout that was proposed by Srivastava et al. (2014) and shown

to perform very similarly to dropout. When $\Sigma$ is diagonal but not necessarily constant on the diagonal, the amount of noise can vary among units, but the noise is still independent across units.

Our focus in this paper is on non-diagonal covariance matrices $\Sigma$, which create correlations between the noise at different units. We call this setting *structured noise injection (SNI)*. In the case of multilayer neural network, it is natural to create correlations within layers, and not between layers, which translates to a block-diagonal structure for $\Sigma$, where each block corresponds to the units of a given layer. We further consider two flavors of SNI.

### 5.3.1 SNI with fixed noise covariance

The basic flavor of SNI is when we fix the noise covariance $\Sigma$ *a priori* and independently from the data, using for example the structure of the network as prior knowledge. This is a way to input prior knowledge about the problem in the learning algorithm, and has already been proposed as a promising technique in different settings, particularly with binary noise. For example, Tompson et al. (2014) proposed the SpatialDropout method, where entire feature maps corresponding to adjacent pixels are randomly discarded together instead of individual pixels, corresponding to binary SNI with a block-diagonal covariance matrix with constant blocks equal to 1 for each set of pixels in a feature map. Similarly, DeVries and Taylor (2017) applies binary SNI at the input layer of a convolutional network by masking contiguous sections of inputs rather than individual pixels, yielding new state-of-the-art results in image classification. Implementing a SNI strategy with fixed, block-diagonal covariance matrix at the layer level, is only a slight generalisation of standard parameter inference with SGD and backpropagation. For example, Algorithm 5 illustrates the forward pass between layers $l-1$ and $l$ with SNI regularisation, where $\Sigma^{(l-1)}$ is the block of $\Sigma$ corresponding to layer $l-1$.

**Algorithm 5** Feed-forward pass with SNI at layer $l$

**Require:** Mini-batch of outputs from the previous $(l-1)$-th layer $y_1^{(l-1)}, \ldots, y_n^{(l-1)} \in \mathbb{R}^{d^{(l-1)}}$, regularisation parameter $\lambda \in \mathbb{R}_+$, covariance matrix of the noise $\Sigma^{(l-1)}$ **OUTPUT:** The mini-batch of outputs from the $l$-th layer

1: **for** $i = 1$ to $n$ **do**
2: Sample $r_i^{(l-1)} \sim \mathcal{N}(\mathbf{1}_{d^{(l-1)}}, \lambda\Sigma^{(l-1)})$
3: $\widetilde{y}_i^{(l-1)} \leftarrow r_i^{(l-1)} \odot y_i^{(l-1)}$
4: $z_i^{(l)} \leftarrow W^{(l)}\widetilde{y}_i^{(l-1)} + b^{(l)}$
5: $y_i^{(l)} \leftarrow \sigma(z_i^{(l)})$
6: **end for**
7: **return** $y_1^{(l)}, \ldots, y_n^{(l)} \in \mathbb{R}^{d^{(l)}}$

### 5.3.2 Adaptive SNI

Another SNI approach is to define a noise structure with covariance $\Sigma(\mathcal{D}, \theta)$ which may depend on the data and on the model parameters. We refer to this situation as *Adaptive Structured Noise Injection (ASNI)*. An example of ASNI approach, where $\Sigma$ depends on the data $\mathcal{D}$ but not on the model parameters $\theta$, was recently proposed by Aydore et al. (2019): they first use the data $\mathcal{D}$ to identify groups of correlated features, and then perform dropout at the group level, reporting promising empirical results. In other words, they impose a block diagonal correlation structure on the noise, where each block corresponds to a group of correlated features, and the correlation of the noise within a group is 1. While grouping is needed when one wants to perform dropout by group, more general correlation matrices are possible when the noise in Gaussian. An obvious extension of the work of Aydore et al. (2019) is to impose over the noise the same covariance structure as observed in the data.

When a single-layer linear model is considered, as in Aydore et al. (2019), then $\Sigma$ depends only on the data $\mathcal{D}$, but not on the model parameters $\theta$. In the case of multi-layer networks, the covariance of the units at a given internal layer not only depends on the data distribution, but also on the parameters $\theta$ of the models; as a result, the covariance of the noise to be injected in internal layers depends on both $\mathcal{D}$ and $\theta$ in this case.

To solve (5.2) with ASNI, we can still follow a SGD approach where at each step a minibatch of examples is randomly chosen, and a realisation of the noise on these examples is sampled. One difficulty in ASNI is that the statistics of the noise to be injected depends on the data themselves, and on the parameters of the DNN which evolve during optimisation. Similarly to the procedure used in batch normalisation (Ioffe and Szegedy, 2015), we propose to re-estimate the covariance of the noise at each SGD iteration from the mini-batch itself, before sampling the noise on the mini-batch using this structure. Algorithm 6 details the feed-forward pass for one layer, in the case where $\Sigma$ is block diagonal with a block for each layer equal to the covariance of the units in that layer.

We note that in order to sample the noise with a given covariance matrix $\Sigma$, one typically needs to factorise $\Sigma = UU^\top$ by spectral decomposition or Cholesky decomposition (Gentle, 2009), and then get the samples as $U\epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{1}, \mathbf{I})$. This can create significant computational burden, since this must be performed at each SGD iteration, and is one of the reasons why, for example, batch normalisation only scales each feature but does not perform whitening (Ioffe and Szegedy, 2015). In the particular case of ASNI where $\Sigma$ is the covariance of the data, we can get an important speed-up since the estimate of $\Sigma$ on a mini-batch is already factorised as $\hat{\Sigma} = \tilde{Y}^\top \tilde{Y}$, where $\tilde{Y}$ is the $n \times d^{(l-1)}$ matrix of mini-batch outputs, centered and divided by $\sqrt{n}$. In other words, the estimation of $\Sigma^{(l-1)}$ in line 1 of Algorithm 6 can be completely bypassed, and the matrix $U$ in line 4 replaced by $\tilde{Y}$. We note that this is particularly efficient when the mini-batch size $n$ is not larger than the number of units. Finally, a further important speed-up is possible by sampling a single noise vector for each sample in a mini-batch, instead of sampling a different vector for each sample. This results in algorithm 7.

**Algorithm 6** Feed-forward pass with ASNI at layer $l$

**Require:** Mini-batch of outputs from the previous layer $y_1^{(l-1)}, \dots, y_n^{(l-1)} \in \mathbb{R}^{d^{(l-1)}}$, regularisation parameter $\lambda \in \mathbb{R}_+$

1: **OUTPUT:** The mini-batch of outputs from the $l$-th layer
2: Estimate $\Sigma^{(l-1)} = UU^\top$ with $U \in \mathbb{R}^{d^{(l-1)} \times d_U}$ from the batch
3: **for** $i = 1$ to $n$ **do**
4: Sample $\epsilon_i \sim \mathcal{N}(\mathbf{1}_{d_u}, \mathbf{I})$
5: $r_i^{(l-1)} \leftarrow \sqrt{\lambda} U \epsilon_i$
6: $\tilde{y}_i^{(l-1)} \leftarrow r_i^{(l-1)} \odot y_i^{(l-1)}$
7: $z_i^{(l)} \leftarrow W^{(l)} \tilde{y}_i^{(l-1)} + b^{(l)}$
8: $y_i^{(l)} \leftarrow \sigma(z_i^{(l)})$
9: **end for**
10: **return** $y_1^{(l)}, \dots, y_n^{(l)} \in \mathbb{R}^{d^{(l)}}$

**Algorithm 7** Fast Feed-forward pass with ASNI at layer (l)

**Require:** Mini-batch of outputs from the previous layer $y_1^{(l-1)}, \dots, y_n^{(l-1)} \in \mathbb{R}^{d^{(l-1)}}$, regularisation parameter $\lambda \in \mathbb{R}_+$

**OUTPUT:** The mini-batch of outputs from the $l$-th layer

$m^{(l-1)} \leftarrow \frac{1}{n}\sum_{i=1}^n y_i^{(l-1)}$

Sample $r^{(l-1)} \sim \mathcal{N}(\mathbf{1}_n, \lambda \mathbf{I})$

$r^{(l-1)} \leftarrow \frac{1}{\sqrt{n}} \begin{pmatrix} y_1^{(l-1)} - m^{(l-1)} \\ \dots \\ y_n^{(l-1)} - m^{(l-1)} \end{pmatrix}^\top r^{(l-1)}$

**for** $i = 1$ to $n$ **do**

  $\widetilde{y}_i^{(l-1)} \leftarrow r^{(l-1)} \odot y_i^{(l-1)}$

  $z_i^{(l)} \leftarrow W^{(l)} \widetilde{y}_i^{(l-1)} + b^{(l)}$

  $y_i^{(l)} \leftarrow \sigma(z_i^{(l)})$

**end for**

**return** $y_1^{(l)}, \dots, y_n^{(l)} \in \mathbb{R}^{d^{(l)}}$

## 5.4 regularisation effect

It is well known that learning with noisy data can be related to regularisation. For example, adding uncorrelated Gaussian noise to data in ordinary least squares regression is equivalent, in the case of squared error, to ridge regression on the original data Bishop (1995b). The following result clarifies how correlations in the noise impacts this regularisation. We consider a setting with no hidden layer and no bias term, i.e., a simple linear model of the form $f(x) = w^\top x$ where $w \in \mathbb{R}^d$ is the vector of weights of the model.

**Lemma 1.** *Given a training set $(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ with centred inputs ($\sum_{i=1}^n x_i = 0$), and given $R_1, \dots, R_N$ i.i.d. random variables following $\mathcal{N}(\mu, \lambda\Sigma)$ for some $\lambda \geq 0$ and covariance matrix $\Sigma$, the following holds for any $w \in \mathbb{R}^d$:*

$$\frac{1}{N}\sum_{i=1}^N \mathbb{E}\left(w^\top(R_i \odot x_i) - y_i\right)^2 = \frac{1}{N}\sum_{i=1}^N \left(w^\top x_i - y_i\right)^2 + \lambda w^\top (C \odot \Sigma)\, w\,,$$

*where $C = \frac{1}{n}\sum_{i=1}^N x_i x_i^\top$ is the covariance matrix of the data. Furthermore, if $L : \mathbb{R}^2 \to \mathbb{R}$ is a general loss function and for any $y \in \mathbb{R}$, the function $u \in \mathbb{R} \mapsto \ell_y(u) = L(u, y)$ twice-differentiable, then the following holds:*

$$\frac{1}{N}\sum_{i=1}^N \mathbb{E}L\left(w^\top(R_i \odot x_i), y_i\right) = \frac{1}{N}\sum_{i=1}^N L\left(w^\top x_i, y_i\right) + \frac{\lambda}{2} w^\top \left(J(w) \odot \Sigma\right) w + o(\lambda)\,,$$

*where*

$$J(w) = \frac{1}{N}\sum_{i=1}^N \ell''_{y_i}\left(w^\top x_i\right) x_i x_i^\top\,.$$

The proof of these results follows the same marginalisation argument in chapter 2, equation 3.6. Depending on the noise covariance $\Sigma$, we derive several interesting situations:

- In the case of standard dropout regularisation with i.i.d. nodes ($\Sigma = \mathbf{I}$), we recover known results of Wager et al. (2013); Baldi and Sadowski (2013).

- When $\Sigma = \mathbf{1}_d\mathbf{1}_d^\top$, i.e., when the noise is the same for all units, then $\Sigma$ is the neutral multiplication of the Hadamard product. This implies that the regularisation boils down to $\lambda w^\top C w$ in the least squares regression case, and to $\lambda w^\top J(x) w/2$ in the more general case. Interestingly, in the least squares regression with centered data, we have the following:

  **Lemma 2.** *Given a training set $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ with centred inputs $(\sum_{i=1}^n x_i = 0)$, and given $R_1, \ldots, R_N$ i.i.d. random variables following $\mathcal{N}(\mu, \lambda \mathbf{1}_d\mathbf{1}_d^\top)$ for some $\lambda \geq 0$, the following holds for any $w \in \mathbb{R}^d$:*

$$\frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\left(w^\top(R_i \odot x_i) - y_i\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\tilde{w}^\top \tilde{x}_i - y_i\right)^2 + \lambda \tilde{w}^\top \tilde{w}\,,$$

  *where $C = \frac{1}{n}\sum_{i=1}^N x_i x_i^\top$ is the covariance matrix of the data, $\tilde{x}_i$ is a whitened version of $x_i$ for $i = 1, \ldots, n$, and $\tilde{w}$ is the whitened version of $w$.*

  In other words, SNI on centered data is equivalent in the case of squared error to standard ridge regression on the whitened data. Remember that whitening data is obtained by multiplying each vector by a whitening matrix $Z \in \mathbb{R}^{d\times d}$ that satisfy $Z^\top Z = C^{-1}$. There exist an infinite number of possible whitening matrix, a standard choice being $Z = C^{-1/2}$ for ZCA whitening. Hence, in Lemma 2, $\tilde{x}_i = Z x_i$ and $\tilde{w} = Z w$, and the proof of Lemma 2 results from simple algebric manipulations of the results of Lemma 1. Interestingly, when noise injection is done per layer, then Lemma 2 shows that injecting the same noise to all units of a given layer is equivalent to data whitening at the layer input, combined with $\ell_2$ regularisation (a.k.a. weight decay in the neural network terminology). Data whitening is usually associated with high computational cost due to diagonalisation of the covariance matrix, and is replaced in practice by batch normalisation which only normalises the variance of each unit; our analysis suggests that SNI with strong noise correlation within a layer provides a computationally efficient approach to obtain the same result as complete data whitening.

- Finally, we propose $\Sigma = C$ as another interesting structure for ASNI, which generalises the idea of Aydore et al. (2019) to creating noise correlation on correlated units and in case of non-linear models. In the case of least square regression, this is equivalent by Lemma 1 to a regularisation by $w^\top C^{\odot 2} w$, where $C^{\odot 2}$ denotes the Hadamard g of the covariance matrix $C$. While detailed analysis of this choice is complicated by the fact that the eigenstructure of $C^{\odot 2}$ is not easily related to the one of $C$, we show empirically below that it leads to promising results.

## 5.5 Effect on learned representation

While Lemma 1 interprets SNI as regularisation in the one-layer, linear model case, the same analysis can be done at each layer of a multi-layer network and even for non-linear activations. In this case, though, both the inputs of the layer (i.e., the $x_i$'s in Lemma 1) and the weights $w$ are jointly optimised, since the inputs depend on the parameters of the previous layers. Hence SNI may affect the *representation* learned by a multi-layer network.
Let us take the case of least squares regression as an example, where SNI is equivalent to a regularisation by $\lambda w^\top (C \odot \Sigma) w$ according to Lemma 1. When $\Sigma = \mathbf{1}_d\mathbf{1}_d^\top$, we have seen that SNI is equivalent to ridge regression on the whitened data. Hence, any rotation of the inputs has no impact on the model learned, since both the whitening of the $x_i$'s and the ridge penalty on $w$ are invariant to rotation. As a consequence, SNI in that case does not promote

independence of the units in a layer, since any rotation of the units can change the correlation between units without affecting the objective function of SNI regression.

