Décompositions tensorielles pour la complétion de bases de connaissance

Timothée Lacroix

UNIVERSITÉ PARIS-EST

UNIVERSITÉ PARIS-EST

Mathématiques & Sciences et Technologies

de l'Information et de la Communication

Thèse de doctorat

Traitement du Signal et des Images

TIMOTHÉE LACROIX

TENSOR DECOMPOSITIONS FOR KNOWLEDGE BASE COMPLETION

Thèse soutenue le 3 juillet 2020

Jury :

Renaud Marlet	École des Ponts ParisTech	Directeur de thèse
Guillaume Obozinski	Swiss Data Science Center	Co-Directeur de thèse
Nicolas Usunier	Facebook AI Research	Co-Directeur de thèse
Anthony Nouy	Centrale Nantes	Rapporteur - Président du jury
Pierre Comon	Université de Grenoble Alpes	Rapporteur
Ryota Tomioka	Microsoft Research Cambridge	Rapporteur
Thomas Hofmann	ETH Zürich	Examinateur
Mikaella Keller	Université de Lille	Examinatrice

Résumé

Dans cette thèse, nous abordons le problème de prédiction de liens dans des tenseurs binaires d'ordre trois et quatre contenant des observations positives uniquement. Ce type de tenseur apparait dans les problèmes de recommandations sur le web, en bio-informatique pour compléter des bases d'interactions entre protéines, ou plus généralement pour la complétion des bases de connaissances. Ces dernières nous permettent d'évaluer nos méthodes de complétion à grande échelle et sur des types de graphes relationels variés.

Notre approche est parallèle à celle de la complétion de matrice. Nous résolvons un problème non-convexe de minimisation empirique régularisé sur des tenseurs de faible rangs. Dans un premier temps, nous validons empiriquement notre approche en obtennant des performances supérieures à l'état de l'art sur de nombreux jeux de données.

Ces performances ne peuvent néanmoins être atteintes que pour des rangs trop élevés pour que cette méthode soit applicable à l'échelle de bases de connaissances complètes. Nous nous intéressons donc dans un second temps à la décomposition Tucker, plus expressive que la paramétrisation de faible rang, mais plus difficile à optimiser. En corrigeant l'algorithme adaptatif Adagrad, nous arrivons à optimiser efficacement des décompositions Tucker dont le tenseur coeur est aléatoire et fixé. Ces méthodes nous permettent d'améliorer les performances en complétion pour une quantité faible de paramètres par entités.

Finalement, nous étudions le cas de base de connaissances temporelles, dans lesquelles les prédicats ne sont valides que sur certains intervalles de temps. Nous proposons une formulation de faible rang et une régularisation adaptée à la structure du problème, qui nous permet d'obtenir des performances supérieures à l'état de l'art.

Mot clés: Apprentissage automatique, Base de connaissance, Décomposition tensorielles

Abstract

In this thesis, we focus on the problem of link prediction in binary tensors of order three and four containing positive observations only. Tensors of this type appear in web recommender systems, in bio-informatics for the completion of protein interaction databases, or more generally for the completion of knowledge bases. We benchmark our completion methods on knowledge bases which represent a variety of relationnal data and scales.

Our approach is parallel to that of matrix completion. We optimize a non-convex regularised empirical risk objective over low-rank tensors. Our method is empirically validated on several databases, performing better than the state of the art.

These performances can however only be reached for ranks that would not scale to full modern knowledge bases such as Wikidata. We focus on tensor factorization in the Tucker form which is more expressive than low-rank parametrizations but also harder to optimize. By fixing the adaptive algorithm Adagrad, we obtain a method to efficiently optimize over tensors in Tucker form with a fixed random core tensor. With these methods, we obtain improved performances on several benchmarks for limited parameters per entities.

Finally, we study the case of temporal knowledge bases, in which the predicates are only valid over certain time intervals. We propose a low-rank formulation and regularizer adapted to the temporal structure of the problem and obtain better performances than the state of the art.

Keywords: Machine learning, Knowledge base, Tensor decompositions

Acknowledgments

I would like to thank the members of the committee - Pierre Comon, Anthony Nouy, and Ryota Tomioka - for taking the time to review this manuscript.

Guillaume, ta clarté et ta patience ont mené à de longues explications au tableau desquelles je ressortais toujours plus instruit. Ton goût pour le détail et la précision auront évité l'écriture de bien des âneries pendant cette thèse. Merci aussi pour ta maîtrise du vspace négatif qui aura su calmer mes paniques de mise en page à l'approche des soumissions. Enfin, merci pour cette superbe opportunité d'enseignement au Rwanda, durant laquelle j'ai probablement autant appris que nos élèves.

Nicolas, tu m'as lentement mais sûrement sorti de mes fantasmes combinatoires pour me guider vers des questions aux réponses envisageables et surtout des solutions qui marchent. Pendant ces trois années, je t'ai sans cesse abreuvé de questions sur des domaines allant de la sparsité en scipy à la calibration de perte sans jamais te mettre en défaut. Merci d'avoir aussi volontairement partagé ton savoir, que ce soit au bureau, dans le métro ou dans des bars.

Les frustrations et les joies d'une thèse auraient été autrement plus dures à supporter hors des murs douillets de Facebook Paris. Sans la délicieuse cuisine de Pini, la nourriture spirituelle de la thèse n'aurait pas suffit à me sustenter. Un grand merci aux autres thésards de Facebook, anciens comme nouveaux, pour l'ambiance chaleureuse du bureau ainsi que le partage des succès et échecs au quotidien. Merci aussi à tout les joueurs de Baby-Foot qui m'auront permis d'exorciser mes frustrations à travers de nombreuses gamelles et railleries de bon goût.

En dehors de Facebook, merci aux Ronds pour leur amitié infaillible, les nombreux week-ends, soirées, projets et vacances toujours aussi remplis en fou rires et activités. Merci à mes parents et à ma famille d'avoir su acérer mon sens critique au fil des ans, et d'avoir été un havre de repos, de paix et de randonnées pendant les trois années de la thèse. Enfin, merci Lauriane d'avoir égayé ces trois années en partageant mon goût pour les jeux, la nourriture et le repos bien mérité.

Contents

Chapter 1

Introduction

The focus of this thesis is link prediction for multi-relational data. Given a fixed set of entities and a set of graphs over these entities the goal of link prediction is to provide a ranking of entities as potential candidates to be neighbors of a node in any of these graphs.

Such link prediction can be useful in various settings, from web scale recommender systems to bioinformatics. Moreover, the link prediction problem can be used as a proxy to learn latent representations of entities (so-called entity *embeddings*) which can be used for various downstream tasks. Such embeddings summarize the observed connectivity patterns of entities in a single vector and can have good properties with respect to classification or algebraic transformations. For example, word embeddings are obtained by solving a link prediction problem over word co-occurences and can then be re-used for many natural language processing applications [27, 107, 42, 48]. We will go into potential applications of link prediction systems further in Section 1.1. In this thesis, we focus on applications to the completion of Knowledge Bases, as they provide a variety of benchmarks, both in terms of scale and in terms of the relational systems they represent.

Our approach is based on classical low-rank or low-multirank tensor parametrizations. We focus on the regularization and optimization of such parametrizations in the context of knowledge base completion.

Some research has focused on finding the best structural constraints to model the

underlying relational data, while other has strived to include external, non-relational data into the learning process. We review the context in which this thesis was written in Section 1.2. This context justifies the direction of research that we followed throughout the thesis and which lead to the contributions which we quickly present in Section 1.3. This thesis established a strong method for link prediction in knowledge bases. This method obtains state of the art results on all benchmark considered and provides the basis for the development of two other contributions. The first one is a method that allows for reduced memory footprint, which is made possible by a careful study of the optimization process. The second contribution is the addition of temporal information in the context of knowledge base completion.