The situation is different for the standard dropout ($\Sigma = \mathbf{I}$) and ASNI with $\Sigma = C$, as the penalty $w^\top (C \odot \Sigma) w$ is not invariant to rotation anymore. Interestingly, the following holds:

**Lemma 3.** *For $\Sigma = \mathbf{I}$ or $\Sigma = C$, where $C$ is the covariance matrix of a set of points $x_1, \ldots, x_n \in \mathbb{R}^d$, it holds that:*

1. $\forall i, j \in [1, d], (C \odot \Sigma)_{ij} \geq 0$,
2. $\sum_{i,j=1}^{d} (C \odot \Sigma)_{ij}$ *is invariant to rotation of the points.*

*Proof.* For the first point, just notice that for $\Sigma = \mathbf{I}$, the entries of $C \odot \Sigma$ are $C_{ii} \geq 0$ on the diagonal, and 0 elsewhere; for $\Sigma = C$, the entries of $C \odot \Sigma$ are $C_{ij}^2 \geq 0$. For the second point, note that for $\Sigma = \mathbf{I}$, the sum considered is just the trace of the covariance matrix $C$, which is invariant to rotation (sum of eigenvalues); for $\Sigma = C$, the sum considered it the squared Frobenius norm of $C$, which is also invariant to rotation (sum of the squares of eigenvalues). □

As the penalty induced by SNI is

$$\lambda w^\top (C \odot \Sigma) w = \lambda \sum_{i,j=1}^{d} (C \odot \Sigma)_{ij} w_i w_j \,, \tag{5.4}$$

Lemma 3 suggests an interplay between the optimisation of the inputs (which impact $C \odot \Sigma$) and the optimisation of $w$. If we fix $w$ and just optimise over a rotation of the inputs, then the penalty (5.4) is just a linear function in $C \odot \Sigma$, which according to Lemma 3 stays in a linear polyhedron defined by linear equalities and inequalities, and one might expect the best rotation to push $C \odot \Sigma$ near the boundary of that polyhedron, where some entries are 0. Of course a more careful analysis is needed to make this reasoning rigorous (in particular, $w$ should also be rotated, and $C \odot \Sigma$ can not span the whole polyhedron), but it may hints that both dropout and ASNI with $\Sigma = C$ tend to create representations with small values in $C \odot \Sigma$. While this only concerns variance terms for usual dropout, a possible benefit of ASNI with $\Sigma = C$ is that it involves all off-diagonal terms $C_{ij}^2$ as well, suggesting that ASNI may create less correlated representations by penalising the correlation among units of a given layer. We study this effect more precisely in the experiments below.

## 5.6 Experiments

### 5.6.1 Simulation

In order to study the performance of ASNI on a toy dataset and a simple model, we use the classical simulation setting proposed by Guyon et al. (2007) for the MADELON dataset. In short, we generate 100 samples for training and 10 000 test samples from 2 balanced classes, and train a *linear model* on the same training set using (1) no dropout, (2) i.i.d Gaussian dropout with different values of $\lambda$, and (3) ASNI using $\Sigma = C$ with different values of $\lambda$. The MADELON procedure allows to vary total number of features, as well as the number of redundant features. We report in Tables 5.1 and 5.2 the test accuracies of the different models, when we vary the total number of features on the one hand, and when we vary the number of redundant features on the other hand. For the regularised methods, the test accuracy is taken over the best regularisation parameter $\lambda$ (same for the shown figures).

We notice that in most settings ASNI performs best, particularly when the total number of features grows. This suggests that ASNI acts as an effective regulariser even for linear models. It also significantly stands out in the presence of redundant features. An intuition is

Table 5.1: Best 10 runs average test classification score (± standard deviation) of a linear model without noise injection, with i.i.d Gaussian dropout, and with ASNI on the MADELON simulation with 10% useful features and no redundant features: varying the total number of features.

| **Total features** | **No drop.** | **i.i.d Gauss.** | **ASNI** |
|---|---|---|---|
| $10^2$ | 66.6 ±2.6 | **68.3** ±2.0 | 68.0 ±2.0 |
| $10^3$ | 68.1 ±2.3 | 68.6 ±2.0 | **68.9** ±2.2 |
| $10^4$ | 55.6 ±2.5 | 54.9 ±2.8 | **56.2** ±2.6 |

Table 5.2: Best 10 runs average test classification score (± standard deviation) of a linear model without noise injection, with i.i.d Gaussian dropout, and Structured dropout (ASNI) on the MADELON simulation with 1000 features and 100 useful : varying the number of redundant features

| **Redundant features** | **No drop.** | **i.i.d Gauss.** | **ASNI** |
|---|---|---|---|
| 0 | 66.6 ±2.3 | **68.3** ±2, 0 | 68.0 ±2.0 |
| 100 | 65.8 ±1.6 | 67.6 ±1.3 | **68.2** ±1.3 |
| 800 | 68.9 ±1.6 | 69.1 ±1.6 | **71.6** ±1.6 |

that ASNI allows us to use the weights of the redundant features in accordance to the useful features they are created from and minimises prediction disagreement among single weights (since it ties weights in the regularisation).

### 5.6.2 MNIST

We now assess the performance of ASNI on image classification, using the classical MNIST benchmark. For simplicity, we train a network with only 2 dense ReLU-activations hidden layers. We do not expect to obtain state-of-the-art results as we do not perform any data augmentation or other regularisation. The goal of this set of experiments is mainly to study the difference of performance and the effect of ASNI on a hidden layer activations compared with independent noise injection. The number of the second hidden layer units $d^{(2)}$ is fixed to the number of classes (10 here), and we vary the number of units in the first hidden layer $d^{(1)}$.

Table 5.3: Best 5 runs average classification score (± standard deviation) on the MNIST dataset of a 2 hidden layer without noise injection, with i.i.d noise injection (Gaussian and Bernoulli dropout), and with ASNI, as one varies the number of units in the first hidden layer.

| $d^{(1)}$ | **No drop.** | **iid Gauss.** | **iid Bern.** | **ASNI** |
|---|---|---|---|---|
| 32 | 93.7 ± 0.2 | 94.2 ± 0.8 | 94.4 ± 0.9 | **95.8** ± 0.4 |
| 64 | 95.8 ± 0.6 | 95.4 ± 0.7 | 95.9 ± 0.6 | **96.6** ± 0.7 |
| 256 | 96.1 ± 0.6 | 97.0 ± 0.7 | 97.4 ± 0.7 | **97.8** ± 0.7 |
| 512 | 96.5 ± 0.1 | 97.5 ± 0.1 | 97.6 ± 0.1 | **98.1** ± 0.1 |
| 1,024 | 96.2 ± 0.1 | 97.6 ± 0.1 | 97.6 ± 0.2 | **98.1** ± 0.3 |

We summarise in Table 5.3 the test accuracy defined as the proportion of well classified examples from the test set, after training the 2 hidden layer network with varying number of units. Figure 5.1 shows the evolution of this test accuracy during the training process of the

model with 256 units in the hidden layer, as a function of the number of SGD iterations.

We see from Table 5.3 that all methods involving noise injection tend to outperform the baseline approach without no regularisation, which confirms the benefits of noise injection for performance. Second, we notice that among the three methods that perform noise injection, ASNI constantly outperforms both Gaussian and Bernoulli i.i.d dropout for small and large number of units. The 2 hidden layers, that seems to overfit even for a small number of units in both hidden layers, has however a better accuracy with larger number of units, which indicates that there is still information to be gained from the data. The network with 64 units however, with ASNI regularisation, seems to capture more information than the network without dropout with 1024 units, which can be largely explained by the quality of representation learnt by structured dropout, as we will show below. Figure 5.1 also shows that ASNI leads to faster convergence, an effect observed as well in batch normalisation (Ioffe and Szegedy, 2015).

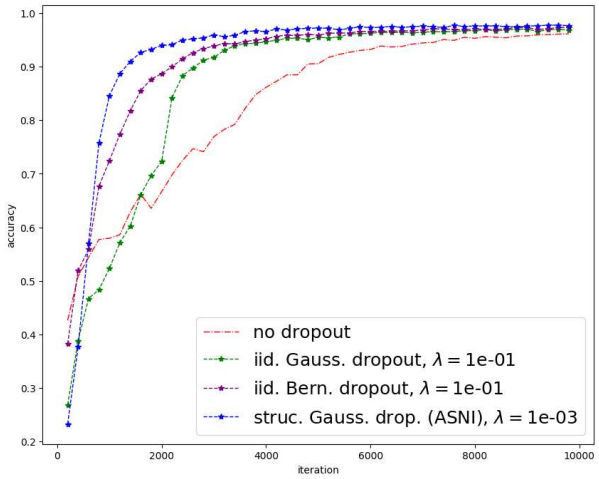



Figure 5.1: Test classification during training for a 2-hidden layers MLP, with 256 units in the first hidden layer, trained without noise injection, with i.i.d. noise injection or with ASNI (with the best regularisation hyper-parameter $\lambda$).

To see the effects of ASNI on units co-adaptation we measure the total correlation of the units' activations at a layer $l$ as the Frobenius norm of the activations correlations matrix $T$ defined as:

$$T_{ij} \frac{\Sigma_{i,j}}{\sqrt{\Sigma_{i,i}\Sigma_{j,j}}},$$

where $\Sigma$ is the covariance of the unit activations introduced in methods description. The evolution of this quantity is shown in Figure 5.2, for the network with 1,024 units in the hidden layer. We see that for all methods, correlations among units tends to decrease during optimisation, which confirms that better performance is obtained when units are less redundant. We also see that adding noise has a dramatic effect on the decrease of correlation, as all three methods regularised by noise injection see their units' correlations decrease much faster and much lower than the un-regularised baseline. Gaussian and Bernoulli i.i.d. noise injection lead to a very similar curve, confirming that both methods behave very similarly. Finally, we observe that ASNI lead s to faster decrease of the correlation matrix norm, and that it reaches after $10^4$ iterations a lower value than all other methods. This empirically confirms the active role played by non-i.i.d. noise injection, in particular ASNI, in promoting non-redundant representations.

To further study the quality of the representations learned by different methods, we visualise the vectors of hidden layer activations on the test set using t-SNE in order to assess how well the different classes are separated. Figure 5.3 shows the 2-component t-SNE

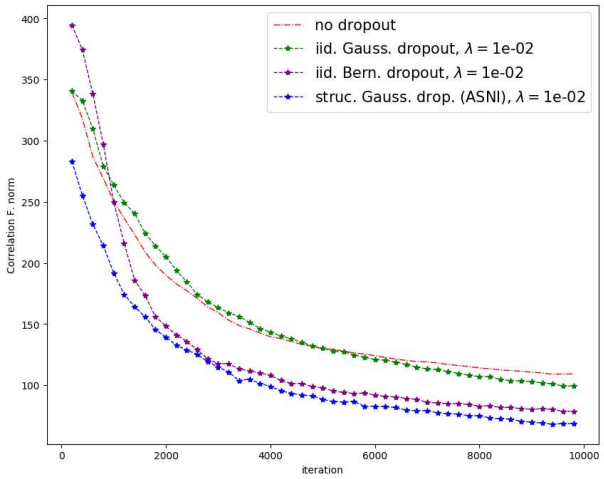



Figure 5.2: Correlation matrix norm of the first hidden layer activations during training for a 2-hidden layers MLP trained without noise injection, with i.i.d. noise injection or with ASNI (with the best regularisation hyper-parameter), with 1,024 units in the first hidden layer.

embeddings of the second hidden layer activations (32 in this case) applied on a sample of 1,000 test samples, trained respectively without noise injection, with i.i.d. Bernoulli dropout, and with ASNI. Visually, we see that the class are better separated in the representation learned

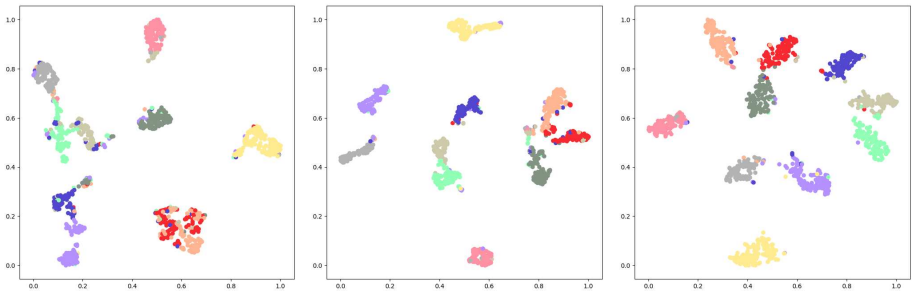


Figure 5.3: t-SNE visualisation of the second hidden layer activations for 1,000 MNIST test images, for a 2 hidden layers MLP with 32 units on the 1st layer. We compare a network trained without noise injection (upper left), with i.i.d Bernoulli dropout (upper right), and with ASNI (bottom). The points are colored according to the class of the images.

by ASNI than by the other methods. To quantify this visual impression, we measure the quality of the representation by computing the Silhouette coefficient of each t-SNE embedding (Rousseeuw, 1987). A larger Silhouette value indicates that the representation is better at recovering the known classes of images (Chen et al., 2002). We report in Table 5.4 the mean silhouette coefficients over all test samples for 10 clusters with respectively 32, 256 and 1,024 units in the first hidden layer. These results confirm what we qualitatively observed in Figure 5.3, namely, that noise injection improves the quality of the representations compared to the non-regularised version, and more importantly that ASNI clearly outperforms i.i.d. noise injection in all settings. An example of the Silhouette analysis on MNIST with the 2 hidden layers MLP with 32 units on the 1st layer is visualised in figure A.2.

In order to see which layers are decorrelated by (ASNI), we train the same 2 hidden layers architecture trained on MNIST, but we apply independent noise injection or ASNI only on one layer for each experiment. The layer can be :

- The input layer

Table 5.4: Average Silhouette coefficient scores of the last hidden layer t-SNE embeddings on MNIST test dataset, without noise injection, with i.i.d Gaussian and Bernoulli dropout, and structured dropout (ASNI). .

| $d^{(1)}$ | **No drop.** | **iid Gauss.** | **iid Bern.** | **ASNI** |
|---|---|---|---|---|
| 32 | 0.60 | 0.57 | 0.53 | **0.69** |
| 256 | 0.58 | 0.63 | 0.63 | **0.72** |
| 1024 | 0.58 | 0.73 | 0.74 | **0.80** |

- The first hidden layer
- The second hidden layer

The evolution of the first and second hidden layer's correlations during training (represented again by the Frobenius norm of the activations correlation matrix at iteration $t$) for each experiment is shown in figure A.3.
A first important observation is that i.i.d. Bernoulli and gaussian dropout do not necessarily reduce the correlations between units, and thus in this sense do not always prevent co-adaptations in terms of activations correlations. ASNI on the other hand forces units to be more independent when it is applied on that layer, but does not reduce cross-correlations completely to 0 since the norm of the correlation matrix continues to decrease during the training. In this sense, ASNI is different from the whitening techniques mentioned in the introduction in that it does not explicitly change the input and does not force units to be independent such as batch normalisation and its decorrelated variants, but rather encourages units through the structure of dropout to be more independent. Interestingly, figure A.3 also shows that in the case of dense multilayer networks, applying (ASNI) on one layer does decorrelate the activations of that layer but not the next layers, as it can be hinted by the studied property of ASNI in hidden layer network in the previous section. However, applying ASNI on the second hidden layer decorrelates its activations but also the first hidden layer activations.

Our experiments also show that ASNI leads to a higher level of sparsity than independent noise injection as shown by the figure A.4, that reports the histogram of the obtained activation after the training the same 2 Re-LU hidden layers network without dropout, with i.i.d. gaussian or Bernoulli dropout, or with ASNI. The visualised histogram distributions confirm findings in (Srivastava et al., 2014) that dropout may lead to sparser representations. However, we can see that ASNI provides a sparser activations distribution, while improving on accuracy as previously shown in table 5.3. One might expect even a link between sparsity of the activations, the accuracy of the network and particularly the representation quality of the hidden layers, that might be worth further investigation. We also notice that Bernoulli dropout and its gaussian variant result in a similar level of sparsity, in this experiment at least, which leads to think that this effect is independent from the sparsity of the multiplicative noise itself.

### 5.6.3 CIFAR10 and CIFAR100

Finally, we compare ASNI to i.i.d. noise injection on a more realistic setting, namely, a LeNet convolutional network architecture (LeCun et al., 1998) with 4 convolutional layers followed by 2 dense layers tested on the CIFAR10 and CIFAR100 datasets. We again compare the different noise injection schemes applied on the 2 dense hidden layers, without data augmentation or additional regularisation.

Table 5.5 summarises the test accuracy reached by the different training procedures. We again observe that all noise injection methods outperform the baseline, that Gaussian and Bernoulli i.i.d. dropout behave very similarly, and that ASNI has the best performance for these datasets. We also notice that ASNI has less variance in performance compared to all other methods, which might be explained by the faster convergence observed already in MNIST experiments.

Table 5.5: Best 5 runs average test classification score (± standard deviation) of LeNet without noise injection, with i.i.d. noise injection (Gaussian and Bernoulli dropout), and with ASNI on CIFAR10 and CIFAR100 benchmarks. .

| Data | No drop. | iid Gauss. | iid Bern. | ASNI |
|---|---|---|---|---|
| CIFAR10 | 66.5 ± 0.1 | 67.9 ± 0.3 | 67.7 ± 0.4 | **68.3 ± 0.2** |
| CIFAR100 | 32.9 ± 0.2 | 33.8 ± 0.5 | 33.8 ± 0.5 | **34.4 ± 0.3** |

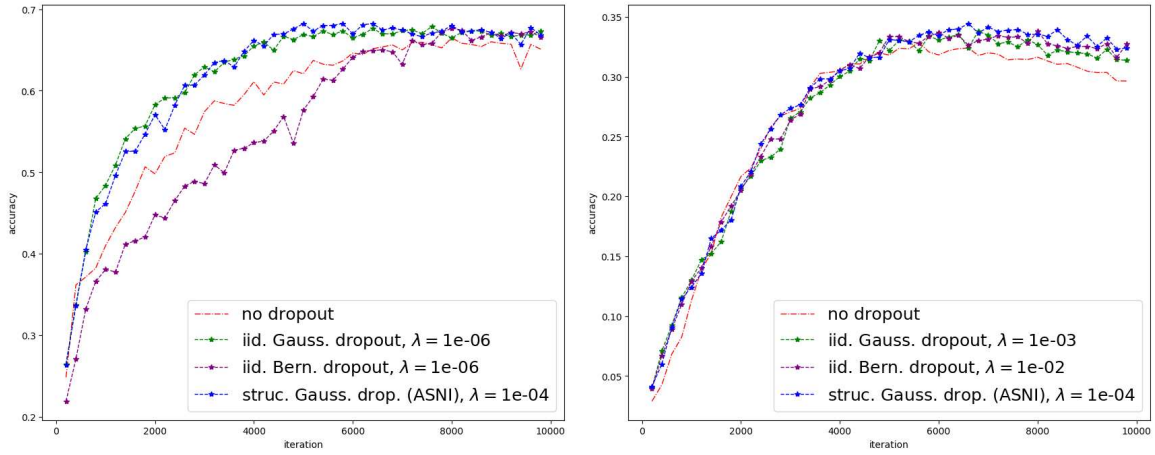



Figure 5.4: Test classification during training for LeNet on CIFAR10 (left) and CIFAR100 (right), trained without noise injection, with i.i.d. noise injection or with ASNI (with the best regularisation hyper-parameter $\lambda$), .

As for the MNIST experiments, we also measure the amount of correlations between unit activations, evaluated by the Frobenius norm of the correlation matrix, and show how it evolves over training for the different methods in Figure 5.5. We notice that standard Bernoulli dropout has a weaker effect on reducing correlations than other methods on CIFAR10, but that overall all methods significantly reduce correlation during training. After convergence, ASNI keeps a small advantage on both datasets in terms of correlation level reached. As shown in Table 5.6, the representation learned by ASNI has also a larger Silhouette than other methods on the test sets.

Table 5.6: Average Silhouette coefficient of LeNet's last hidden layer t-SNE embeddings on CIFAR test datasets, without noise injection, with i.i.d Gaussian and Bernoulli dropout, and with ASNI. .

| Data | No drop. | iid Gauss. | iid Bern. | ASNI |
|---|---|---|---|---|
| CIFAR10 | 0.38 | 0.43 | 0.42 | **0.48** |
| CIFAR100 | 0.35 | 0.37 | 0.36 | **0.38** |

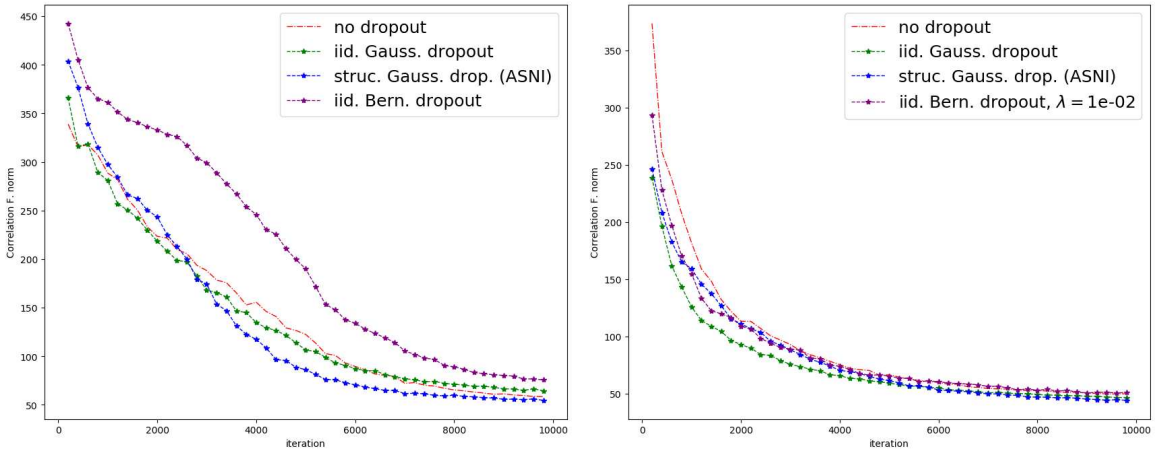



Figure 5.5: Correlation matrix norm of the first dense hidden layer activations with LeNet, with either no noise injection, iid noise injection or ASNI, during training on CIFAR10 (left) and CIFAR100 (right).

## 5.7 Discussion

We proposed new regularisation schemes that generalise dropout, by creating correlations between the noise components. We focused particularly on ASNI, an adaptive approach that replicates the structure of the data correlation in the noise correlation. We showed both theoretical and empirical results suggesting that ASNI improves the representation and performance of shallow and deep neural network, while maintaining computation efficiency. The ASNI framework opens new research directions. First, one might consider different ways to create the noise correlation structure, using for example the structure of the network, or may even think about *learning* it. Second, while Gaussian noise is convenient to impose a particular correlation structure, discrete noises such as binary noise can be computationally advantageous; sampling binary random variables with a given covariance matrix is however not an easy task (Leisch et al., 1998; Preisser and Qaqish, 2014), and progress in that direction may be directly useful for DNN regularisation.

## 5.8 Availability

All code concerning the real data experiments is available at `https://github.com/BeyremKh/ASNI`

# Chapter 6

# Conclusion

Despite its late emergence since only a few decades, machine learning, as an interdisciplinary applied scientific field that stemmed from artificial intelligence as the product of a marriage between statistics, optimisation and computer science, witnesses at the present day a large and growing interest not only from the different scientific communities, but also from the engineering and business world, and most importantly from young students, such as myself (if I might use this relative term). This interest, I believe, has multiple roots: first, I do think that the recent success of the field, namely in tasks that were thought to be only achievable by human minds (Mnih et al., 2015; He et al., 2015) is the most important reason, especially that these successes have multiplied in the last years. As mentioned in the introduction, part of this success is due to the development of more complex models such as multilayer neural networks, but also to technology development resulting in a decrease in memory and storage prices allowing for increase in volume of data and the increase in computational power [1]. This brings to the second reason for the rise of interest in the domain in my mind, that we have briefly discussed in the context of bioinformatics in the introduction, which is the proliferation of datasets provided by the increasing availability of precise measurement and of high-throughput technologies (Reuter et al., 2015). These datasets are typically high-dimensional and might suffer from a high noise-to-signal ratio (Donoho, 2000). This brings us to the third motivation and interest of research in the field, which is finding innovative ways to dig into the signal to provide meaningful insights and develop predictive models that are capable of being accurate without being fooled by the amount of present noise, in order to be useful on unseen data, which we have summarised in the introduction of this manuscript.

This was at least the beginning of my thesis story, while being confronted to a real prediction problem and facing the problem of overfitting, as described in the first chapter, through the *Rheumatoid Arthritis challenge*. I was convinced after this interesting experience, that I want to work on new ways to perform regularisation which can be at the same time *generic* and fit different datasets without being *ad hoc* tricks, and at the same time that respect the specificity of these datasets and the prior expert knowledge one has about these datasets and the corresponding prediction targets.

This led me to dig into the old technique of *noise injection*, which has witnessed a re-emergence, in the context of deep learning for instance, in the years just preceding the start of my thesis (Vincent et al., 2010; Hinton et al., 2012b). The regularising effect of adding noise stochastically to the input or the learning algorithm fascinated me, as it seemed to inherit the beauty and the mystery of randomisation in mathematics and computer science (Gallicchio et al., 2017). I found randomised algorithms to be indeed well-suited in probabilistic problems much more than in computational number theory, and it was very interesting to discover that randomisation was not only a mere speed-up or a tool to derive simpler algorithm but rather maybe an important step towards building intelligent machines. I have in particular read and used with great interest the work of *Leo Breiman* that made extensive use of randomness and

[1] https://hblok.net/blog/posts/2015/12/27/historical-cost-of-computer-memory-and-storage-3/

data perturbations to design novel, efficient and popular algorithms (Breiman, 2001, 1996b). I started not only to be charmed by the idea of noise injection in the input , but convinced by its efficiency as one of the very important ideas to study and expand at the present day in machine learning, both supervised and unsupervised. However, at the start, I found myself dismayed on one hand by the lack of theoretical ground that probably nourished the magical mystery around the use of randomness but that showed me how hard is analyse an algorithmic model, a feeling that was at least shared among some colleagues that worked for instance on the analysis of the *Random Forest* model (Breiman, 2001; Scornet et al., 2015). One the other hand, as it is often the case in applied research, the success of randomised models led to a spread of related variants that claim to be more efficient than one other, which I spent some, sometimes fruitless, efforts trying to study and implement. Focusing on the particular study of input noise injection as a regulariser in the case of supervised learning and linear models, and the comparaison between additive and multiplicative cases with the landmark paper of Wager et al. (2013) guided my progress and provided me with sufficient insights, that have been mainly described in chapter 3, to build interesting and theoretically grounded generalisations and subsequent variants of INI methods, in the case of linear models with the particular application of single-cell RNA sequencing classification and biomarkers discovery in chapter 4, and in the case of multilayer dense and convolutional neural networks in general in chapter 4.

In the remainder of this chapter, we summarise our contributions in each chapter, state some corresponding general insights gained from these investigations and interesting future directions that would be worth pursuing.

### The RA challenge and how to avoid overfitting:

In chapter 2, we summarised the experience of our team *"Outliers"* in the Rheumatoid Arthritis DREAM challenge which consisted in designing predictive models from genotype and clinical data of Rheumatoid Arthritis patients in order to predict their response to anti-TNF treatment (as a continuous or a categorical variable). Among the 73 participating teams, we succeeded to build the best performing final models in terms of accuracy on the unseen CORRONA test set, with an AUC of 0.62 for the classification task and a correlation of 0.4 for the regression task. Our best full model did not include the genotype SNP information, which was later found by a comparative analysis all teams results to not to convey any improvement, despite a significant genetic heritability estimate of treatment non-response trait (Wray et al., 2007). More details and comparative results can be found at chapter 2 and in our published paper (Sieberts et al., 2016).

These results show that even with thousands of samples, available biological prior knowledge and literature and different sources of data, predicting a continuous or a categorical related target is not an easy task. We believe indeed that one of the most important conditions to the success of a prediction task are the quality of the data and its relatedness to the problem at hand. The quality of the data is a broad term that can include many features, as for instance: Power of the data in terms of signal, but also quality of preprocessing such as normalisation, batch effects (or any additional sources of variation) and their effect on the outcome, relatedness of the data to the problem, objectivity of the measured variables, among other features.

Indeed, although SNP data here were found to be robustly correlated with Rheumatoid Arthritis (see introduction of chapter 2 and (Sieberts et al., 2016)), this does not induce an immediate relationship between SNP data and RA patients' response to treatment. Another pitfall of the data is the very measure of the treatment outcome, which we didn't discuss in depth in chapter 2. If we actually look at the DAS28 : It is a composite score derived from 4 measures, which are then integrated following some mathematical formula:

1. Count the number of swollen joints (out of the 28).

2. Count the number of tender joints (out of the 28).

3. Take blood to measure the erythrocyte sedimentation rate (ESR) or C reactive protein (CRP).

4. Ask the patient to make a "global assessment of health" (indicated by marking a 10 cm line between very good and very bad).

These measures, as we see, include for instance a qualitative and subjective measure about *pain* and asking the patient for a *global assessment of health*, they also require the doctor to look at 28 joints and decide whether the joint is swollon or tender (see figure 2.1 in chapter 2). the DAS28 score has not been adopted in day-to-day (non anti-TNF) practice by all rheumatologists in the UK for example. This is in part because there are some pitfalls in the interpretation of the score. For instance if you never have a very high ESR blood result even momentarily, or if your RA particularly affects the feet (these are not included in the 28 joint count) the score may be misleadingly low. It can also be difficult to decide whether an individual joint is swollen or tender, and this uncertainty may lead to misleading variability in the score. Studies have already showed that DAS28 "may not be an optimal tool for assessment of remission in RA" (Mäkinen et al., 2005). Such issue may actually also arise in other fields such as clinical psychology or text-based emotion prediction for instance (Alm et al., 2005). One suggestion that did not appear in (Sieberts et al., 2016) would be to use multiple indicators in order to account for the variability of the outcome, such as physical function and structural joint damage in the case of RA remission (Paulus, 2004), could make the task itself easier and thus the predictions more robust, by using frameworks such as multitask learning (Yuan et al., 2016). Another strategy would be to consider ranking-based machine learning approaches, where patients can be ranked via the relative severity of the disease or the relative response to treatment and then predict the ranking for the unseen test dataset, which might be a more objective approach (Liu, 2009).

Facing a problem of this sort, practitioners might be tempted by building more complex models to capture eventual variabilities of the target measure and build more precise predictors. Eventually, and in an ironic way, this might end up overfitting on the training set, which was the case of many teams in the RA challenge, as exhibited in chapter 2. We found that overall, some best practices that we advice in complex prediction problems include:

- The use of simple models permits a more straightforward evaluation of the dataset and a higher interpretability of the selected features by the model. In the case of strong overfitting, a regularised linear model is also more robust.

- The use of prior knowledge based on biological literature can tremendously improve the model accuracy and reduce computational overhead (e.g. feature selection).

- A careful pipeline including cross-validation at each model-building step is mandatory.

Finally, we believe that the organisation of these crowdsourced evaluations and modelling challenges provide some real additional benefits to research, such as assembling the most comprehensive datasets, comparing a wide variety of state-of-the-art models that can not be implement by an individual or a handful of individuals in such a short time, as well as assessing with more confidence the outcome and the power of the particular study and the possibility of improving over the individual proposed models by building an ensemble model that might provide better accuracy (see introduction and (Sieberts et al., 2016)). Beyond immediate benefits, insightful discussions between teams might shed light on subsidiary and unseen questions that might be important, and encourage future international collaborations.