Finally, in Section 1.4, we describe the state of the art after this thesis, and the important research directions remaining. We believe that along this thesis, we have outgrown popular benchmarks, on which the algorithms we propose obtain state of the art results in a matter of minutes. Our last contribution suggests a new benchmark, with 5 times as many entities and 7 times as many training examples as the largest knowledge base benchmark currently in use. This allows for the study of larger scale methods, both for the reduction of the models memory footprint, but also for the added computational cost of making predictions at that scale. Even this increased dataset size should only be a beginning, since most web-scale databases can be expected to have hundreds of millions of entities. Another research direction is the inclusion of other data sources on top of the method we proposed. As it is, our approach without external data performs better than any published result that uses external data. We made a step in that direction by adding temporal metadata, but have yet to address the variety of external data that can be used for this purpose. For example, since the Wikidata knowledge base has its entities linked to Wikipedia pages whenever possible, one could use the data in these pages to inform learning on Wikidata.

1.1 Link Prediction

Link prediction can prove useful in diverse settings:

Recommender Systems In many web applications, users will interact with each others, or with items in different ways. Being able to predict the neighborhood of a user for each type of interactions can help make recommandation systems better. Exploiting a variety of interactions helps making prediction for valuable but sparse interactions. For example, a user might view many more items than they will buy. Predicting the "buy" signal however, is more valuable. Note that in this setting, no negative (user,item) pair can be observed (since the interaction would not happen). Thus, we work in the so-called positive-unlabeled [34] setting, where all the observed edges are positive examples and any other edge is considered either missing or negative.

Bioinformatics Many interaction networks are available. These networks can represent protein-protein interactions [84, 24] or medicine side-effects when taken jointly [69]. By jointly modeling the proteins appearing in medicines, their interactions and observed side-effects [128], one could predict potential adverse side-effects for new pairs of medicines despite lack of experimental evidence. Such predicted pairs could then reduce the experimental search space and help build a more comprehensive list of medicine side-effects. This setting was also used as an example application of the positive unlabeled setting in Elkan and Noto [34].

Knowledge Bases There have been many efforts to compile available knowledge about the world in so-called Knowledge Bases [44, 120, 82]. These databases contain triples (subject, predicate, object) that encode true facts. These facts have either been obtained *via* crowd sourcing or automatic extraction from crawling the web or a mix of both. However, despite their growing size (Wikidata [120] has $60M$ entities and over $700M$ statements at the time of writing), these knowledge bases are still incomplete. Being able to provide good quality candidates for incomplete triples such as (Barack Obama, Married to, ?) can prove instrumental both as a helper for human curators, or for automatic systems who can use it to score the likelihood of an automatically extracted fact being true. Similarly to the recommender setting, no negative edge is observed in this setting.

Representation Learning Independent of the underlying data source, we can view the observed connectivity pattern as the manifestation of a latent signal. The best known application of this idea is word2vec [85]. As made explicit by Levy and Goldberg [73], word2vec learns latent factors of a transformed co-occurence matrix between words. These so called "word embeddings" contain rich semantic information about the words and can be used for applications ranging from language evolution studies [48] to text classification [58]. The success of word embeddings have popularized the concept of learning entity embeddings from various sources, including relational data, in order to use these embeddings for other tasks.

In this thesis, we focus on Knowledge Bases, which provide rich benchmarks with various scales and underlying data sources. Our benchmarks include the Wordnet ontology [86] which represents 18 mostly hierarchical relations between $40k$ English words. We also study a dataset sampled from Freebase [44] which includes various entities from the real world (actors, presidents, institutions, etc...) and over a thousand relations between these entities. In order to deal with such diversity, we design generic methods to deal with sparse positive unlabeled tensors and do not include any apriori knowledge on properties of the underlying relational system.

1.2 Context

Link prediction in graphs or social networks consists in assigning scores $s(x, y)$ to potential neighbors of a node x. For multi-relational data, the problem is similar, but the score to be computed then depends on the relation considered : $s(x, r, y)$. Computing such scores can be done without learning, by computing similarity metrics between the nodes x and y. Liben-Nowell and Kleinberg [75] provides an overview and benchmark of such methods on a subset of the Arxiv [83] citation network, comparing metrics such as the shortest distance between x and y, or the Jaccard's coefficient (intersection over union) of their neighborhoods. These methods do not provide explicit embeddings of the entities. This can be beneficial since these methods can work at very large scales without memory requirements beyond storage of the network itself. A

natural generalization of such methods is to compute many features for potential node pairs (x, y) and learn a classifier on top of this feature vector. This is the approach of the path ranking algorithm [71] in multi-relational graphs, which learns to classify vectors of bounded-length path types between nodes.

In this thesis, we focus on methods that learn embeddings of entities. These methods are more general, as they don't require handtuned similarity features. Moreover, the learned embeddings can be useful in other applications. The generic approach for link prediction with learned entity embeddings is to parameterize the score function for triples (subject, predicate, object) with vectorial representations of these three elements. For example, TransE [13] uses $d(i, j, k) = \|u_i + v_j - u_k\|_2^2$, for vectors $u_i, v_j, u_k \in \mathbb{R}^d$ and ranks potential neighbors according to this distance. A variety of other methods have been proposed, some re-using the translation idea [90, 80], others based on classical tensor parametrizations [91, 118, 126] or on ideas from deep-learning [93, 30]. All these methods focused on building the right amount of "interactions" between entity and predicate parameters in order to encode priors about the relationships naturally found in knowledge bases. However, priors such as the translation based scoring function of TransE lead to models with limited expressivity [61].

Noting that low-rank tensor parametrizations are lagging far behind competing methods [93] at the beginning of this thesis, we decided to explore the reasons for this performance gap. We thus study the use of classical low-rank tensor parametrizations for link prediction.

1.3 Contributions

Noticing the favorable impact that trace-norm regularizers had on matrix completion compared to pure low-rank methods [114, 37, 66], we decided in this thesis to take a parallel approach: use generic tensor parametrizations and use tensor norm based regularizers to ensure generalization. This approach leads to conceptually simple methods with only two hyperparameters: the rank and the regularization weight.

Since regularization is independent from all other factors, the rank can be set *a priori* to the maximum rank allowed by the hardware.

In our first work described in Chapter 3 we describe a simple method for knowledge base completion based on the low-rank parametrization [53]. This work lead to a publication at the ICML conference in 2018:

- Timothee Lacroix, Nicolas Usunier, Guillaume Obozinski. Canonical Tensor Decomposition for Knowledge Base Completion. In *Proceedings of the 35th International Conference on Machine Learning* volume 80 pages 2863-2872, 2018.

We first note that any knowledge base completion method should be invariant to the inclusion of a predicate or its reciprocal in the training data. For example, the predicates has_played_in_movie rather than has_actor represent the same connectivity data, and the choice of including one or the other in the knowledge base should not affect the outcome of the learning algorithm. Bailly et al. [5] address this issue for classical tensor factorization algorithm, giving conditions under which an algorithm will be "semantically invariant". Other methods such as [12, 126, 118] optimized two losses at once: one where the right-hand side of positive triples is considered as missing, and one where the left-hand side is considered as missing. In our work, we add reciprocal predicates to the training set. Doubling the number of predicates in the dataset allows us to only predict missing objects. It also makes the training procedure invariant to the inclusion of a predicate or its reciprocal in the dataset. We propose to optimize a multiclass logistic loss, rather than a pairwise ranking loss to solve the link prediction problem. Indeed, computing the softmax over all entities can be done efficiently on modern gpus. Finally, we show that the usual weight-decay penalty cannot be related to a tensor norm and is non-convex over tensors, we propose to use instead a variational form of the tensor nuclear 3-norm. Our experiments show that our method applied to the ComplEx tensor parametrization [118] leads to large gains over the state of the art on all benchmarks considered.

The best solutions found by the previous method use many parameters per entity.