**Input Noise Injection as a regulariser:**

In chapter 3, we focused on a particular regularisation technique in the context of supervised learning, namely Input Noise Injection (INI). We provided a brief overview of the different forms under which INI can be performed, following different frameworks. These frameworks include the Bayesian setting, robust optimisation, learning from distributions, ensemble learning and the expected risk minimisation framework on which we focus as the main framework in which this regularisation has most been developed. This brief review already revealed interesting directions that may be explored to complement the INI picture and left me with some questions. In the distributions learning framework for instance, is there a link or a certain equivalence between the use of mean embedding, the main tool to convey INI into the learning problem, and a certain class of regularisers? One particular case hints that the answer to this question can be positive, since it can be shown, via the convolution theorem of Gaussian distributions, that the mean embedding of a Gaussian RBF kernel corresponds to a Gaussian RBF kernel with a larger data-dependent bandwidth (Muandet et al., 2012).

In the ERM framework, we re-formulate the INI procedure in its generality (which has been described in the case of linear models and general noising functions in (van der Maaten et al., 2013; Wager et al., 2013) , and in the case of neural netowrks and particular additive or multiplicative in the remaining literature (Bishop, 1995b; Baldi and Sadowski, 2014; Noh et al., 2017)) and describe the different implementations of gradient descent with INI. In particular, exact and approximate marginalisation, not only provide a deterministic alternative to minimise the INI empirical risk, but also shed light on the effect of INI as a data-dependent regulariser, in particular for non-quadratic loss and multiplicative noise. Although very interesting, this second order Taylor approximation has been shown to provide some wrong insights about multiplicative noise both in linear models (Helmbold and Long, 2015) and multilayer neural networks (Helmbold and Long, 2017). This led us to the design of another simple approximation of the particular case of *dropout*, which is equivalent to scaled multiplicative Bernoulli noise when it is only applied on the input layer. This approximation provides another deterministic alternative to the stochastic approximation of INI and reveals interesting insights on the effect of INI as emphasising the robustness of the model to deletion of subsets of features at training. We then presented several experiments on simulated and real datasets that have shown that:

- As hinted by our approximation, dropout can be beneficial in the setting of rare and useful features and in the setting of many redundant features.
- Our novel approximation improves on the second order Taylor approximation in the case of linear logistic loss in terms of test accuracy in the case of rare and useful features.
- We didn't observe a significative difference between dropout and multiplicative Gaussian INI in terms of efficiency in preventing overfitting.
- Depending on the dataset, INI is not always more effective than $\ell_2$-norm regularisation for preventing overfitting, and multiplicative INI is not always more effective than additive INI. This was not related to the task but rather to the data structure.
- No INI scheme provided sparse models even in a setting where the true model is sparse.

These first insights are only the beginning of research directions paved with interesting questions. Indeed, although research in noise injection in either the input data or the models has been active again in the last years and has led to several new variants as previously discussed, many intuitions such as the stability and invariance intuitions around INI are still not understood and remain at the intuition level (e.g. not backed neither by theory, nor by strong empirical evidence). Some of these questions, from the simplest to the most intricate in our opinion, are:

- How does dropout and other INI methods behave on biological datasets other than gene expression microarrays?
- Does our approximation enjoy some of the properties of the dropout regulariser?
- When does INI result in feature selection? Or alternatively how to make embed feature selection in INI?
- Which noise distribution and which noise function is most suitable to a given dataset, data distribution and a loss function?

We partially address some of these questions in the following chapters, by showing how dropout might suit some particular kind of data, and how changing the distribution of the injected noise might improve accuracy results and have an effect on the model extracted features.
We end the conclusion concerning this chapter by this quote in (Baldi and Sadowski, 2014), who was a pioneer in studying dropout as described in section 3.5: *"at first sight dropout seems like another clever hack. More careful analysis, however reveals an underlying web of elegant mathematical properties. This mathematical structure is unlikely to be the result of chance alone and leads one to suspect that dropout is more than a clever hack and that over time it may become an important concept for AI and machine learning."*

### DropLasso as a robust variant of Lasso for single cell RNA-seq data

In chapter 4, we further focused on dropout as an INI regularisation technique, by designing and applying a generalisation of dropout it in the context of linear models on a recent and increasingly popular type of biological datasets provided by the technology of single cell RNA sequencing.

From a bioinformatics point of view, we developed in this chapter a novel method that is useful not only to classify cells into predefined classes, but also to learn identify potential biomarkers of this classification, that is a molecular signature of the particular studied phenotype. From a machine learning point of view, this method, that we term DropLasso, is on one hand a generalisation of the dropout regularisation technique in linear model that allows for embedded feature selection, and on the other hand a generalisation of the $\ell_1$-norm regularisation (Lasso) by blending it with INI. In order to solve the DropLasso objective efficiently, we designed a novel proximal stochastic gradient descent and an efficient implementation for the regularisation path for different values of the regularisation parameters.

We study DropLasso and relate it to the elastic net regularisation which was shown to improve the stability and generalisation performance of $\ell_1$-norm regularisation (Zou and Hastie, 2005). Results on single cell RNA-seq datasets experiment we performed showed that DropLasso outperform Lasso in terms of accuracy, and although it usually does not have a better accuracy than dropout and elastic net regularisations, it performs a more efficient feature selection and leads to the discovery of potentially interesting biomarkers related to the studied problem.

This work was first guided by the invariance and the data augmentation intuitions mentioned in the previous chapter, in which dropout simulates *pseudo-examples* that could have been the result of the same experiment with additional *dropout events* and that share therefore the same class to be predicted. The success of such these intuitions in this case is rather encouraging and paves the way for other applications to other supervised learning tasks on biological datasets. The main challenge remains to characterise the noise by which the outcome to be predicted is invariant. Another interesting direction is to make use of prior knowledge about gene networks in order to improve our method as promoted by structured sparse regularisation methods on standard RNA-seq (Rapaport et al., 2007). This can be done for instance by proposing another generalisation of INI where the injected noise follows

a predefined structure that can be related to the graph Laplacian of gene regulatory network, as it has been shown that such a Laplacian regularisation can improve the classification performance and feature selection (Rapaport et al., 2007), although it is not yet clear if this method does always improve the model (Lavi et al., 2012). This idea will be explored and applied in a particular case where structure will be learnt from the data in the next chapter.

### Adaptive structured noise injection for shallow and deep neural networks

In chapter 5, we proposed a generalisation of dropout and other noise injection schemes for shallow and deep neural networks, where the noise is not necessarily drawn from an independent identical distribution but follows a structure that can be fixed or learnt automatically during training. We call this generalisation Structured Noise Injection (SNI). When the structure is learnt during training we term it as Adaptive Structured Noise Injection (ASNI). We theoretically justified the use of such a structure, and show on several benchmark image datasets that:

1. It can better prevent overfitting than Independent Noise Injection strategies.
2. It can better disentangle the hidden layer representations without hurting their quality.
3. It can lead to sparser hidden layer representations.

Our aim here was mainly to show that it is possible to generalise stochastic regularisation by noise injection, and that the use of structured noise injection can open new interesting directions such as building better features. It will now be interesting to apply (ASNI) to state-of-the-art networks, and also adapt it and apply it to Recurrent Neural Networks and auto-encoders for instance (for which the application is straightforward). Another application of ASNI to be explored is network compression after training, since it seems that our method is able to sparsify the unit activations and compress information in the hidden layers. Finally, Structured Noise Injection opens the way for experimenting with other structures that might be of interest in other applications. It seems indeed that creating a duality between noise structure and model information can be very interesting. An example that can be explored in future work is when we want a certain structure for hidden layer units that is either known or imposed by the problem context and invariances.

### A final note

Allowing myself to end on a subjective note, I see today the field of machine learning as a successful, if not one of the most successful field, in bringing scientific disciplines together for building elaborate tools to automate learning on one hand, and enhancing human knowledge on the other. My work however did not only shed light the bright side, as to say, fortunately. It also revealed to me several paradoxes that the field is witnessing, from its growing popularity and number of research papers to the lack, until today, of solid theoretical work in sensitive areas. Talking to known researchers, I recall the confidence of *Yann Le Cun* and *Jürgen Schmidhuber* saying "supervised learning is over", probably due the the recent practical successes that were mentioned also in this manuscript, and the growing number of datasets that would allow better accuracy. Fundamental questions are however for me still open and maybe their difficulty lies in their simplicity, as it is often the case in mathematics and more generally in science. These questions do not only have in mind the practical implications of machine learning models, but also the modelling and understanding of computational learning in general. Some of these questions are:

- Given a dataset and a model that is overfitting, what is the best regularisation (in terms of generalisation performance) that should be applied?
- Given a labeled dataset, what is the best accuracy that one can ever achieve?

- Why do correlations of one layer units decrease in the manner we have seen while training a multilayer neural networks?
- How much is our understanding of simple models such as linear models transferrable to more complex multilayer neural networks? Is there a new learning theory for multilayer neural networks?

Ultimately, an interesting progress of the field will be a progress of its foundations, such as the integration of more developed mathematical tools either to understand the current framework (for instance the ERM framework), or to build new frameworks. One of the basic tools that were needed in this thesis was the understanding of the expectation of a non linear function of a simple random variable. We have seen for instance how a second Taylor approximation can be misleading in this case.

We hope that the works in this thesis will inspire new fruitful ideas. We suggest, if ever supervised learning really vanished, that another interesting direction to be explored is the application of INI methods and our variants in new unsupervised learning tasks. An additional direction that particularly stems from structured noise injection is the impact of INI on feature learning and interpretation, which is as important as generalisation accuracy for several fields such as bioinformatics. We think that INI will still be an active field in the next decade, as our guess is that there should be more intimate links between randomness, statistical learning and automatisation.

# Appendix A

# Supplementaries

## A.1 Supplementary tables

Table A.1: A summary of the real-word sc-RNA seq datasets used for DropLasso.

| ***Real Data*** | *Nb. train samples* | *Nb. test samples* | *Nb. classes* | *Dimension* | *Source* |
|---|---|---|---|---|---|
| *IMDB* | *25,000* | *25,000* | *1,000* | *2* | https://keras.io/datasets/ |
| *Reuters* | *8,982* | *2,246* | *19,449* | *46* | https://keras.io/datasets/ |
| *MNIST* | *60,000* | *10,000* | *784* | *10* | https://keras.io/datasets/ |
| *CIFAR10* | *50,000* | *10,000* | *3072* | *10* | https://keras.io/datasets/ |
| *CIFAR100* | *50,000* | *10,000* | *3072* | *100* | https://keras.io/datasets/ |
| *WANG* | *286* | - | *7,910* | *2* | (Wang et al., 2005) |
| *VANT* | *295* | - | *7,910* | *2* | (Van De Vijver et al., 2002) |

## A.2 Supplementary figures

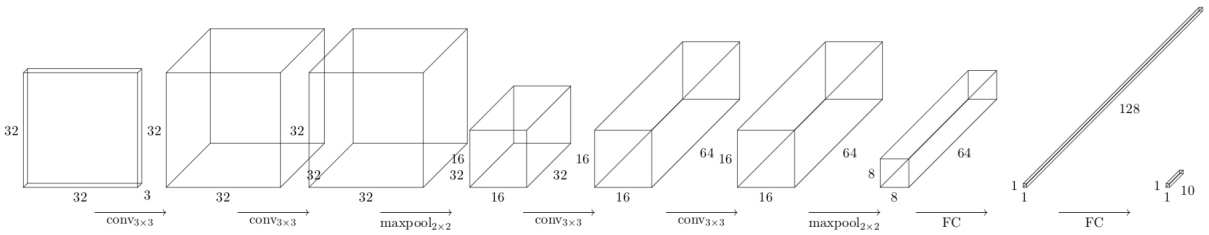



Figure A.1: LeNet architecture for CIFAR10 and CIFAR100.

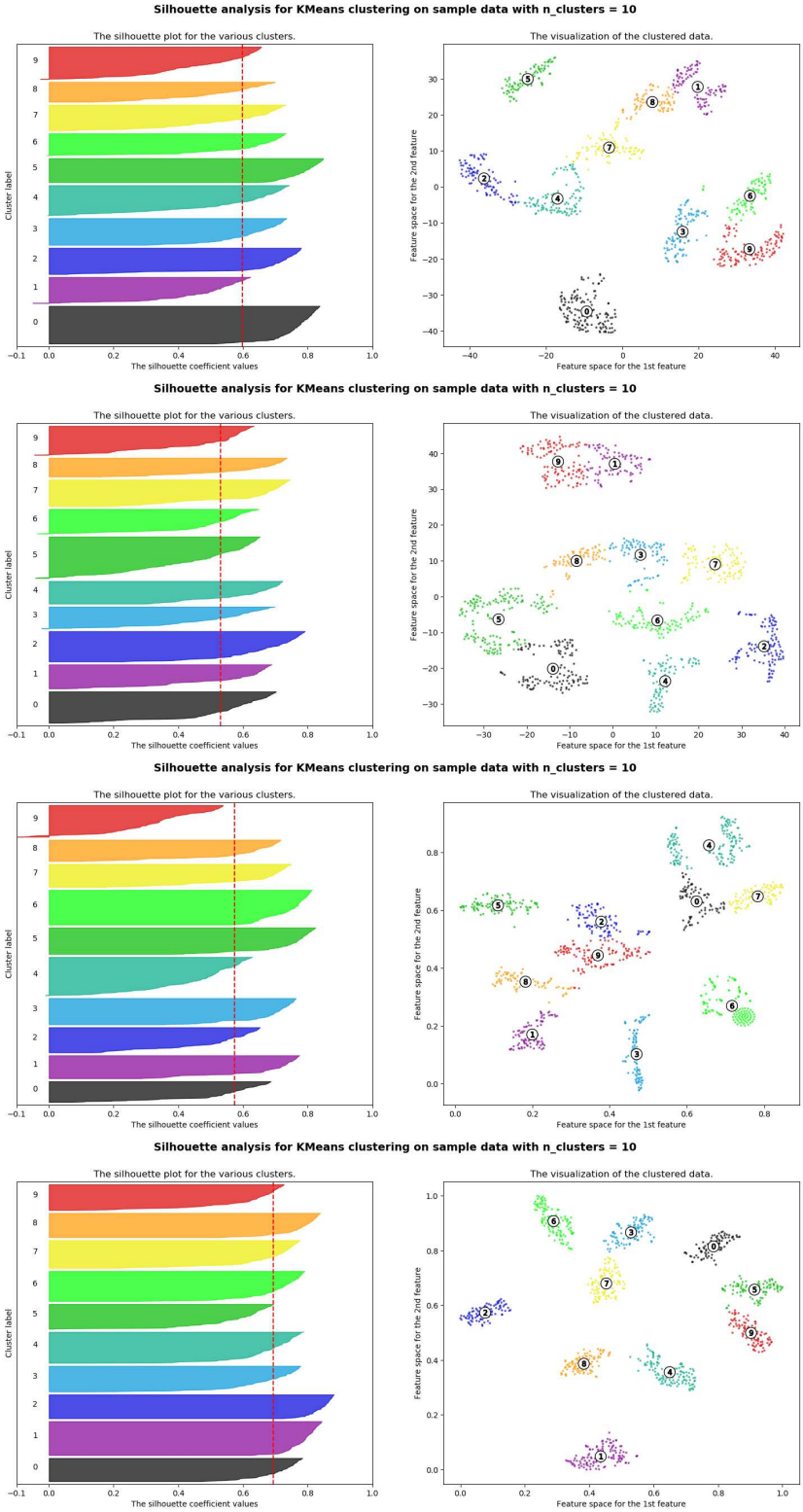



Figure A.2: Silhouette plots for the t-SNE embeddings of the first hidden layer activations on the test data (2 hidden layers MLP with 32 units on the 1st layer). From above to below respectively: without noise injection, with i.i.d gaussian dropout, with Bernoulli dropout, and structured dropout (ASNI)