This can become a problem when dealing with modern knowledge bases such as Wikidata [120], which contains $60M$ entities. Indeed, storing embeddings of each entity for this knowledge base over 100 floats leads to a memory usage of $24GB$ which is impractical on most GPUs available today. In order to address this memory limitation, it is natural to turn to the Tucker parametrization [119] which provides richer interactions between entities and predicates *via* a core-tensor. However, learning this parametrization is difficult, as exemplified by the use of deep-learning heuristics in [7]. In Chapter 4 , we show that these difficulties stem from the use of adaptive algorithms such as Adagrad [32] which apply a diagonal rescaling to the learning rate. We propose a new algorithm for the optimization of a constrained version of Tucker which implicitely applies the rescaling in a higher dimensional space, before projecting the embeddings on a fixed lower dimensional subspace. The resulting Tucker-style extension of ComplEx obtains similar best performances as ComplEx, with substantial gains on some datasets under constraints on the number of parameters.

Finally, we apply our methods to *temporal* relational data in Chapter 5, published at the ICLR conference in 2020:

- Timothee Lacroix, Guillaume Obozinski, Nicolas Usunier. Tensor Decompositions for temporal knowledge base completion. In *International Conference on Learning Representations (ICLR)*, 2020.

Facts in multi-relational databases can often be associated with temporal metadata. For example, the timestamp of a certain user-item interaction, or the validity interval for a triple such as (François Hollande, spouse, Ségolène Royal). In order to deal with this metadata, we discretize time, and add a 4-th mode to the tensor we try to complete. We justify new regularization schemes and present an extension of ComplEx that achieves state-of-the-art performance. Additionally, we propose a new dataset for knowledge base completion constructed from Wikidata, larger than previous benchmarks by an order of magnitude, as a new reference for evaluating temporal and non-temporal link prediction methods.

All of the results in this thesis can be reproduced with the code available at

https://github.com/facebookresearch/kbc.

1.4 Future research directions

Current benchmarks for knowledge base completion use under $100k$ entities, which is far from the scale at which these algorithms are meant to be applied. One of the issue that needs to be addressed in order to be able to deal with larger benchmark such as the one we propose in Chapter 5 is the issue of approximating the cross-entropy loss for millions of entities. There is a rich literature in natural language processing on this topic, and a thorough study of the application of these methods to large knowledge base completion would be beneficial. Even equipped with a fast loss computation method, the issue of model size remains. Our work of Chapter 4 allows us to get closer to a usable Tucker parametrization. Finally, all our attempts at using feed-forward neural networks to predict the score of a triple have lead to similar or worse results than the tensor based methods of this thesis. These failures likely stem from a lack of proper regularization but also from similar optimization problems than those highlighted in Chapter 4. Understanding and fixing those issues would likely have impact beyond knowledge base completion, but can easily be studied in this setting, with the strong baselines provided in this work.

1.5 Organization of the thesis

In Chapter 2, we describe the framework for knowledge base completion in greater detail and introduce the two tensor parametrizations that we will be working with. We review related work on link prediction for knowledge base completion. Finally, we discuss previous results on matrix and tensor completion, which inspired the methods in this thesis.

In Chapter 3 we present a method for knowledge base completion based on low-rank tensor parametrization with regularizers inspired from tensor nuclear norms. With this method, we obtain state of the art results on popular knowledge base completion

benchmarks.

In Chapter 4, we study difficulties associated with learning a Tucker parametrization. We design a simple experiment that exhibits the failure of the diagonal rescaling in Adagrad [32]. We formulate a solution which allows us to train a constrained version of the Tucker parametrization adapted to ComplEx [118]. This formulation matches the state of the art on all datasets and obtains better performances in the low-memory regime.

In Chapter 5, we show how our approach can be generalized to deal with temporal metadata. By discretizing the time, we include it as a 4-th mode in our tensors and develop extensions of ComplEx [118] with adapted regularizers. We obtain state-of-the-art results on current temporal knowledge base completion benchmarks and propose a new larger-scale dataset based on Wikidata [120].

Chapter 2

Related Work

> Coming back to where you started is not the same as never leaving.
>
> Terry Pratchett

In this thesis, we study tensor parametrizations for link prediction at scale. We focus on an application to knowledge bases because it provides a number of benchmarks with associated literature. We begin this chapter with a definition of the link prediction problem for knowledge bases. In Section 2.1, we describe the learning set-up and the metrics we optimize for. We also discuss the various losses used in the literature to optimize these metrics.

Ultimately, our goal is to learn a tensor $\hat{X}$ that assigns scores to all potential triples (subject, predicate, object). We approach the learning of this tensor through low-rank or low-multirank parametrizations. In Section 2.2, we discuss two important parametrizations, the low-rank parametrization which expresses a tensor as a sum of rank-1 tensors and the Tucker parametrization.

We then briefly review the various approaches to knowledge base completion in Section 2.3, but do not focus too much on approaches that are tailored to the problem of knowledge base completion which we only use as a benchmark.

Finally, we discuss in greater details the previous results on matrix and tensor completion which inspired the methods we study in this thesis.

2.1 Framework

Notations Matrices and tensors will be denoted by uppercase letters and lowercase letters will be used for scalars and vectors. The order of the objects will always be clear from context. For a matrix X, we use X_i to denote its i-th row treated as a column vector and $X_{:,j}$ to denote its j-th column. No matter the order of the object considered, the dot-product notation $\langle \cdot, \cdot \rangle$ is always to be understood as the dot-product of the vectorization of the objects. That is, for order-three tensors :

$$\langle X, Y \rangle = \sum_{i,j,k} X_{i,j,k} Y_{i,j,k}.$$

The tensor product will be denoted $\otimes$ and the Hadamard product (or elementwise product) $\odot$. Uses of $\otimes$ for the Kronecker product will be made clear in the text. All norms will be written $\| \cdot \|$ and identified by a subscript. Working on knowledge bases, we assume a fixed set of entities E and predicates (or relations) P. Given a knowledge base containing true triples (subject, predicate, object), we split this set uniformly at random into a training set S, a validation set and a test set. We denote by X the tensor such that $X_{i,j,k} = 1$ if and only if the triple (i, j, k) is a true fact and $X_{i,j,k} = 0$ otherwise. We learn a tensor $\hat{X}$ such that $\hat{X}_{i,j,k}$ represent the score of the triple (i, j, k).

In Chapter 5, we consider temporal knowledge bases with a fixed set of timestamps T. These timestamps will lead us to introducing tensors of order 4 in which $X_{i,j,k,t}$ will be 1 if and only if the triple (i, j, k) is true at timestamp t.

In the next subsection 2.1.1, we define the metrics that we will optimize for and report in our experiments. In the following subsection 2.1.2 we discuss some assumptions on the observed triples and their consequences on the learning objective. Finally, in subsection 2.1.3, we present and compare the three main losses that have been used in the knowledge base completion literature so far.

2.1.1 Metrics

The practical use we optimize for is that of queries by human users looking for the completion of incomplete triples in a knowledge base. For example, a user might want to query the knowledge base looking for presidents of the United States. We formalize this query as (United States, has president, ?). Notice that such queries will always admit at least one correct completion. We dismiss non-sensical partial queries such as (Table, has president, ?) since they are not part of our query framework. Given an incomplete triple, we want our system to output a ranked list of entities that are likely to be valid completion of the query. Let $rank(\hat{X}_{i,j,:}; k)$ be the rank of $\hat{X}_{i,j,k}$ in the sorted list of values of $\hat{X}_{i,j,:}$. We consider the following definition of rank

$$rank(\hat{X}_{i,j,:}, k) = \sum_{k' \in E} \mathbb{1}(\hat{X}_{i,j,k'} \geq \hat{X}_{i,j,k}). \tag{2.1}$$

This definition is not a true rank when there are ties. However, since all the methods assume "lower rank is better", this definition of rank yields a pessimistic evaluation of performance.