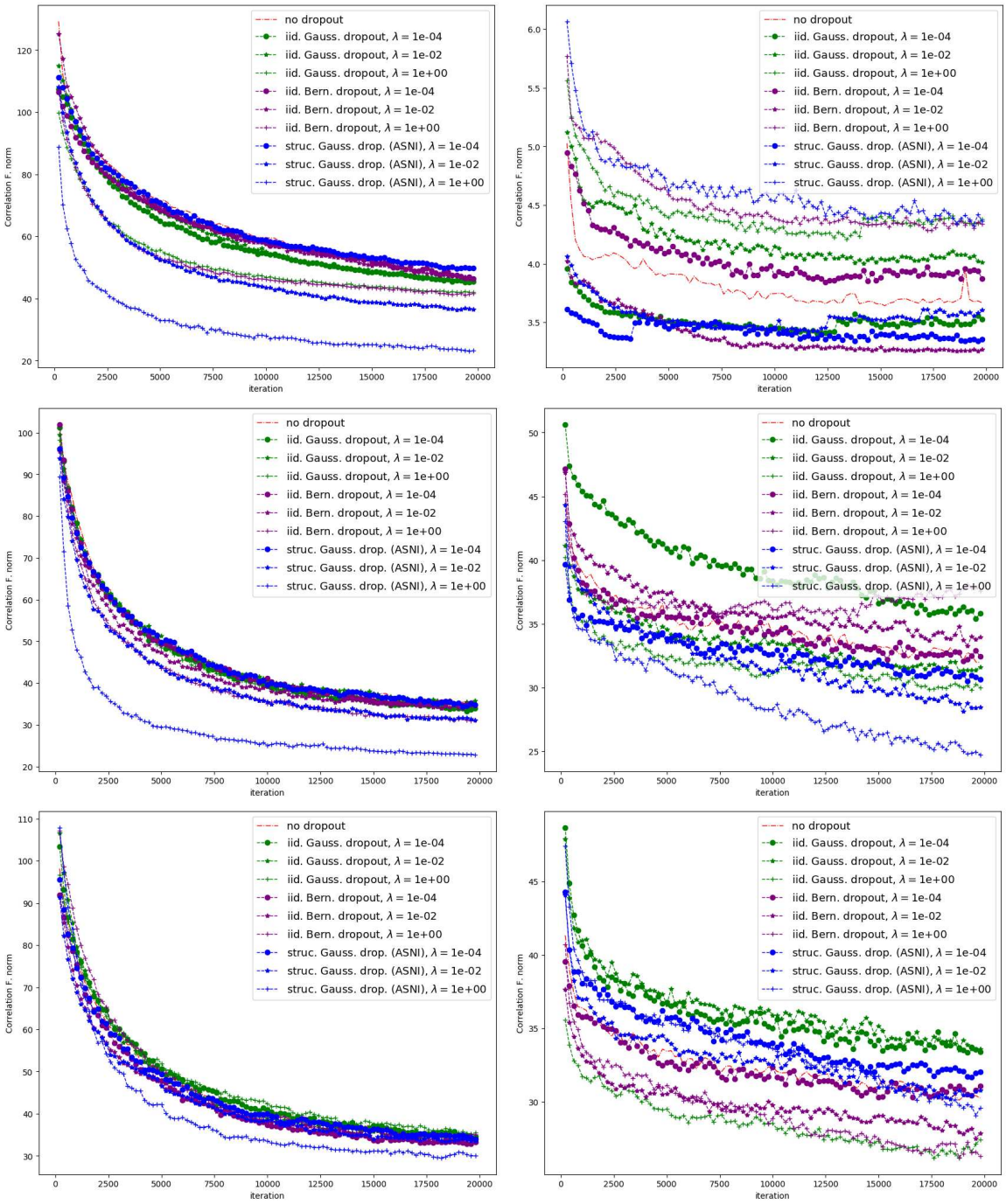



Figure A.3: Correlation matrix norm of the first (left figures) and second (right figures) hidden layer activations during training for a 2-hidden layers MLP with no noise injection, with iid noise injection and ASNI, applied on the first hidden layer only (above), on the the second hidden layer only (middle) or on the input layer only (below).

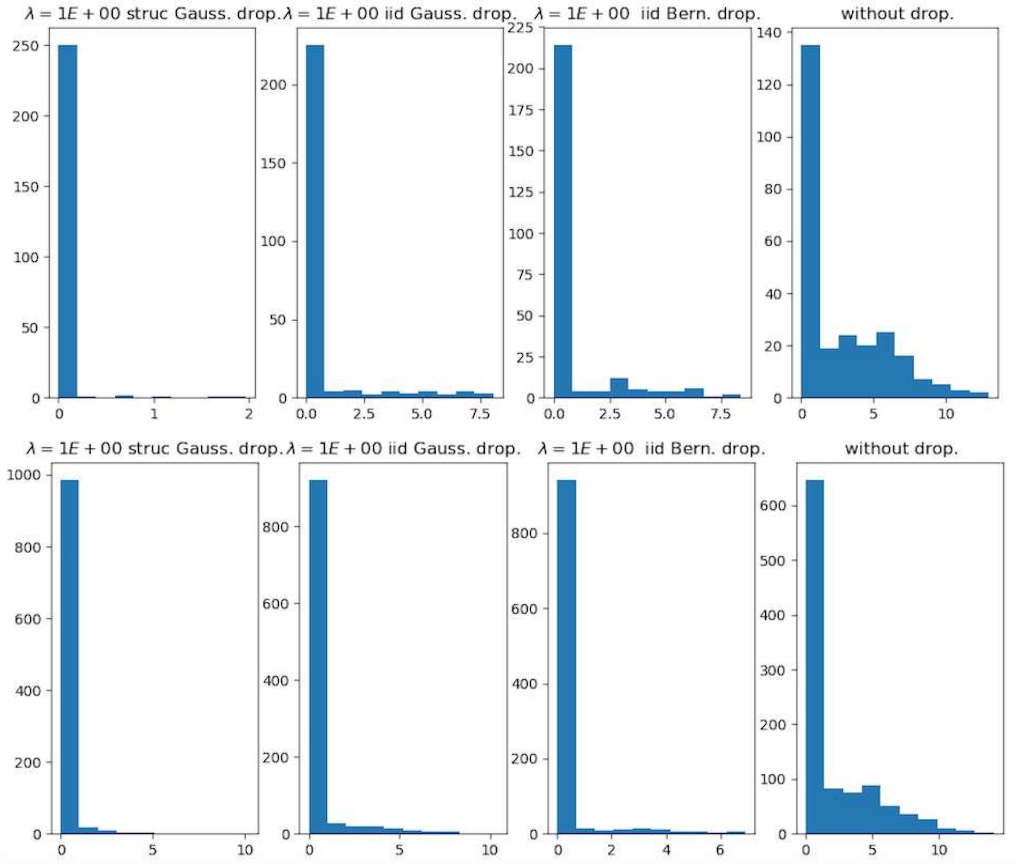



Figure A.4: First layer activations after training our 2 hidden layers Network on MNIST, without dropout, with i.i.d. gaussian dropout, i.i.d. Bernoulli dropout or structured dropout (ASNI). With 256 units (above) and 1024 units (below).

# Bibliography


Abdelmoez, M. N., Iida, K., Oguchi, Y., Nishikii, H., Yokokawa, R., Kotera, H., Uemura, S., Santiago, J. G., and Shintaku, H. (2018). Sinc-seq: correlation of transient gene expressions between nucleus and cytoplasm reflects single-cell physiology. *Genome biology*, 19(1):66. 96

Abu-Mostafa, Y. S. (1989). The vapnik-chervonenkis dimension: Information versus complexity in learning. *Neural Computation*, 1(3):312–317. 16, 22

Achille, A. and Soatto, S. (2018a). Emergence of invariance and disentanglement in deep representations. *Journal of Machine Learning Research*, 19(50). 54

Achille, A. and Soatto, S. (2018b). Information dropout: Learning optimal representations through noisy computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 54

Alm, C. O., Roth, D., and Sproat, R. (2005). Emotions from text: machine learning for text-based emotion prediction. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 579–586. Association for Computational Linguistics. 119

Alspector, J., Allen, R. B., Hu, V., and Satyanarayana, S. (1988). Stochastic learning networks and their electronic implementation. In *Neural information processing systems*, pages 9–21. 49

An, G. (1996). The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674. 50

Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). Vqa: Visual question answering. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2425–2433. IEEE Computer Society. 3

Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79. 14

Atchadé, Y. F., Fort, G., and Moulines, E. (2017). On perturbed proximal gradient algorithms. *Journal of Machine Learning Research*, 18:1–33. 90

Aydore, S., Thirion, B., and Varoquaux, G. (2019). Feature grouping as a stochastic regularizer for high-dimensional structured data. In *International Conference on Machine Learning*, pages 385–394. 103, 105, 106, 108

Ba, J. and Frey, B. (2013). Adaptive dropout for training deep neural networks. In *Advances in Neural Information Processing Systems*, pages 3084–3092. 65, 103

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2011). Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106. 89

Bach, F., Jenatton, R., Mairal, J., and Obozinski, G. (2012). Optimization with sparsity-inducing penalties. *Foundations and Trends® in Machine Learning*, 4(1):1–106. 7

Bacher, R. and Kendziorski, C. (2016). Design and computational analysis of single-cell RNA-sequencing experiments. *Genome Biology*, 17(1):63. 86

Baldi, P. and Sadowski, P. (2014). The dropout learning algorithm. *Artificial Intelligence*, 210:78–122. 64, 120, 121

Baldi, P. and Sadowski, P. J. (2013). Understanding dropout. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Adv. Neural. Inform. Process Syst.*, volume 26, pages 2814–2822. Curran Associates, Inc. 22, 87, 89, 91, 98, 103, 107

Barlow, H. B. e. (1959). Possible principles underlying the transformations of sensory messages. *Sensory Communication, Contributions: Contributions*, 217. 24, 103

Barsig, J., Küsters, S., Vogt, K., Volk, H.-D., Tiegs, G., and Wendel, A. (1995). Lipopolysaccharide-induced interleukin-10 in mice: role of endogenous tumor necrosis factor-$\alpha$. *European journal of immunology*, 25(10):2888–2893. 97

Barton, A. and Pitzalis, C. (2016). Stratified medicine in rheumatoid arthritis —the matura programme. 30

Baskin, C., Liss, N., Chai, Y., Zheltonozhskii, E., Schwartz, E., Giryes, R., Mendelson, A., and Bronstein, A. M. (2018). NICE: Noise Injection and Clamping Estimation for Neural Network Quantization. *arXiv:1810.00162 [cs]*. arXiv: 1810.00162. 45

Battiti, R. (1992). First and second-order methods for learning: between steepest descent and newton's method. *Neural computation*, 4(2):141–166. 57

Beceren, K., Ohka, M., Jin, T., Miyaoka, T., and Yussof, H. (2012). Optimization of human tactile sensation using stochastic resonance. *Procedia Engineering*, 41:792–797. 45

Ben-Tal, A., El Ghaoui, L., and Nemirovski, A. (2009). *Robust optimization*, volume 28. Princeton University Press. 49

Ben-Tal, A. and Nemirovski, A. (2000). Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical programming*, 88(3):411–424. 49

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: a review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828. 24, 103

Benzi, R., Sutera, A., and Vulpiani, A. (1981). The mechanism of stochastic resonance. *Journal of Physics A: Mathematical and General*, 14(11):L453–L457. 45

Bishop, C. M. (1995a). Training with Noise is Equivalent to Tikhonov Regularization. *Neural Computation*, 7(1):108–116. 50, 55, 58, 64

Bishop, C. M. (1995b). Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116. 107, 120

Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer. ix, 12, 16, 17, 18, 22

Black, J. R. and Clark, S. J. (2016). Age-related macular degeneration: genome-wide association studies to translation. *Genetics in Medicine*, 18(4):283. 30

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. (2017). Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877. 48

Bos, P. D., Zhang, X. H.-F., Nadal, C., Shu, W., Gomis, R. R., Nguyen, D. X., Minn, A. J., van de Vijver, M. J., Gerald, W. L., and Foekens, J. A. (2009). Genes that mediate breast cancer metastasis to the brain. *Nature*, 459(7249):1005. 75

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer. 60

Bouthillier, X., Konda, K., Vincent, P., and Memisevic, R. (2015). Dropout as data augmentation. *arXiv:1506.08700 [cs, stat]*. arXiv: 1506.08700. 22

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press. 17

Breiman, L. (1995). Better subset regression using the nonnegative garrote. *Technometrics*, 37(4):373–384. 19

Breiman, L. (1996a). Bagging predictors. *Mach. Learn.*, 24(2):123–140. 87

Breiman, L. (1996b). Bagging predictors. *Machine Learning*, 24(2):123–140. 118

Breiman, L. (1996c). Bias, variance, and arcing classifiers. Technical report, Statistics Department, University of California, Berkeley. 19

Breiman, L. (2001). Random forests. *Mach. Learn.*, 45(1):5–32. 19, 87, 118

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and autonomous systems*, 6(1-2):3–15. 3

Buchanan, B. G. (2005). A (very) brief history of artificial intelligence. *Ai Magazine*, 26(4):53–53. 2

Burton, P. R., Clayton, D. G., Cardon, L. R., Craddock, N. J., Deloukas, P., Duncanson, A., Kwiatkowski, D. P., McCarthy, M. I., Ouwehand, W. H., and Samani, N. J. (2007). Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature*, 447(7145):661. 30

Casella, G. and George, E. I. (1992). Explaining the gibbs sampler. *The American Statistician*, 46(3):167–174. 48

Chakraborty, S. (2005). *Bayesian Machine Learning*. PhD thesis, University of Florida, Gainesville, FL, USA. AAI3188070. 48

Chen, C. and Huang, J. (2012). Compressive sensing mri with wavelet tree sparsity. In *Advances in neural information processing systems*, pages 1115–1123. 22

Chen, G., Jaradat, S. A., Banerjee, N., Tanaka, T. S., Ko, M. S., and Zhang, M. Q. (2002). Evaluation and comparison of clustering algorithms in analyzing es cell gene expression data. *Statistica Sinica*, pages 241–262. 112

Chen, G., Schell, J. P., Benitez, J. A., Petropoulos, S., Yilmaz, M., Reinius, B., Alekseenko, Z., Shi, L., Hedlund, E., Lanner, F., Sandberg, R., and Deng, Q. (2016). Single-cell analyses of X Chromosome inactivation dynamics and pluripotency during differentiation. *Genome Res.*, 26(10):1342–1354. 85

Chen, N., Zhu, J., Chen, J., and Chen, T. (2015). Dropout training for svms with data augmentation. *Frontiers of Computer Science*, pages 1–20. 65

Chen, S. and Donoho, D. (1994). Basis pursuit. In *Proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers*, volume 1, pages 41–44. IEEE. 19

Chen, X. and Lawrence Zitnick, C. (2015). Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2422–2431. 3

Chiu, C.-t., Mehrotra, K., Mohan, C. K., and Ranka, S. (1994). Training Techniques to Obtain Fault Tolerant Neural Networks. In *In FTCS-24: 24th international*, pages 360–369. IEEE Computer Society Press. 50

Choy, E., Yelensky, R., Bonakdar, S., Plenge, R. M., Saxena, R., De Jager, P. L., Shaw, S. Y., Wolfish, C. S., Slavik, J. M., and Cotsapas, C. (2008). Genetic analysis of human traits in vitro: drug response and gene expression in lymphoblastoid cell lines. *PLoS genetics*, 4(11):e1000287. 33

Chuang, H.-Y., Lee, E., Liu, Y.-T., Lee, D., and Ideker, T. (2007). Network-based classification of breast cancer metastasis. *Molecular systems biology*, 3(1):140. 75

Coates, A., Ng, A., and Lee, H. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. 73

Cogswell, M., Ahmed, F., Girshick, R., Zitnick, L., and Batra, D. (2015). Reducing overfitting in deep networks by decorrelating representations. *arXiv preprint arXiv:1511.06068*. 25, 103

Consortium, I. H. G. S. et al. (2001). Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860. 4

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. (2018). Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*. 22

Cui, J., Stahl, E. A., Saevarsdottir, S., Miceli, C., Diogo, D., Trynka, G., Raj, T., Mirkov, M. U., Canhao, H., and Ikari, K. (2013). Genome-wide association study and gene expression analysis identifies cd84 as a predictor of response to etanercept therapy in rheumatoid arthritis. *PLoS genetics*, 9(3):e1003394. 31

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314. 9

Davenport, W. B. and Root, W. L. (1958). *An introduction to the theory of random signals and noise*, volume 159. McGraw-Hill New York. 45

Decoste, D. and Schölkopf, B. (2002). Training invariant support vector machines. *Machine learning*, 46(1-3):161–190. 50

Deng, Q., Ramsköld, D., Reinius, B., and Sandberg, R. (2014). Single-cell RNA-seq reveals dynamic, random monoallelic gene expression in mammalian cells. *Science*, 343(6167):193–6. 85

Denil, M., Shakibi, B., Dinh, L., Ranzato, M., and de Freitas, N. (2013). Predicting parameters in deep learning. In *Advances in neural information processing systems*, pages 2148–2156. 25

Dennis Jr, J. E. and Schnabel, R. B. (1996). *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16. Siam. 7

Desjardins, G., Simonyan, K., Pascanu, R., and Kavukcuoglu, K. (2015). Natural neural networks. In *Advances in Neural Information Processing Systems*, pages 2071–2079. 25, 103

DeVries, T. and Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*. 22, 65, 103, 105

Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer. 19, 25, 49, 103

Dietterich, T. G. (2002). Ensemble learning. *The Handbook of Brain Theory and Neural Networks*. 49

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*. 46

Donoho, D. L. (2000). High-dimensional data analysis: The curses and blessings of dimensionality. *AMS math challenges lecture*, 1(32):375. 117

Edwards, P. J. and Murray, A. F. (1996). Modelling weight- and input-noise in MLP learning. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 78–83 vol.1. 50

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *The Annals of statistics*, 32(2):407–499. 19

Elisseeff, A., Evgeniou, T., and Pontil, M. (2005). Stability of randomized learning algorithms. *Journal of Machine Learning Research*, 6(Jan):55–79. 65

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York. ix, 13

Friedman, J. H. and Hall, P. (2007). On bagging and nonlinear estimation. *Journal of statistical planning and inference*, 137(3):669–683. 19

Gal, Y. and Ghahramani, Z. (2016a). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. 48, 64

Gal, Y. and Ghahramani, Z. (2016b). A theoretically grounded application of dropout in recurrent neural networks. In *Advances in neural information processing systems*, pages 1019–1027. 48, 65, 103

Gallicchio, C., Martin-Guerrero, J., Micheli, A., and Soria-Olivas, E. (2017). Randomized machine learning approaches: Recent developments and challenges. In *Proceedings of the 25th European Symposium on Artificial Neural Networks (ESANN)*, pages 77–86. i6doc. com. 46, 117

Gao, W. and Zhou, Z. (2015). Dropout rademacher complexity of deep neural networks. *Sci China Inf Sci*. 64

Gardner, E., Wallace, D., and Stroud, N. (1989). Training with noise and the storage of correlated patterns in a neural network model. *Journal of Physics A: Mathematical and General*, 22(12):2019. 50

Geng, Q. and Viswanath, P. (2016). The optimal noise-adding mechanism in differential privacy. *IEEE Transactions on Information Theory*, 62(2):925–951. 46

Gentle, J. E. (2009). *Computational statistics*, volume 308. Springer. 106

Gibofsky, A. (2012). Overview of epidemiology, pathophysiology, and diagnosis of rheumatoid arthritis. *The American journal of managed care*, 18(13 Suppl):S295–302. 29

Girosi, F., Jones, M., and Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural computation*, 7(2):219–269. 16

Globerson, A. and Roweis, S. (2006). Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on Machine learning*, pages 353–360. ACM. 54

Goh, J. and Sim, M. (2010). Distributionally robust optimization and its tractable approximations. *Operations research*, 58(4-part-1):902–917. 49

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680. 21

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. *arXiv preprint arXiv:1302.4389*. 65

Gorissen, B. L., Yanıkoğlu, I., and Hertog, D. d. (2015). A Practical Guide to Robust Optimization. *Omega*, 53:124–137. arXiv: 1501.02634. 49

Graham, B. (2014). Fractional max-pooling. *arXiv preprint arXiv:1412.6071*. 21

Grandvalet, Y., Canu, S., and Boucheron, S. (1997). Noise injection: Theoretical prospects. *Neural Computation*, 9(5):1093–1108. 59

Graves, A. (2011). Practical variational inference for neural networks. In *Advances in neural information processing systems*, pages 2348–2356. 48

Grün, D., Lyubimova, A., Kester, L., Wiebrands, K., Basak, O., Sasaki, N., Clevers, H., and Oudenaarden, A. v. (2015). Single-cell messenger RNA sequencing reveals rare intestinal cell types. *Nature*, 525(7568):251–255. 5

Guyon, I., Li, J., Mader, T., Pletscher, P. A., Schneider, G., and Uhr, M. (2007). Competitive baseline methods set new standards for the nips 2003 feature selection benchmark. *Pattern recognition letters*, 28(12):1438–1444. 70, 109

Hansen, B. E. (2016). The risk of james–stein and lasso shrinkage. *Econometric Reviews*, 35(8-10):1456–1470. 19

Hardt, M., Recht, B., and Singer, Y. (2016). Train faster, generalize better: Stability of stochastic gradient descent. In *International Conference on Machine Learning*, pages 1225–1234. 54, 65

Harmer, G. P., Davis, B. R., and Abbott, D. (2002). A review of stochastic resonance: circuits and measurement. *IEEE Transactions on Instrumentation and Measurement*, 51(2):299–309. 45

Hassibi, B. and Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, pages 164–171. 25, 57, 103

Haury, A.-C. and Vert, J.-P. (2010). On the stability and interpretability of prognosis signatures in breast cancer. In *Proceedings of the Fourth International Workshop on Machine Learning in Systems Biology (MLSB10)*. To appear. 13, 75

Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR. 55

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034. 117

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. 21

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. (1998). Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28. 49

Helmbold, D. P. and Long, P. M. (2015). On the inductive bias of dropout. *The Journal of Machine Learning Research*, 16(1):3403–3454. 58, 64, 78, 81, 120

Helmbold, D. P. and Long, P. M. (2017). Surprising properties of dropout in deep networks. *The Journal of Machine Learning Research*, 18(1):7284–7311. 64, 81, 103, 120

Hidaka, I., Nozaki, D., and Yamamoto, Y. (2000). Functional stochastic resonance in the human brain: noise induced sensitization of baroreflex system. *Physical review letters*, 85(17):3740. 45

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., et al. (2012a). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29. 3

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012b). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580 [cs]*. arXiv: 1207.0580. 61, 63, 66, 103, 117

Hoerl, A. E. and Kennard, R. W. (1970). Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67. 17, 18, 88

Hogeweg, P. (2011). The roots of bioinformatics in theoretical biology. *PLoS computational biology*, 7(3). 4

Holland, P. W. and Welsch, R. E. (1977). Robust regression using iteratively reweighted least-squares. *Communications in Statistics-theory and Methods*, 6(9):813–827. 16

Holmstrom, L. and Koistinen, P. (1992). Using additive noise in back-propagation training. *IEEE Transactions on Neural Networks*, 3(1):24–38. 50

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558. 2

Huang, D. W., Sherman, B. T., and Lempicki, R. A. (2009). Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists. *Nucl. Acids Res.*, 37:1–13. 97

Huang, G.-B., Wang, D. H., and Lan, Y. (2011a). Extreme learning machines: a survey. *International journal of machine learning and cybernetics*, 2(2):107–122. 46

Huang, J., Zhang, T., and Metaxas, D. (2011b). Learning with structured sparsity. *Journal of Machine Learning Research*, 12(Nov):3371–3412. 22

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer. 16

Hunkapiller, T., Kaiser, R., Koop, B., and Hood, L. (1991). Large-scale and automated dna sequence determination. *Science*, 254(5028):59–67. 4

Hutter, M. and Poland, J. (2004). Prediction with expert advice by following the perturbed leader for general weights. In *International Conference on Algorithmic Learning Theory*, pages 279–293. Springer. 46

Hyvärinen, A. (2013). Independent component analysis: recent advances. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110534. 25, 103

Iannelli, J., Yariv, A., Chen, T., and Zhuang, Y. (1994). Stochastic resonance in a semiconductor distributed feedback laser. *Applied physics letters*, 65(16):1983–1985. 45

Im, D. J., Ahn, S., Memisevic, R., and Bengio, Y. (2015). Denoising Criterion for Variational Auto-Encoding Framework. *arXiv:1511.06406 [cs]*. arXiv: 1511.06406. 46

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR.org. 22, 65, 106, 111

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2017). Quantization and training of neural networks for efficient integer-arithmetic-only inference. *arXiv preprint arXiv:1712.05877*. 45

Jacob, L., Obozinski, G., and Vert, J.-P. (2009). Group lasso with overlap and graph lasso. In *Proceedings of the 26th annual international conference on machine learning*, pages 433–440. ACM. 22, 99

Jaitin, D. A., Weiner, A., Yofe, I., Lara-Astiaso, D., Keren-Shaul, H., David, E., Salame, T. M., Tanay, A., van Oudenaarden, A., and Amit, I. (2016). Dissecting Immune Circuits by Linking CRISPR-Pooled Screens with Single-Cell RNA-Seq. *Cell*, 167(7):1883–1896.e15. 85

Jaitly, N. and Hinton, G. E. (2013). Vocal tract length perturbation (vtlp) improves speech recognition. In *Proc. ICML Workshop on Deep Learning for Audio, Speech and Language*, volume 117. 21

Jia, K., Chan, T.-H., and Ma, Y. (2012). Robust and practical face recognition via structured sparsity. In *European conference on computer vision*, pages 331–344. Springer. 22

Johnston, C., Finkelstein, J., Gelein, R., and Oberdörster, G. (1998). Pulmonary cytokine and chemokine mrna levels after inhalation of lipopolysaccharide in c57bl/6 mice. *Toxicological Sciences*, 46(2):300–307. 97

Jordan, M. I. and Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260. 46

Kang, G., Li, J., and Tao, D. (2018). Shakeout: A new approach to regularized deep neural network training. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1245–1258. 65

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer. 2

Kharchenko, P. V., Silberstein, L., and Scadden, D. T. (2014). Bayesian approach to single-cell differential expression analysis. *Nat. Methods*, 11(7):740–742. 24, 86, 99

Kieffer, J. (1994). Elements of Information Theory (Thomas M. Cover and Joy A. Thomas). *SIAM Review*, 36(3):509–511. 45

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 8

Kingma, D. P., Salimans, T., and Welling, M. (2015). Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583. 65

Klein, R. J., Zeiss, C., Chew, E. Y., Tsai, J.-Y., Sackler, R. S., Haynes, C., Henning, A. K., SanGiovanni, J. P., Mane, S. M., and Mayne, S. T. (2005). Complement factor h polymorphism in age-related macular degeneration. *Science*, 308(5720):385–389. 30

Kolodziejczyk, A. A., Kim, J. K., Svensson, V., Marioni, J. C., and Teichmann, S. A. (2015). The technology and biology of single-cell RNA sequencing. *Molecular Cell*, 58(4):610–620. 85

Konda, K., Bouthillier, X., Memisevic, R., and Vincent, P. (2015). Dropout as data augmentation. *stat*, 1050:29. 53

Kopydlowski, K. M., Salkowski, C. A., Cody, M. J., van Rooijen, N., Major, J., Hamilton, T. A., and Vogel, S. N. (1999). Regulation of macrophage chemokine expression by lipopolysaccharide in vitro and in vivo. *The Journal of Immunology*, 163(3):1537–1544. 97

Kosko, B. (2006). *Noise*. Penguin. 45

Kosko, B., Lee, I., Mitaim, S., Patel, A., and Wilde, M. M. (2009). Applications of forbidden interval theorems in stochastic resonance. In *Applications of Nonlinear Dynamics*, pages 71–89. Springer. 45

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, Citeseer. 73

Krizhevsky, A., Sutskever, I., and Hinton, G. (2014). Imagenet classification with deep convolutional neural. In *Neural Information Processing Systems*, pages 1–9. 11, 21

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Adv. Neural. Inform. Process Syst.*, volume 25, pages 1097–1105. Curran Associates, Inc. 3, 87

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90. 24, 103

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957. 17, 18

Kukačka, J., Golkov, V., and Cremers, D. (2017). Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*. 16, 22

Kuncheva, L. I. (2004). *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons. 19, 20

Kuncheva, L. I. and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2):181–207. 25, 49, 103

Lauer, F. and Bloch, G. (2008). Incorporating prior knowledge in support vector machines for classification: A review. *Neurocomputing*, 71(7-9):1578–1594. 22

Lavi, O., Dror, G., and Shamir, R. (2012). Network-induced classification kernels for gene expression profile analysis. *Journal of Computational Biology*, 19(6):694–709. 22, 122

Lazer, D., Kennedy, R., King, G., and Vespignani, A. (2014). The parable of google flu: traps in big data analysis. *Science*, 343(6176):1203–1205. 14

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. 113

LeCun, Y., Denker, J. S., and Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605. 25, 103

LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., and Sackinger, E. (1995). Comparison of learning algorithms for handwritten digit recognition. In *International conference on artificial neural networks*, volume 60, pages 53–60. Perth, Australia. 11

Lederberg, J. (1987). How dendral was conceived and born. In *Proceedings of ACM conference on history of medical informatics*, pages 5–19. ACM. 2

Lee, S.-I., Lee, H., Abbeel, P., and Ng, A. Y. (2006). Efficient l˜ 1 regularized logistic regression. In *AAAI*, volume 6, pages 401–408. 19

Leen, T. K. (1995). From Data Distributions to Regularization in Invariant Learning. In Tesauro, G., Touretzky, D. S., and Leen, T. K., editors, *Advances in Neural Information Processing Systems 7*, pages 223–230. MIT Press. 50, 54

Leisch, F., Weingessel, A., and Hornik, K. (1998). On the generation of correlated artificial binary data. *SFB Adaptive Information Systems and Modelling in Economics and Management Science, WU Vienna University of Economics and Business*. 115

Lemberger, P. (2017). On generalization and regularization in deep learning. *arXiv preprint arXiv:1704.01312*. 3, 16

Lemley, J., Bazrafkan, S., and Corcoran, P. (2017). Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869. 22

Leonard, D. S. and Reichl, L. (1994). Stochastic resonance in a chemical reaction. *Physical Review E*, 49(2):1734. 45

Li, H., Horns, F., Wu, B., Xie, Q., Li, J., Li, T., Luginbuhl, D. J., Quake, S. R., and Luo, L. (2017). Classifying Drosophila Olfactory Projection Neuron Subtypes by Single-Cell RNA Sequencing. *Cell*, 171(5):1206–1220.e22. 5

Li, Y. and Liu, F. (2016). Whiteout: Gaussian adaptive noise regularization in feedforward neural networks. *arXiv preprint arXiv:1612.01490*. 65

Liu, J. S. and Wu, Y. N. (1999). Parameter expansion for data augmentation. *Journal of the American Statistical Association*, 94(448):1264–1274. 21