Following previous work, we focus on the Mean Reciprocal Rank [93] and Hits@K [13] metrics:

$$MRR(\hat{X}) = \frac{1}{|S|} \sum_{(i,j,k) \in S} \frac{1}{rank(\hat{X}_{i,j,:}; k)} \qquad H@\kappa(\hat{X}) = \frac{1}{|S|} \sum_{(i,j,k) \in S} \mathbb{1}(rank(\hat{X}_{i,j,:}; k) \leq \kappa). \tag{2.2}$$

The Mean Reciprocal Rank focuses mainly on having true triples at the top of the ranking list. A change that affects triples around the k-th position will have an effect of the order of $1/k^2$ on the metric. Similarly, the hits at k metrics only requires proper ranking until position k. This is common in information retrieval where only a limited amount of information can be displayed to the user and only the relevance of the displayed information matters.

In these definitions, we deviated slightly from the literature by only focusing on one type of fibers from the estimated tensor $\hat{X}$ whereas Bordes et al. [12] considers ranking in both $\hat{X}_{i,j,:}$ and $\hat{X}_{:,j,k}$. We argue in Chapter 3 that reciprocal predicates

should be included in the training set, which allows us to answer queries of the form (?, stars in movie, Alien) with the reciprocal query (Alien, has actors, ?). Finally, we use a filtered version of these metrics using the filtered rank $\overline{rank}$. For a query $(i, j, ?)$ there might be multiple positive answers which we denote by $K^{i,j}$. The filtered ranks ignore these positive answers in its computation. Formally, we have :

$$\overline{rank}(\hat{X}_{i,j,:}; k) = \sum_{k' \in (E \setminus K^{i,j})} \mathbb{1}(\hat{X}_{i,j,k'} \geq \hat{X}_{i,j,k}).$$

Since our computation of the rank is pessimistic, all of the metrics we report based on the rank are also pessimistic. However, the models we consider should not produce ties, and thus, should not be affected by these considerations.

2.1.2 Unobserved Triples

Unobserved triples can either be true triples that haven't been observed, or triples that do not hold. This setting is called positive-unlabeled and has been previously studied [34, 8]. A typical result in positive-unlabeled learning is that the posterior probabilities for the problem of separating positive and unlabeled examples and the problem of separating positive and negative examples lead to the same rankings [34, 10].

These results and the metrics defined in Equation (2.2) lead us to to consider all unlabeled examples as negatives. This is not to be misunderstood with the formal logic interpretation of knowledge bases which contrasts the *closed-world* (all unlabeled triples are negative) or *open-world* (unlabeled triples may either be positive or negative) assumptions. In order to learn a proper ranking of triples under the *open-world* assumption, namely that triples that are unlabeled are either negative or positive, we are allowed to consider missing triples as negative in our training objective.

2.1.3 Losses

As previously discussed, we aim to provide a correct ranking of entities for incomplete queries $(i, j, ?)$. We find three categories of loss that aim to do this in the knowledge

base completion literature :

Logistic loss According to the previous discussion, we consider all observed triples as positive examples and all unobserved triples as negative ones. This leads to the optimization of the following loss, where N and P are the numbers of entities and predicates:

$$\mathcal{L}(\hat{X}) = \frac{1}{N^2 P}\left(\sum_{(i,j,k)\in S} \log(1+\exp(\hat{X}_{i,j,k})) + \sum_{(i,j,k)\notin S} \log(1+\exp(-\hat{X}_{i,j,k}))\right). \quad (2.3)$$

In order to minimize this loss for large tensors, the stochastic gradient descent algorithm [99] can be used. Previous work [118] produces a stochastic estimate of the loss (2.3) by sampling a "negative set" $\tilde{K}$ for each positive triple (i, j, k) to act as a set of negative samples. Larger sets $\tilde{K}$ lead to better estimation of the negative part of the loss, but longer computations. In order to obtain more "informative" negative sets, previous work obtain negative sets using perturbed version of true triples (i, j, k) by sampling either a new object or subject [13, 118, 30, 61]. Since we consider dataset with added reciprocal relations, this simplifies to only sampling new objects. Thus, the instantaneous stochastic loss these authors optimize for a given true triple (i, j, k) and negative set $\tilde{K}$ is:

$$\mathcal{L}(\hat{X}; (i,j,k); K) = \log(1+\exp(\hat{X}_{i,j,k})) + \sum_{k'\in\tilde{K}} \log(1+\exp(-\hat{X}_{i,j,k'})).$$

This loss is similar in spirit to the negative sampling loss used for learning word embeddings in [85]. Note that in this loss, an implicit weight is given to negatives, since $|\tilde{K}|$ negatives are sampled for each positive. This can be interpreted as the negative weighting scheme suggested in [34]. A normalizing factor $\alpha/|\tilde{K}|$ in front of the loss associated to negative samples would allow to decouple this weighting from the computing budget but is often forgotten in practice.

Triplet loss Triplet losses have been successful for large scale learning to rank applications [123, 25]. Consider the definition of the rank in Equation (2.1). It can be interpreted as an expectation over the negative triples for a single positive triple (i, j, k). Then, by sampling a negative triple in a similar fashion as in the previous paragraph, one can obtain a convex surrogate of the rank by replacing the indicator function in Equation (2.1) with a hinge loss. Such a margin-based ranking criterion has been used in [14, 13, 126, 93]. For γ the margin parameter of the hinge-loss, we minimize for a positive triple (i, j, k) and sampled negative entity k' :

$$\mathcal{L}(\hat{X}; (i, j, k); k') = \max(0, \gamma - (\hat{X}_{i,j,k} - \hat{X}_{i,j,k'})). \tag{2.4}$$

Intuitively, we want the score of a positive triple to be greater than the score of negative triples by at least γ to incur no loss. Note that in this formulation, there is no implicit weighting associated to the number of sampled negatives. The loss in Equation (2.4) will minimize the average rank. In order to put more importance on higher ranks, the authors of [123] use a different sampling scheme along with a weighting of the loss. The impact of such re-weighting of the loss (2.4) has not been evaluated in the knowledge base completion setting to the best of our knowledge. Another alternative to the ranking loss in Equation (2.4) would be to use the binary cross-entropy in order to obtain a surrogate for the rank. We have not found this loss used in the knowledge base completion literature, despite this option removing the margin parameter in the hinge loss. Early experiments with this loss showed no improvements over the hinge loss surrogate, and didn't perform as well as the cross-entropy loss we present next.

Cross-Entropy Considering the observed neighborhood of a subject i for a certain predicate j as a sampling from a multi-class distribution $P_{i,j}$ justifies the minimization of the cross-entropy:

$$\mathcal{L}(\hat{X}; (i, j, k)) = -\hat{X}_{i,j,k} + \log \left(\sum_{k' \in E} \exp(\hat{X}_{i,j,k'}) \right). \tag{2.5}$$

This formulation has been deemed impractical due to the high computational complexity of computing the leftmost term of Equation (2.5). This has lead to approximations being used in [117, 60, 59]. However, considering the small size of current knowledge base completion benchmark, we use the complete cross-entropy to obtain state-of-the-art results as described in Chapter 3. This approach has the benefits of removing the γ parameter of margin-based approaches, as well as the number of negative samples (and/or negative weights) for sampling based approaches, reducing the experiment space. Moreover it is computationally efficient for the models considered in Chapter 3,4,5, since computing scores over batch of fibers can be done *via* matrix-matrix products.

2.2 Tensor parametrizations

In order to enable learning of the score tensor $\hat{X}$, we write this tensor as a small sum of simple components. This motivates a review in this section of two tensor parametrizations. Despite most of our work being with order 3 tensors, we present in this section the parametrizations for any order. This will be useful in Chapter 5 where we work with tensors of order 4.

A matrix X can be parameterized as a sum of rank 1 matrices : $X = \sum_r u_r \otimes v_r = UV^\top$. In a similar way, tensors can be written as a sum of pure (ie, rank 1) tensors, which we call the low-rank parametrization presented in subsection 2.2.1. Another parametrization however, the Tucker parametrization, may allow for more compact tensor representation. We present this second parametrization in subsection 2.2.2. Our presentation is limited to what is strictly necessary for the understanding of this thesis. For a detailed description of tensor operations and parametrizations, the reader is referred to the survey by Kolda and Bader [64].