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331. 119

Luo, P. (2017). Learning deep architectures via generalized whitened neural networks. In *International Conference on Machine Learning*, pages 2238–2246. 25, 103

Ma, S. and Huang, J. (2008). Penalized feature selection and classification in bioinformatics. *Briefings in bioinformatics*, 9(5):392–403. 17, 18, 19

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1*, pages 142–150. Association for Computational Linguistics. 72

Maass, W. (2014). Noise as a Resource for Computation and Learning in Networks of Spiking Neurons. *Proceedings of the IEEE*, 102:860–880. 45

Macosko, E. Z., Basu, A., Satija, R., Nemesh, J., Shekhar, K., Goldman, M., Tirosh, I., Bialas, A. R., Kamitaki, N., Martersteck, E. M., Trombetta, J. J., Weitz, D. A., Sanes, J. R., Shalek, A. K., Regev, A., and McCarroll, S. A. (2015). Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell*, 161(5):1202–1214. 85

Maeda, S.-i. (2014). A Bayesian encourages dropout. *arXiv:1412.7003 [cs, stat]*. arXiv: 1412.7003. 48, 64, 65, 103

Mäkinen, H., Kautiainen, H., Hannonen, P., and Sokka, T. (2005). Is das28 an appropriate tool to assess remission in rheumatoid arthritis? *Annals of the rheumatic diseases*, 64(10):1410–1413. 119

Mallat, S. (2012). Group invariant scattering. *Comm. Pure Appl. Math.*, 65(10):1331–1398. 103

Mariet, Z. and Sra, S. (2016). Diversity networks. In *Proceedings of ICLR*. 25, 103

Matsuoka, K. (1992). Noise injection into inputs in back-propagation learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(3):436–440. 50

Maxam, A. M. and Gilbert, W. (1977). A new method for sequencing dna. *Proceedings of the National Academy of Sciences*, 74(2):560–564. 4

McAllester, D. (2013). A pac-bayesian tutorial with a dropout bound. *arXiv preprint arXiv:1307.2118*. 64

McClelland, J. L., Rumelhart, D. E., Group, P. R., et al. (1986). Parallel distributed processing. *Explorations in the Microstructure of Cognition*, 2:216–271. 3

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133. 2

McDermott, J. (1982). R1: A rule-based configurer of computer systems. *Artificial intelligence*, 19(1):39–88. 2

McDonnell, M. D. and Ward, L. M. (2011). The benefits of noise in neural systems: bridging theory and experiment. *Nat. Rev. Neurosci.*, 12(7):415–426. 45

McInnes, I. B. and Schett, G. (2011). The pathogenesis of rheumatoid arthritis. *New England Journal of Medicine*, 365(23):2205–2219. 29

Medvedev, A. E., Kopydlowski, K. M., and Vogel, S. N. (2000). Inhibition of lipopolysaccharide-induced signal transduction in endotoxin-tolerized mouse macrophages: dysregulation of cytokine, chemokine, and toll-like receptor 2 and 4 gene expression. *The Journal of Immunology*, 164(11):5564–5574. 97

Meinshausen, N. and Bühlmann, P. (2010). Stability selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473. 19

Meng, X.-L. and Van Dyk, D. A. (1999). Seeking efficient data augmentation schemes via conditional and marginal augmentation. *Biometrika*, 86(2):301–320. 21

Meyer, R. (1970). Theoretical and computational aspects of nonlinear regression. In *Nonlinear programming*, pages 465–486. Elsevier. 57

Mikołajczyk, A. and Grochowski, M. (2018). Data augmentation for improving deep learning in image classification problem. In *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pages 117–122. IEEE. 21

Min, S., Lee, B., and Yoon, S. (2017). Deep learning in bioinformatics. *Briefings in bioinformatics*, 18(5):851–869. ix, 3

Minn, A. J., Gupta, G. P., Siegel, P. M., Bos, P. D., Shu, W., Giri, D. D., Viale, A., Olshen, A. B., Gerald, W. L., and Massagué, J. (2005). Genes that mediate breast cancer metastasis to lung. *Nature*, 436(7050):518. 75

Minsky, M. and Papert, S. A. (2017). *Perceptrons: An introduction to computational geometry*. MIT press. 9

Mitzenmacher, M. and Upfal, E. (2005). *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge university press. 46

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., and Ostrovski, G. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529. 3, 117

Molchanov, D., Ashukha, A., and Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. *arXiv preprint arXiv:1701.05369*. 65

Moon, T. K. (1996). The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6):47–60. 48

Muandet, K., Fukumizu, K., Dinuzzo, F., and Schölkopf, B. (2012). Learning from distributions via support measure machines. In *Advances in neural information processing systems*, pages 10–18. 48, 120

Muandet, K., Fukumizu, K., Sriperumbudur, B., and Schölkopf, B. (2017). Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends® in Machine Learning*, 10(1-2):1–141. 48

Murray, A. and Edwards, P. (1994). Enhanced MLP performance and fault tolerance resulting from synaptic weight noise during training. *IEEE Transactions on Neural Networks*, 5(5):792–802. 50

Murray, A. F. (1992). Multilayer perceptron learning optimized for on-chip implementation: A noise-robust system. *Neural Computation*, 4(3):366–381. 50

Nasrabadi, N. M. (2007). Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901. 57

Nelder, J. A. and Wedderburn, R. W. (1972). Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384. 35

Newell, A., Shaw, J. C., and Simon, H. A. (1959). Report on a general problem solving program. In *IFIP congress*, volume 256, page 64. Pittsburgh, PA. 2

Neyshabur, B. (2017). Implicit regularization in deep learning. *arXiv preprint arXiv:1709.01953*. 16

Noh, H., You, T., Mun, J., and Han, B. (2017). Regularizing deep neural networks by noise: Its interpretation and optimization. In *Advances in Neural Information Processing Systems*, pages 5109–5118. 120

Norton, M., Takeda, A., and Mafusalov, A. (2017a). Optimistic Robust Optimization With Applications To Machine Learning. *arXiv:1711.07511 [cs, math, stat]*. arXiv: 1711.07511. 49

Norton, M., Takeda, A., and Mafusalov, A. (2017b). Optimistic robust optimization with applications to machine learning. *arXiv preprint arXiv:1711.07511*. 49

Obozinski, G., Jacob, L., and Vert, J.-P. (2011). Group lasso with overlaps: the latent group lasso approach. *arXiv preprint arXiv:1110.0413*. 22

Ogikubo, Y., Norimatsu, M., Sasaki, Y., Yasuda, A., Saegusa, J., and Tamura, Y. (2004). Effect of lipopolysaccharide (lps) injection on the immune responses of lps-sensitive mice. *Journal of veterinary medical science*, 66(10):1189–1193. 97

Oneto, L., Ridella, S., and Anguita, D. (2017). Generalization performances of randomized classifiers and algorithms built on data dependent distributions. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*. 46

Orozco, G., Rueda, B., and Martin, J. (2006). Genetic basis of rheumatoid arthritis. *Biomedicine & Pharmacotherapy*, 60(10):656–662. 31

Osborne, M. R., Presnell, B., and Turlach, B. A. (2000). On the lasso and its dual. *Journal of Computational and Graphical statistics*, 9(2):319–337. 19

Osoba, O., Mitaim, S., and Kosko, B. (2011). Noise benefits in the expectation-maximization algorithm: Nem theorems and models. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 3178–3183. IEEE. 46

Ozsolak, F. and Milos, P. M. (2011). Rna sequencing: advances, challenges and opportunities. *Nature reviews genetics*, 12(2):87. 86

Pappas, D. A., Kremer, J. M., Reed, G., Greenberg, J. D., and Curtis, J. R. (2014). Design characteristics of the corrona certain study: a comparative effectiveness study of biologic agents for rheumatoid arthritis patients. *BMC musculoskeletal disorders*, 15(1):113. 31

Patel, A. P., Tirosh, I., Trombetta, J. J., Shalek, A. K., Gillespie, S. M., Wakimoto, H., Cahill, D. P., Nahed, B. V., Curry, W. T., Martuza, R. L., Louis, D. N., Rozenblatt-Rosen, O., Suvà, M. L., Regev, A., and Bernstein, B. E. (2014). Single-cell RNA-seq highlights intratumoral heterogeneity in primary glioblastoma. *Science*, 344(6190):1396–1401. 85

Paulus, H. E. (2004). Defining remission in rheumatoid arthritis: what is it? does it matter? *The Journal of rheumatology*, 31(1):1–4. 119

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann. 3

Peeling, S., Moore, R., and Tomlinson, M. (1986). The multi-layer perceptron as a tool for speech pattern processing research. In *Proceedings of the 10th Autumn Conference on Speech and Hearing*. 50

Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1226–1238. 25, 103

Perou, C. M., Sørlie, T., Eisen, M. B., van de Rijn, M., Jeffrey, S. S., Rees, C. A., Pollack, J. R., Ross, D. T., Johnsen, H., Akslen, L. A., Fluge, O., Pergamenschikov, A., Williams, C., Zhu, S. X., Lønning, P. E., Børresen-Dale, A. L., Brown, P. O., and Botstein, D. (2000). Molecular portraits of human breast tumours. *Nature*, 406(6797):747–752. 86

Pierson, E. and Yau, C. (2015). ZIFA: Dimensionality reduction for zero-inflated single-cell gene expression analysis. *Genome Biology*, 16(1):241. 86

Plaut, D. C., Nowlan, S. J., and Hinton, G. E. (1986). Experiments on learning by back propagation. *Citeseer*. 49, 50

Preisser, J. S. and Qaqish, B. F. (2014). A comparison of methods for simulating correlated binary variables with specified marginal means and correlations. *Journal of Statistical Computation and Simulation*, 84(11):2441–2452. 115

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106. 46

Raginsky, M., Rakhlin, A., and Telgarsky, M. (2017). Non-convex learning via stochastic gradient langevin dynamics: a nonasymptotic analysis. *arXiv preprint arXiv:1702.03849*. 50

Ramaswamy, S., Tamayo, P., Rifkin, R., Mukherjee, S., Yeang, C.-H., Angelo, M., Ladd, C., Reich, M., Latulippe, E., and Mesirov, J. P. (2001). Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences*, 98(26):15149–15154. 85

Rapaport, F., Zinovyev, A., Dutreix, M., Barillot, E., and Vert, J.-P. (2007). Classification of microarray data using gene networks. *BMC bioinformatics*, 8(1):35. 99, 121, 122

Raser, J. M. and O'shea, E. K. (2005). Noise in gene expression: origins, consequences, and control. *Science*, 309(5743):2010–2013. 45

Ratner, A. J., Ehrenberg, H., Hussain, Z., Dunnmon, J., and Ré, C. (2017). Learning to compose domain-specific transformations for data augmentation. In *Advances in neural information processing systems*, pages 3236–3246. 22

Raviv, Y. and Intrator, N. (1996). Bootstrapping with noise: An effective regularization technique. *Connection Science*, 8(3-4):355–372. 49

Reed, R., Marks, R., and Oh, S. (1995). Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter. *IEEE Transactions on Neural Networks*, 6(3):529–538. 58

Reuter, J. A., Spacek, D. V., and Snyder, M. P. (2015). High-throughput sequencing technologies. *Molecular cell*, 58(4):586–597. 117

Richard, M. D. and Lippmann, R. P. (1991). Neural network classifiers estimate bayesian a posteriori probabilities. *Neural computation*, 3(4):461–483. 55

Rifai, S., Glorot, X., Bengio, Y., and Vincent, P. (2011). Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*. 50

Risso, D., Perraudeau, F., Gribkova, S., Dudoit, S., and Vert, J.-P. (2018). A general and flexible method for signal extraction from single-cell rna-seq data. *Nature communications*, 9(1):284. 86, 99

Robbins, H. and Siegmund, D. (1971). A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing methods in statistics*, pages 233–257. Elsevier. 8, 60, 62, 90

Rodríguez, P., Gonzàlez, J., Cucurull, G., Gonfaus, J. M., and Roca, X. (2016). Regularizing cnns with locally constrained decorrelations. *arXiv preprint arXiv:1611.01967*. 103

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65. 112

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*. 8, 57

Rudi, A. and Rosasco, L. (2017). Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*, pages 3215–3225. 65

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*. 2, 7, 10

Ruschhaupt, M., Huber, W., Poustka, A., and Mansmann, U. (2004). A compendium to ensure computational reproducibility in high-dimensional classification tasks. *Statistical Applications in Genetics and Molecular Biology*, 3(1):1–24. 14

Russell, S. J. and Norvig, P. (2016). Artificial intelligence: A modern approach. 2, 3

Sajjadi, M., Javanmardi, M., and Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 1163–1171. 81

Salazar-Roa, M. and Malumbres, M. (2017). Fueling the cell division cycle. *Trends in cell biology*, 27(1):69–81. 97

Samacoits, A., Chouaib, R., Safieddine, A., Traboulsi, A.-M., Ouyang, W., Zimmer, C., Peter, M., Bertrand, E., Walter, T., and Mueller, F. (2018). A computational framework to study sub-cellular rna localization. *Nature communications*, 9(1):4584. 5

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:210–229. 2

Sanger, F. and Coulson, A. R. (1975). A rapid method for determining sequences in dna by primed synthesis with dna polymerase. *Journal of molecular biology*, 94(3):441–448. 4

Scardapane, S. and Wang, D. (2017). Randomness in neural networks: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(2):e1200. 65

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117. 3, 11

Schölkopf, B., Burges, C., and Vapnik, V. (1996). Incorporating invariances in support vector learning machines. In *International Conference on Artificial Neural Networks*, pages 47–52. Springer. 21, 87

Schuchman, L. (1964). Dither Signals and Their Effect on Quantization Noise. *IEEE Transactions on Communication Technology*, 12(4):162–165. 45

Scornet, E., Biau, G., and Vert, J.-P. (2015). Consistency of random forests. *The Annals of Statistics*, 43(4):1716–1741. 118

Segerstolpe, Å., Palasantza, A., Eliasson, P., Andersson, E.-M., Andréasson, A.-C., Sun, X., Picelli, S., Sabirsh, A., Clausen, M., Bjursell, M. K., Smith, D. M., Kasper, M., Ämmälä, C., and Sandberg, R. (2016). Single-Cell Transcriptome Profiling of Human Pancreatic Islets in Health and Type 2 Diabetes. *Cell Metab.*, 24(4):593–607. 5

Shaham, U., Yamada, Y., and Negahban, S. (2015). Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*. 49

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2016). Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175. 48

Sieberts, S. K., Zhu, F., García-García, J., Stahl, E., Pratap, A., Pandey, G., Pappas, D., Aguilar, D., Anton, B., and Bonet, J. (2016). Crowdsourced assessment of common genetic contribution to predicting anti-tnf treatment response in rheumatoid arthritis. *Nature communications*, 7:12460. ix, 23, 28, 37, 38, 39, 40, 118, 119

Sietsma, J. and Dow, R. J. (1988). Neural net pruning-why and how. In *IEEE international conference on neural networks*, volume 1, pages 325–333. IEEE San Diego. 50

Sietsma, J. and Dow, R. J. (1991). Creating artificial neural networks that generalize. *Neural networks*, 4(1):67–79. 50

Simard, P., Victorri, B., LeCun, Y., and Denker, J. (1992). Tangent prop-a formalism for specifying selected invariances in an adaptive network. In *Advances in neural information processing systems*, pages 895–903. 21

Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best practices for convolutional neural networks applied to visual document analysis. In *null*, page 958. IEEE. 53

Smith, L. M., Sanders, J. Z., Kaiser, R. J., Hughes, P., Dodd, C., Connell, C. R., Heiner, C., Kent, S. B., and Hood, L. E. (1986). Fluorescence detection in automated dna sequence analysis. *Nature*, 321(6071):674. 4

Soneson, C. and Robinson, M. D. (2017). Bias, robustness and scalability in differential expression analysis of single-cell RNA-seq data. Technical Report 143289, bioRxiv. 94, 95

Sørlie, T., Perou, C. M., Tibshirani, R., Aas, T., Geisler, S., Johnsen, H., Hastie, T., Eisen, M. B., Van De Rijn, M., and Jeffrey, S. S. (2001). Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications. *Proceedings of the National Academy of Sciences*, 98(19):10869–10874. 85

Soyster, A. L. (1973). Technical Note—Convex Programming with Set-Inclusive Constraints and Applications to Inexact Linear Programming. *Operations Research*, 21(5):1154–1157. 48

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. In *ICLR (workshop track)*. 21

Sra, S., Nowozin, S., and Wright, S. J. (2012). *Optimization for machine learning*. Mit Press. 7, 22

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958. x, 22, 50, 52, 58, 61, 62, 63, 73, 78, 87, 89, 90, 103, 104, 113

Stahl, E. A., Raychaudhuri, S., Remmers, E. F., Xie, G., Eyre, S., Thomson, B. P., Li, Y., Kurreeman, F. A., Zhernakova, A., and Hinks, A. (2010). Genome-wide association study meta-analysis identifies seven new rheumatoid arthritis risk loci. *Nature genetics*, 42(6):508. 31

Sung, K. K. and Poggio, T. (1994). Example based learning for view-based human face detection. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE ARTIFICIAL INTELLIGENCE LAB. 21

Szabó, Z., Sriperumbudur, B. K., Póczos, B., and Gretton, A. (2016). Learning theory for distribution regression. *The Journal of Machine Learning Research*, 17(1):5272–5311. 48

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*. 54

Tak, P. P. (2011). A personalized medicine approach to biologic treatment of rheumatoid arthritis: a preliminary treatment algorithm. *Rheumatology*, 51(4):600–609. 30

Tang, F., Barbacioru, C., Wang, Y., Nordman, E., Lee, C., Xu, N., Wang, X., Bodeau, J., Tuch, B. B., Siddiqui, A., Lao, K., and Surani, M. A. (2009). mRNA-Seq whole-transcriptome analysis of a single cell. *Nature Methods*, 6(5):377–382. 85

Tasic, B., Menon, V., Nguyen, T. N., Kim, T. K., Jarsky, T., Yao, Z., Levi, B., Gray, L. T., Sorensen, S. A., Dolbeare, T., Bertagnolli, D., Goldy, J., Shapovalova, N., Parry, S., Lee, C., Smith, K., Bernard, A., Madisen, L., Sunkin, S. M., Hawrylycz, M., Koch, C., and Zeng, H. (2016). Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nat. Neurosci.*, 19(2):335–346. 85

Thiel, C. (2008). Classification on soft labels is robust against label noise. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 65–73. Springer. 52

Thom, H. C. (1958). A note on the gamma distribution. *Monthly Weather Review*, 86(4):117–122. 35

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288. 19, 88, 89

Tibshirani, R., Wainwright, M., and Hastie, T. (2015). *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC. 19

Tishby, N. and Zaslavsky, N. (2015). Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE. 24, 103

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2014). Efficient Object Localization Using Convolutional Networks. *arXiv:1411.4280 [cs]*. arXiv: 1411.4280. 103, 105

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 648–656. 65

Tsimring, L. S. (2014). Noise in Biology. *Rep Prog Phys*, 77(2):026601. 45

Tu, Y., Stolovitzky, G., and Klein, U. (2002). Quantitative noise analysis for gene expression microarray experiments. *Proceedings of the National Academy of Sciences*, 99(22):14031–14036. 75

Turing, A. M. (1950). Can a machine think. *Mind*, 59(236):433–460. 2

Uhlich, S., Porcu, M., Giron, F., Enenkl, M., Kemp, T., Takahashi, N., and Mitsufuji, Y. (2017). Improving music source separation based on deep neural networks through data augmentation and network blending. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 261–265. IEEE. 21

Van De Vijver, M. J., He, Y. D., Van't Veer, L. J., Dai, H., Hart, A. A., Voskuil, D. W., Schreiber, G. J., Peterse, J. L., Roberts, C., and Marton, M. J. (2002). A gene-expression signature as a predictor of survival in breast cancer. *New England Journal of Medicine*, 347(25):1999–2009. 75, I

van der Maaten, L., Chen, M., Tyree, S., and Weinberger, K. Q. (2013). Learning with marginalized corrupted features. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*, number 28 in JMLR Proceedings, pages 410–418. JMLR.org. 54, 56, 58, 64, 72, 73, 87, 89, 120

Van Dyk, D. A. and Meng, X.-L. (2001). The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50. 21, 53

van Rooden, S. M., Heiser, W. J., Kok, J. N., Verbaan, D., van Hilten, J. J., and Marinus, J. (2010). The identification of parkinson's disease subtypes using cluster analysis: a systematic review. *Movement disorders*, 25(8):969–978. 5

Vemuri, G. and Roy, R. (1989). Stochastic resonance in a bistable ring laser. *Physical Review A*, 39(9):4668. 45

Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., Smith, H. O., Yandell, M., Evans, C. A., Holt, R. A., et al. (2001). The sequence of the human genome. *science*, 291(5507):1304–1351. 4

Vera, M., Piantanida, P., and Vega, L. R. (2018). The role of information complexity and randomization in representation learning. *arXiv preprint arXiv:1802.05355*. 46

Villani, A.-C., Satija, R., Reynolds, G., Sarkizova, S., Shekhar, K., Fletcher, J., Griesbeck, M., Butler, A., Zheng, S., and Lazo, S. (2017). Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. *Science*, 356(6335):eaah4573. 85

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM. 54

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of machine learning research*, 11(Dec):3371–3408. 46, 54, 117

Vinod, H. D. (1978). A survey of ridge regression and related techniques for improvements over ordinary least squares. *The Review of Economics and Statistics*, pages 121–131. 18

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164. 3

Visscher, P. M., Brown, M. A., McCarthy, M. I., and Yang, J. (2012). Five years of gwas discovery. *The American Journal of Human Genetics*, 90(1):7–24. 30

Visscher, P. M., Wray, N. R., Zhang, Q., Sklar, P., McCarthy, M. I., Brown, M. A., and Yang, J. (2017). 10 years of gwas discovery: biology, function, and translation. *The American Journal of Human Genetics*, 101(1):5–22. 30

Wager, S., Fithian, W., and Liang, P. (2016). Data augmentation via lévy processes. *Perturbations, Optimization, and Statistics*, page 343. 64

Wager, S., Fithian, W., Wang, S., and Liang, P. S. (2014). Altitude training: Strong bounds for single-layer dropout. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Adv. Neural. Inform. Process Syst.*, pages 100–108. Curran Associates, Inc. 64, 87

Wager, S., Wang, S., and Liang, P. S. (2013). Dropout training as adaptive regularization. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Adv. Neural. Inform. Process Syst.*, volume 26, pages 351–359. Curran Associates, Inc. 22, 56, 58, 64, 68, 69, 72, 87, 89, 91, 92, 98, 99, 103, 107, 118, 120

Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, pages 1058–1066. 50, 65

Wang, J. and Perez, L. (2017). The effectiveness of data augmentation in image classification using deep learning. *Convolutional Neural Networks Vis. Recognit.* 21

Wang, S. and Manning, C. (2013). Fast dropout training. In Dasgupta, S. and Mcallester, D., editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 118–126. JMLR Workshop and Conference Proceedings. 80

Wang, Y., Klijn, J. G., Zhang, Y., Sieuwerts, A. M., Look, M. P., Yang, F., Talantov, D., Timmermans, M., Meijer-van Gelder, M. E., and Yu, J. (2005). Gene-expression profiles to predict distant metastasis of lymph-node-negative primary breast cancer. *The Lancet*, 365(9460):671–679. 75, I

Wannamaker, n., Lipshitz, n., and Vanderkooy, n. (2000). Stochastic resonance as dithering. *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics*, 61(1):233–236. 45

Ward, L. D. and Kellis, M. (2011). Haploreg: a resource for exploring chromatin states, conservation, and regulatory motif alterations within sets of genetically linked variants. *Nucleic acids research*, 40(D1):D930–D934. 34

Weizenbaum, J. (1966). Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45. 2

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688. 50

Welter, D., MacArthur, J., Morales, J., Burdett, T., Hall, P., Junkins, H., Klemm, A., Flicek, P., Manolio, T., and Hindorff, L. (2013). The nhgri gwas catalog, a curated resource of snp-trait associations. *Nucleic acids research*, 42(D1):D1001–D1006. 30

Westra, H.-J., Peters, M. J., Esko, T., Yaghootkar, H., Schurmann, C., Kettunen, J., Christiansen, M. W., Fairfax, B. P., Schramm, K., and Powell, J. E. (2013). Systematic identification of trans eqtls as putative drivers of known disease associations. *Nature genetics*, 45(10):1238. 34

Whirl-Carrillo, M., McDonagh, E. M., Hebert, J., Gong, L., Sangkuhl, K., Thorn, C., Altman, R. B., and Klein, T. E. (2012). Pharmacogenomics knowledge for personalized medicine. *Clinical Pharmacology & Therapeutics*, 92(4):414–417. 34

Wijbrandts, C. A., Dijkgraaf, M. G., Kraan, M. C., Vinkenoog, M., Smeets, T. J., Dinant, H., Vos, K., Lems, W. F., Wolbink, G. J., and Sijpkens, D. (2008). The clinical response to infliximab in rheumatoid arthritis is in part dependent on pretreatment tumour necrosis factor $\alpha$ expression in the synovium. *Annals of the rheumatic diseases*, 67(8):1139–1144. 29

Witten, I. H., Frank, E., Hall, M. A., and Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann. 53

Wolpert, D. H. (2002). The supervised learning no-free-lunch theorems. In *Soft computing and industry*, pages 25–42. Springer. 16

Wolpert, D. H. and Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82. 16

Woodruff, D. P. (2014). Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157. 46

Wray, N. R., Goddard, M. E., and Visscher, P. M. (2007). Prediction of individual genetic risk to disease from genome-wide association studies. *Genome research*, 17(10):1520–1528. 118

Xu, H., Caramanis, C., and Mannor, S. (2009a). Robust regression and lasso. In *Advances in Neural Information Processing Systems*, pages 1801–1808. 49

Xu, H., Caramanis, C., and Mannor, S. (2009b). Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10(Jul):1485–1510. 49

Xu, H. and Mannor, S. (2012). Robustness and generalization. *Machine learning*, 86(3):391–423. 49

Xu, Y., Jia, R., Mou, L., Li, G., Chen, Y., Lu, Y., and Jin, Z. (2016). Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651*. 21

Xue, Z., Huang, K., Cai, C., Cai, L., Jiang, C.-y., Feng, Y., Liu, Z., Zeng, Q., Cheng, L., Sun, Y. E., Liu, J.-y., Horvath, S., and Fan, G. (2013). Genetic programs in human and mouse early embryos revealed by single-cell RNA sequencing. *Nature*, 500(7464):593–597. 85

Yuan, H., Paskov, I., Paskov, H., González, A. J., and Leslie, C. S. (2016). Multitask learning improves prediction of cancer drug sensitivity. *Scientific reports*, 6:31619. 119

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67. 22

Zeisel, A., Manchado, A. B. M., Codeluppi, S., Lonnerberg, P., La Manno, G., Jureus, A., Marques, S., Munguba, H., He, L., Betsholtz, C., Rolny, C., Castelo-Branco, G., Hjerling-Leffler, J., and Linnarsson, S. (2015). Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, 347(6226):1138–42. 85

Zhai, K. and Wang, H. (2018). Adaptive dropout with rademacher complexity regularization. submitted to ICLR 2019. 65

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*. 52

Zhang, C. and Ma, Y. (2012). *Ensemble machine learning: methods and applications*. Springer. 49

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., and Oliva, A. (2014). Learning deep features for scene recognition using places database. In *Advances in neural information processing systems*, pages 487–495. 3

Zhu, J., Chen, J., Hu, W., and Zhang, B. (2017). Big learning with bayesian methods. *National Science Review*, 4(4):627–651. 48

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the Elastic Net. *J. R. Stat. Soc. Ser. B*, 67:301–320. 19, 89, 121

Zou, J. and Schiebinger, L. (2018). Ai can be sexist and racist —it is time to make it fair. *Nature*, 559(7714):324. 14

## RÉSUMÉ

Le sur-apprentissage est un problème général qui affecte les algorithmes d'apprentissage statistique de différentes manières et qui a été approché de différentes façons dans la littérature. Nous illustrons dans un premier temps un cas réel de ce problème dans le cadre d'un travail collaboratif visant à prédire la réponse de patients atteints d'arthrose rhumatoïde à des traitement anti-inflammatoires. Nous nous intéressons ensuite à la méthode d'Injection de bruit dans les données dans sa généralité en tant que méthode de régularisation. Nous donnons une vue d'ensemble de cette méthode, ses applications, intuitions, algorithmes et quelques éléments théoriques dans le contexte de l'apprentissage supervisé. Nous nous concentrons ensuite sur la méthode du *dropout* introduite dans le contexte d'apprentissage profond et construisons une nouvelle approximation permettant une nouvelle interprétation de cette méthode dans un cadre général. Nous complémentons cette étude par des expériences sur des simulations et des données réelles. Par la suite, nous présentons une généralisation de la méthode d'injection de bruit dans les données inspirée du bruit inhérent à certains types de données permettant en outre une sélection de variables. Nous présentons un nouvel algorithme stochastique pour cette méthode, étudions ses propriétés de régularisation et l'appliquons au context de séquençage ARN de cellules uniques. Enfin, nous présentons une autre généralisation de la méthode d'Injection de bruit où le bruit introduit suit une structure qui est déduite d'une façon adaptative des paramètres du modèle, en tant que la covariance des activations des unités auxquelles elle est appliquée. Nous étudions les propriétés théoriques de cette nouvelle méthode qu'on nomme ASNI pour des modèles linéaires et des réseaux de neurones multi-couches. Nous démontrons enfin que ASNI permet d'améliorer la performance de généralisation des modèles prédictifs tout en améliorant les représentations résultantes.



## MOTS CLÉS

apprentissage statistique, bruit , réseau de neurones, régularisation, bioinformatique



## ABSTRACT

Overfitting is a general and important issue in machine learning that has been addressed in several ways through the progress of the field. We first illustrate the importance of such an issue in a collaborative challenge that provided genotype and clinical data to assess response of Rheumatoid Arthritis patients to anti-TNF treatments. We then re-formalise Input Noise Injection (INI) as a set of increasingly popular regularisation methods. We provide a brief taxonomy of its use in supervised learning, its intuitive and theoretical benefits in preventing overfitting and how it can be incorporated in the learning problem. We focus in this context on the *dropout* trick, review related lines of work of its understanding and adaptations and provide a novel approximation that can be leveraged for general non-linear models, to understand how *dropout* works. We then present the *DropLasso* method, as both a generalisation of *dropout* by incorporating a sparsity penalty, and apply it in the case of single cell RNA-seq data where we show that it can improve accuracy of both Lasso and dropout while performing biologically meaningful feature selection. Finally we build another generalisation of Noise Injection where the noise variable follows a structure that can be either fixed, adapted or learnt during training. We present Adaptive Structured Noise Injection as a regularisation method for shallow and deep networks, where the noise structure applied on the input of a hidden layer follows the covariance of its activations. We provide a fast algorithm for this particular adaptive scheme, study the regularisation properties of our method on linear and multilayer networks using a quadratic approximation, and show improved results in generalisation performance and in representations disentanglement in real dataset experiments.



## KEYWORDS

machine learning, noise injection, neural networks, dropout, deep learning, bioinformatics