2.2.1 Low-rank parametrization

Similar to the parametrization of matrices as sum of rank 1 matrices, we can write a tensor X as a sum of rank 1 tensors : $X = \sum_r u_r^{(1)} \otimes \cdots \otimes u_r^{(d)}$. The minimum

number of terms in this parametrization for a fixed tensor X is called the tensor rank. Contrary to the matrix rank, this rank can decrease if the factorization is done over $\mathbb{C}$ rather than over $\mathbb{R}$ [28]. Another difference is that computing the rank of a tensor is NP-hard [52]. Despite these difficulties, this parametrization has been used in practice with non-minimal number of factors [50, 22] in which case it is dubbed CANDECOMP/PARAFAC. Note that the individual elements of X can be written as the multilinear product

$$X_{i_1,\cdots,i_d} = \sum_r \prod_k U^{(k)}_{i_k,r} \stackrel{\text{def}}{=} \langle U^{(1)}_{i_1}, \cdots, U^{(d)}_{i_d} \rangle.$$

In this thesis, we will consider the problem of learning low-rank parametrizations of tensors. In accordance with the terminology in the knowledge base completion literature, we will use the term CP parametrization even when the number of factors is higher than the tensor rank.

In order to easily manipulate this parametrization, some vocabulary is needed. Leaving only one index free in a tensor X: $X_{i_1,\cdots,i_{k-1},:,i_{k+1}\cdots,i_d}$ we obtain the mode-k fibers. Symetrically, fixing only one index and leaving all others free : $X_{:,\cdots,:,i_k,:,\cdots,:}$, we obtain the mode-$k$ slices. The fibers are always vectors, and in the specific case of order 3 tensors which are important to the work in this thesis, the slices will be matrices. Unfolding is the operation of joining multiple modes by considering the Cartesian product of their indices. We will make use of unfoldings on tensors of order 4 in Chapter 5. A tensor of size $(3, 4, 5, 6)$ becomes a tensor of size $(3, 20, 6)$ by unfolding mode 2 and 3 together. Taking this unfolding as an example, note that

$$X_{i_1,(i_2,i_3),i_4} = \langle U^{(1)}_{i_1}, U^{(2)}_{i_2} \odot U^{(3)}_{i_3}, U^{(4)}_{i_4} \rangle.$$

We can write this more conveniently with the Kathri-Rao [106] product $\circ$ (which is the columnwise Kronecker product) as :

$$X_{i_1,(i_2,i_3),i_4} = \langle U^{(1)}_{i_1}, U^{(2,3)}_{(i_2,i_3)}, U^{(4)}_{i_4} \rangle \qquad \text{where } U^{(2,3)} = U^{(2)} \circ U^{(3)}.$$

This shows that under a CP parametrization, a tensor can easily be unfolded by taking the Kathri-Rao product of its modes.

2.2.2 Tucker parametrization

The Tucker parametrization uses a core tensor G of dimensions $p_1, \cdots, p_d$. The score of one element in X can then be written:

$$X_{i_1,\cdots,i_d} = \sum_{a_1,\cdots,a_d} G_{a_1,\cdots,a_d} \prod_k U^{(k)}_{i_k,a_k} = \langle G, U^{(1)}_{i_1} \otimes \cdots \otimes U^{(d)}_{i_d} \rangle.$$

The Tucker parametrization can be written with the shorthand $X = [\![C; U, \ldots, W]\!]$ [65]. It should be noted that for the identity G such that $G_{i,\cdots,i} = 1$ and 0 everywhere, the Tucker parametrization reduces to a CP parametrization of same rank. In that case, we use the shorthand $X = [\![U^{(1)}, \ldots, U^{(d)}]\!]$. However, for arbitrary G, the expressions of $X_{i_1,\cdots,i_d}$ becomes richer which could lead to tensors being decomposed with much lower ranks in the Tucker parametrization than in the CP parametrization [64]. In Chapter 4 we learn a Tucker parametrization by exploiting its links with a CP parametrization that has low-rank factors [17, 18].

2.3 Link prediction in knowledge bases

In this section, we present the various approaches to predicting links in knowledge bases.

Stochastic Block Model An important model in statistics is the Stochastic Block Model for binary matrices. Under this model, entities are grouped into clusters (or communities), and the graph from nodes of clusters a to nodes of cluster b is an Erdös-Rényi graph of parameter $\eta(a, b)$. Precisely, edges from cluster a to b are independent samples from a Bernoulli distribution of parameter $\eta(a, b)$. In this framework, the main goal is to recover the clusters, given the observed connectivity matrix. This model exhibits sharp thresholds for recovery [29] which have been studied in details

[1]. A multi-relational, non-parametric version of this model has been proposed in Kemp et al. [62].

Low-rank parametrizations RESCAL [91] is a special case of a Tucker parametrization : $\hat{X}_{i,j,k} = U_i^\top R_{:,:,j} U_k$ and is a relaxed version of the DEDICOM parametrization [49]. Entity embeddings are shared for subjects and objects. Relations are expressed via a bi-linear product with matrices $R_{:,:,j}$ which allows to model non-symmetric relationships. In Bordes et al. [14], two models are suggested, linear and bi-linear. The first (linear) models write $\hat{X}$ as a sum of interactions between left-hand side and relation, relation and right-hand side or left-hand side and right-hand side. The second (bilinear) models are similar to RESCAL, but express the matrices $R_{:,:,j}$ as a linear combination of fewer matrices and adds biases. Jenatton et al. [56] parameterize the relation matrices, as a sum of few rank-1 matrices which are shared across relations. Yang et al. [126] simplified RESCAL further by learning diagonal relation matrices in a model called DistMult. This restriction however, limits the expressivity of the model, since the slices of $\hat{X}$ along the predicate mode are now symmetric. Despite its limited expressivity, this model has shown experimental success on popular benchmarks [60]. Trouillon et al. [118] introduced complex embeddings in a parametrization called ComplEx in order to model non-symmetric relationships with a model complexity similar to that of DistMult. More recently, attempts have been made at learning a Tucker3 parametrization of $\hat{X}$. Balazevic et al. [7] imposes no restriction on the parameters, and uses deep-learning techniques such as dropout and batch-norm to learn this parametrization. Wang et al. [121] imposes a sparsity constraint on the core tensor of the Tucker parametrization they learn, in order to make it closer to the core tensors of the CP or ComplEx parametrization. Kazemi and Poole [61] uses a CP parametrization and similarly to our work in Chapter 3, introduces the use of reciprocal relations.

Translation based approaches Bordes et al. [13] model the score of a triple $\hat{X}_{i,j,k}$ as $\|e_i + r_j - e_k\|_2^2$ in a method called TransE. Such translation based scoring have lead

to many variants [90, 80, 76, 36]. These methods add extra parameters to TransE by adding relation-dependent linear transforms to modify the left-hand side and right-hand side entities. Another direction in which TransE has been modified is by changing the geometry in which the distance is computed. TorusE [33] embeds this computation on a Torus, whereas MurP [7] uses hyperbolic spaces. It has been argued in Kazemi and Poole [61] that translation based methods may not be able to represent all relational systems.

Other approaches Socher et al. [108] proposes to add a triplet interaction to a neural network's linear layer. They use a classification loss over randomly sampled (but type accurate) triples rather than a ranking loss. Another method based on neural networks favors higher number of layers, but only uses concatenations of entity and relation embeddings [31]. Holographic Embeddings [93] uses circular correlation between embedding entities before taking the dot product with the relation embeddings. This method, which performs faborably against the method of Dong et al. [31], was shown to be equivalent to ComplEx [118]. More recently, Dettmers et al. [30] used convolutional filters on a tiling of entity and relation embeddings in a neural network model called ConvE.

Discussion Bayesian methods do not scale to larger benchmarks such as FB15k and thus have largely been abandoned for knowledge base completion. Tensor parametrizations such as RESCAL [91] or CP showed poor performances on FB15k [13]. In order to reduce the potential overfitting [13, 93] of these methods, the community focused on designing models that were less expressive, adding prior knowledge through the constrained parameterizations of the models. For example, Bordes et al. [13] attempted to focus on hierarchical relationships by modeling the different relations as translations. The simpler parametrization lead to better performance and many variations of the method [90, 80, 76, 36]. Constrained parametrization of the CP parametrization, with shared factors for object and subject lead to DistMult [126] and ComplEx [118] which obtained better performances than translation based methods on

the FB15k benchmark. Currently, state of the art results are obtained with ComplEx parametrizations (see Chapter 3) for large number of parameters. Interest is growing in studying Tucker parametrizations which provides a natural way to reduce the size of the embeddings (see Chapter 4, [7, 121]). Research into constrained parametrization has broadened its scope with a study of the embeddings base space [33, 6], motivated by the success of hyperbolic embeddings [94] for modeling hierarchies.

2.4 Optimization

In order to learn the tensor parametrizations considered in this thesis, we will optimize an objective that can be written as a finite sum of losses (eg. multiclass logistic loss) over the positive examples :

$$\min_{\theta} \sum_{(i,j,k)\in S} \ell(i,j,k;\theta).$$

This problem can be approached for general ℓ with batch or stochastic gradient descent (SGD) [99]. The descent direction in these methods is either the gradient of the entire objective or an unbiased estimate of this gradient. Some optimization methods adapt this descent direction to account for the curvature of the parameter space. Newton-Raphson or L-BFGS [77] use the inverse of the Hessian or an approximation for this adaptation. The heavy-ball method [96] uses an exponential moving average of the descent directions and has proved successful for the practical optimization of deep neural networks. For large datasets, modern approaches favor stochastic gradient descent methods rather than batch [16]. Moreover, gradients for stochastic updates in our case are sparse, leading to efficient SGD steps. Adagrad [32] only uses first-order information to rescale the descent direction coordinate-wise. This algorithm is described in detail in Chapter 4 of this manuscript. Adam [63] combines ideas from the heavy-ball method and Adagrad and has been widely adopted by deep-learning practitioners. Previous work on knowledge base completion used Adagrad [93, 118] or Adam [7]. Our own experiments confirmed that adaptivity of the optimization

algorithm is crucial as stochastic gradient descent fails to efficiently escape the saddle point at the origin or diverges.

A frequent approach in the tensor literature to deal with large tensors is to compress them first with a higher order singular value decomposition (HOSVD) [23, 17], then work on the compressed tensor. However, the HOSVD finds the best Frobenius approximation of a tensor, which is ill-adapted to the losses we optimize in this thesis, presented in section 2.1.3.

2.5 Matrix and Tensor regularizers

We aim at learning a tensor $\hat{X}$ from incomplete or partial informations: the observed relationships. This cannot be done without assumptions on $\hat{X}$ (such as low rank or low norm) to bound the complexity of the model space. For matrix completion, as in the popular Netflix Prize [9], trace-norm regularization has been very successful. In fact, bounding a non-convex variational form of the trace-norm, introduced in section 2.5.1, rather than the rank of the model was used in the winning method for the Netflix Prize [66]. In order to understand the success of this method, and properly generalize it to tensors, we review in the next subsection the bounds and guarantees that can be obtained for learning matrices. Notably, we will see the importance of the sampling distribution of observations and how it affects the norm to penalize [110].

Several norms have been defined over tensors and in particular the matrix trace norm has been generalized to tensor norms in many ways [78, 105, 39, 116]. In section 2.5.2 we will define the tensor trace norm considered in [127, 38], also called tensor nuclear norm which we will use throughout this manuscript.

In the next subsection, we introduce two matrix norms and associated PAC-bounds as well as review some results and assumptions from the exact recovery literature. Then we will see how these results were adapted to higher order tensors using matricization. Finally, we will define and discuss some properties of the tensor nuclear norm and its differences with the matricization-based norms when it comes to exact recovery.

2.5.1 Matrix norms and associated guarantees

Before describing higher order tensor norms, we describe in this section some related results on matrices. We follow the the analysis framework of Srebro [109]. For a fixed elementwise $\text{loss}(X_{i,j}; Y_{i,j})$, we define for X and Y of size $n \times m$ and S a set of observed indices :

$$\mathcal{D}(X;Y) = \frac{1}{nm}\sum_{i,j}\text{loss}(X_{i,j};Y_{i,j}) \quad \mathcal{D}_S(X;Y) = \frac{1}{|S|}\sum_{i,j\in S}\text{loss}(X_{i,j};Y_{i,j})$$

$\mathcal{D}$ is the true average overall error, while $\mathcal{D}_S$ is the average error over the observed entries. Note that this setting does not fit with the ranking or cross-entropy losses that were discussed in section 2.1.3. Matrix completion has been theoretically studied for elementwise squared error or logistic loss.

Low-rank PAC-bounds Let Y be a target sign-matrix, $Y \in \{\pm 1\}^{n\times m}$ and $X \in \mathcal{X}$ an estimate of Y. Consider the sign-agreement loss : $\text{loss}^{\pm}(a,b) = \mathbb{1}(ab \leq 0)$. Noting that the errors for this loss $\mathcal{D}^{\pm}$ only depend on the sign-pattern of X, the authors of [112] consider the number of sign-patterns in set $\mathcal{X}$. Consider :

$$f(\mathcal{X}) = \left|\{\text{sign}X \in \{-,0,+\}^{n\times m} \mid X \in \mathcal{X}\}\right|.$$

The following inequality holds with probability $1-\delta$ over the choice of S for any $1 \geq \delta > 0$ [112] :

$$\forall X \in \mathcal{X} \quad \mathcal{D}(X;Y) \leq \mathcal{D}_S(X;Y) + \sqrt{\frac{\log f(\mathcal{X}) - \log\delta}{2|S|}}. \tag{2.6}$$

In order to bound $f(\mathcal{X})$ for real low-rank matrices, Srebro et al. [112] use the following proposition:

Proposition 1. *[3] The number of $-/0/+$ sign configurations of r polynomials, each of degree at most d, over q variables, is at most $(8edr/q)^q$ (for $r > q > 2$).*

Let $\mathcal{X}^k_{n,m}$ be the set of real matrices of size $n \times m$ and rank at most k. Then for

any $X \in \mathcal{X}^k_{n,m}$, there exists $U \in \mathbb{R}^{n\times k}$, $V \in \mathbb{R}^{m\times k}$ such that $\forall i,j\ X_{i,j} = \langle u_i, v_j\rangle$. Thus, $f(\mathcal{X}^k_{n,m})$ is the number of $-/0/+$ sign configurations of $n \times m$ polynomials of degree at most 2 over $k(n+m)$ variables. According to Proposition 1 :

$$f(\mathcal{X}^k_{n,m}) \leq \left(\frac{8e.2.nm}{k(n+m)}\right)^{k(n+m)} \leq \left(\frac{16em}{k}\right)^{k(n+m)}.$$

Together with inequality (2.6), we have for any $\delta > 0$, with probability $1-\delta$:

$$\forall X \in \mathcal{X}\ \mathcal{D}(X;Y) \leq \mathcal{D}_S(X;Y) + \sqrt{\frac{k(n+m)\log\frac{16em}{k} - \log\delta}{2|S|}}.$$

A matching lower-bound can be obtained for matrices by considering arrangements of hyperplanes in $\mathbb{R}^k$ [109]. Proposition 1 can be used to yield similar upper bounds on the sign-configurations of low-rank tensors. Matching lower-bounds for CP or Tucker parametrizations however do not follow from the same proof as the one used in [109].

Trace-norm We denote the trace-norm (or nuclear-norm) of a matrix M by $\|M\|_*$. The following definitions, where U and V are always full-rank matrices, are all equivalent [111] :

$$\begin{aligned}
\|M\|_* &= \inf_{M=UV^\top} \frac{1}{2}\left(\|U\|_2^2 + \|V\|_2^2\right)\\
\|M\|_* &= \inf_{M=UV^\top} \|U\|_2\|V\|_2\\
\|M\|_* &= \|\sigma\|_1 \qquad \text{where } M \text{ has SVD } M = U\mathrm{Diag}(\sigma)V^\top
\end{aligned}$$

The trace-norm provides a convex surrogate for the rank [35, 113]. Let us define the margin sign-agreement loss as $\mathrm{loss}^\gamma(a,b) = \mathbb{1}(ab \leq \gamma)$. Srebro et al. [113] provide the following PAC-bound for matrices of low trace-norm :

Theorem 1. *[113] For all target matrices $Y \in \{\pm 1\}^{n\times m}$ and sample sizes $|S| > n\log n$ (with $m \leq n$), and for a uniformly selected sample S of $|S|$ entries in Y, with probability $1-\delta$ over the sample selection, the following holds for all matrices $X \in \mathbb{R}^{n\times m}$ and all*

$\gamma > 0$*:*

$$\mathcal{D}^{\pm}(X,Y) \leq \mathcal{D}_S^{\gamma}(X;Y) + \tilde{\mathcal{O}}\left(\frac{\|X\|_*}{\gamma\sqrt{nm}}\sqrt{\frac{(n+m)}{|S|}}\right) + \sqrt{\frac{\ln(4/\delta)}{2|S|}}$$

where $\tilde{\mathcal{O}}$ *is the soft-O notation ignoring log-factors in* n*,* m*,* γ *and* $\|X\|_*$*.*

Note that in this bound, the rank has completely disappeared, leaving all capacity control to the trace-norm. More strikingly, minimizing the trace-norm allows for exact recovery of matrices, with only a fraction of observed entries: an unknown rank r matrix Y of low coherence with the canonical basis (that is, the singular vectors of Y are not aligned with the canonical basis) can be recovered with high-probability from $|S| > C(n+m)r\log^2(n+m)$ entries, where C depends on the coherence [21, 97, 45].

As discussed in subsection 2.5.2, the tensor nuclear-norm enables extensions of these matrix recovery results. However, generalizing the PAC-bound would require a tensor version of the result in Seginer [103] which is not available at the time of writing.

Max-norm Whereas the trace-norm penalizes the average magnitude of entries in the singular value decomposition, the max-norm only penalizes the maximum row-norm of the singular factors :

$$\|M\|_{\max} = \inf_{M=UV^\top} \left(\max_i \|U_i\|_2\right)\left(\max_i \|V_i\|_2\right)$$

A matrix of bounded max-norm 1 has a natural interpretation. One can see the rows of U as points to classify on the hypersphere, and interpret the rows of V as normal vectors to the classifying hyperplanes. Srebro et al. [113] provides the following PAC-bound for matrices of low max-norm:

Theorem 2. *[113] For all targer matrices* $Y \in \{\pm 1\}^{n\times m}$ *and sample sizes* $|S| > n\log n$*, and for a uniformly selected sample* S *of* $|S|$ *entries in* Y*, with probability* $1-\delta$ *over the sample selection, the following holds for all matrices* $X \in \mathbb{R}^{n\times m}$ *and all*

$\gamma > 0$*:*

$$\mathcal{D}^{\pm}(X,Y) \leq \mathcal{D}_S^{\gamma}(X;Y) + \tilde{\mathcal{O}}\left(\frac{\|X\|_{max}}{\gamma}\sqrt{\frac{(n+m)}{|S|}}\right) + \sqrt{\frac{\ln(4/\delta)}{2|S|}}$$

where $\tilde{\mathcal{O}}$ *is the soft-O notation ignoring log-factors in* n*,* m*,* γ *and* $\|X\|_*$*.*

Noting that the maximum is greater than the average shows $\|X\|_*/\sqrt{nm} \leq \|X\|_{\max}$, which explains the difference in scaling between the PAC-bounds for each norm. Generalizing this bound to tensors of arbitrary order can be done *via* multilinear extensions of the Grothendieck inequality such as Equation 12.67 in Blei [11].

Max-norm vs Trace-norm : sampling distributions Uniform sampling of entries is unlikely to hold in many link prediction settings. For example, different users will have different propensities to rate movies online, or some entities in a knowledge base will have more of their positive edge labels than others due to some public interest around them. The bounds in Theorem 1 and Theorem 2 only hold for uniformly selected samples. However, the former generalizes to samples from arbitrary distributions, for a generalization error measured accordingly:

$$\mathcal{D}_{\mathcal{D}}^{\pm}(X,Y) = \mathbb{E}_{\mathcal{D}}[\text{loss}(X_{i,j};Y_{i,j})].$$

In order to obtain bounds for non-uniform samples with the trace-norm, weighting is required [110, 37]. The weighted trace-norm is defined for a sampling distribution $\mathcal{D}$ with marginals over rows p_r and columns over p_c as:

$$\|X\|_{*,\mathcal{D}} = \|\text{Diag}(p_r)^{1/2} X \text{Diag}(p_c)^{1/2}\|_*. \tag{2.7}$$

Bounding the weighted trace-norm with the empirical (smoothed) marginals allows Foygel et al. [37] to bound the generalization error under arbitrary sampling distribution. Interestingly, practical penalties on the weighted trace-norm are obtained

naturally with stochastic gradient descent formulations of the form:

$$\min_{\hat{X}=UV^\top} \frac{1}{|S|} \sum_{i,j \in S} \text{loss}(X_{i,j}, \hat{X}_{i,j}) + \lambda \left(\|U_i\|_2^2 + \|V_j\|_2^2 \right).$$

This formulation was used by the winning entry to the Netflix Prize [66]. A weighting according to the empirical row and column marginals naturally arises from this formulation and is equivalent to a penalty on the weighted trace-norm as defined in Equation (2.7).

Assuming that results qualitatively similar to the results in this section hold in our setting, we use the same weighting of the norm in Chapters 3 and 5.

2.5.2 Tensor norms and guarantees

The results we discussed for matrices are enabled by powerful tools in linear algebra and spectral theory for random matrices. In this section, we describe available results for tensors. Moving into the multi-linear world, we lose many of these tools. We first discuss how this was circumvented by working with matricized tensors [78, 105, 39, 116] to obtain computable convex penalties using previous work on matrices. We then describe results for tensor recovery using the NP-hard tensor nuclear-norm [127], which generalizes matrix recovery results and obtains better sample complexity than the penalties based on matricization.

Matricization based norms Given a tensor X of order K, with $X_{(k)}$ the mode-k matricization (that is, the columns of $X_{(k)}$ are the mode-k fibers of X), define the norm :

$$\|X\|_{\text{SNN}} = \frac{1}{K} \sum_{k=1}^{K} \|X_{(k)}\|_*$$

Minimizing this Sum of Nuclear Norms (SNN) leads to tractable convex programs over tensors, which explains its success and studies [78, 105, 39, 116]. However, the recovery guarantees for this norm ($|S|$ $\mathcal{O}(rn^{K-1})$) are only as good as the requirements of recovery for a tensor that is only low-rank along one mode [87]. This is due to

matricization leading to the recovery of r singular vectors of size n^{K-1}. Another convex penalty has been suggested and its recovery guarantees studied [79], but it also requires measurements of the order of n^{K-1}, which is impractical in our case.

Tensor nuclear norm The spectral p-norm for tensors (of order 3 for notational simplicity) can be defined similarly to the matrix spectral norm :

$$\|X\|_p = \max_{\|u_1\|_p=\|u_2\|_p=\|u_3\|_p=1} \langle X, u_1 \otimes u_2 \otimes u_3 \rangle \qquad \text{where } \langle X, Y \rangle = \sum_{i,j,k} X_{i,j,k} Y_{i,j,k}.$$

The dual-norm to this norm is the tensor nuclear p-norm : $\|\cdot\|_{p*}$ which can also be defined as an atomic norm over pure tensors [38] :

$$\|X\|_{p*} = \inf_R \inf_{\sigma,U,V,W} \Big\{ \|\sigma\|_1 \;\Big|\; X = \sum_{r=1}^{R} \sigma_r U_{:,r} \otimes V_{:,r} \otimes W_{:,r}, \\ \forall r \; \|U_{:,r}\|_p = \|V_{:,r}\|_p = \|W_{:,r}\|_p = 1 \Big\}.$$

As shown in Yuan and Zhang [127], the recovery requirements when minimizing the nuclear 2 norm for tensors $X \in \mathbb{R}^{n \times n \times n}$ of low-coherence are of order $\mathcal{O}(r(n \log n)^{3/2})$ (where the CP-rank of X is in $[r, r^2]$). This result is more attractive than the guarantees for matricization based norms, since half an order is gained on the larger dimensions n (compare 3/2 to 2). However, the tensor nuclear norm is not computationally tractable [38]. Approximation schemes [26] have been developped only for minimizing the nuclear norm under a squared loss data-fitting error. In Chapter 3,4 and 5, we use non-convex variational forms of the nuclear p-norm with good experimental results.

Chapter 3

Canonical Tensor Parametrization for Knowledge Base Completion

> In the midst of chaos, there is also opportunity.
>
> Sun Tzu

The problem of Knowledge Base Completion can be framed as a 3rd-order binary tensor completion problem. In this light, the CP parametrization (CP) [50] seems like a natural solution; however, current implementations of CP on standard Knowledge Base Completion benchmarks are lagging behind their competitors. In this work, we attempt to understand the limits of CP for knowledge base completion. First, we motivate and test a novel regularizer, based on tensor nuclear p-norms. Then, we present a reformulation of the problem that makes it invariant to arbitrary choices in the inclusion of predicates or their reciprocals in the dataset. These two methods combined allow us to beat the current state of the art on several datasets with a CP parametrization, and obtain even better results using the more advanced ComplEx model.

3.1 Introduction

In knowledge base completion, the learner is given triples (subject, predicate, object) of facts about the world, and has to infer new triples that are likely but not yet known to be true. This problem has attracted a lot of attention [92, 89] both as an example application of large-scale tensor factorization, and as a benchmark of learning representations of relational data.

The standard completion task is link prediction, which consists in answering queries (subject, predicate, ?) or (?, predicate, object). In that context, the low-rank tensor parametrization (also called CANDECOMP/PARAFAC or CP) [53] is known to perform poorly compared to more specialized methods. For instance, DistMult [126], a particular case of CP which shares the factors for the subject and object modes, was recently shown to have state-of-the-art results [60]. This result is surprising because DistMult learns a tensor that is symmetric in the subject and object modes, while the datasets contain mostly non-symmetric predicates.

The goal of this paper is to study whether and how CP can perform as well as its competitors. To that end, we evaluate three possibilities.

First, as Kadlec et al. [60] showed that performances for these tasks are sensitive to the loss function and optimization parameters, we re-evaluate CP with a broader parameter search and a multiclass log-loss.

Second, since the best performing approaches are less expressive than CP, we evaluate whether regularization helps. On this subject, we show that the standard regularization used in knowledge base completion does not correspond to regularization with a tensor norm. We then propose to use tensor nuclear p-norms [38], with the goal of designing more principled regularizers.

Third, we propose a different formulation of the objective, in which we model separately predicates and their inverse: for each predicate pred, we create an inverse predicate pred^{-1} and create a triple $(\text{obj}, \text{pred}^{-1}, \text{sub})$ for each training triple $(\text{sub}, \text{pred}, \text{obj})$. At test time, queries of the form $(?, \text{pred}, \text{obj})$ are answered as $(\text{obj}, \text{pred}^{-1}, ?)$. Similar formulations were previously used by Shen et al. [104] and

Joulin et al. [59], but for different models for which there was no clear alternative, so the impact of this reformulation has never been evaluated.

To assess whether the results we obtain are specific to CP, we also carry on the same experiments with a state-of-the-art model, ComplEx [118]. ComplEx has the same expressivity as CP in the sense that it can represent any tensor, but it implements a specific form of parameter sharing. We perform all our experiments on 5 common benchmark datasets of link prediction in knowledge bases.

Our results first confirm that within a reasonable time budget, the performance of both CP and ComplEx are highly dependent on optimization parameters. With systematic parameter searches, we obtain better results for ComplEx than what was previously reported, confirming its status as a state-of-the-art model on all datasets. For CP, the results are still way below its competitors.

Learning and predicting with the inverse predicates, however, changes the picture entirely. First, with both CP and ComplEx, we obtain significant gains in performance on all the datasets. More precisely, we obtain state-of-the-art results with CP, matching those of ComplEx. For instance, on the benchmark dataset FB15K [13], the mean reciprocal rank of vanilla CP and vanilla ComplEx are 0.40 and 0.80 respectively, and it grows to 0.86 for both approaches when modeling the inverse predicates.

Finally, the new regularizer we propose based on the nuclear 3-norm, does not dramatically help CP, which leads us to believe that a careful choice of regularization is not crucial for these CP models. Yet, for both CP and ComplEx with inverse predicates, it yields small but significant improvements on the more difficult datasets.

3.2 Tensor Factorization of Knowledge Bases

We describe in this section the formal framework we consider for knowledge base completion and more generally link prediction in relational data, the learning criteria, as well as the approaches that we will discuss.

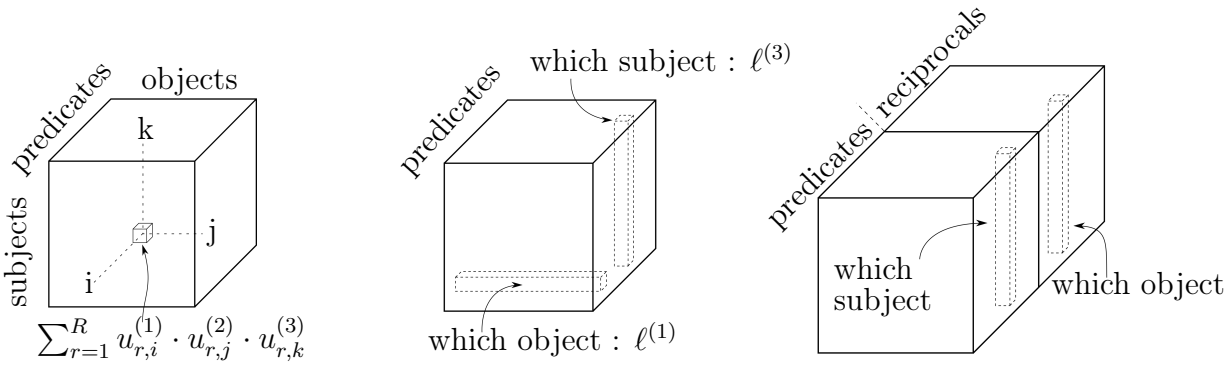


Figure 3-1: (a) On the left, the link between the score of a triple (i,j,k) and the tensor estimated via CP. (b) In the middle, the two type of fiber losses that we will consider. (c) On the right, our semantically invariant reformulation, the first-mode fibers become third-mode fibers of the reciprocal half of the tensor.

3.2.1 Link Prediction in Relational Data

We consider relational data that comes in the form of triples (subject, predicate, object), where the subject and the object are from the same set of entities. In knowledge bases, these triples represent facts about entities of the world, such as $(Washington, capital\_of, USA)$. A training set $\mathcal{S}$ contains triples of indices $\mathcal{S} = \{(i_1, j_1, k_1), ..., (i_{|\mathcal{S}|}, j_{|\mathcal{S}|}, k_{|\mathcal{S}|})\}$ that represent predicates that are known to hold. The validation and test sets contain queries of the form $(?, j, k)$ and $(i, j, ?)$, created from triples (i, j, k) that are known to hold but held-out from the training set. To give orders of magnitude, the largest datasets we experiment on, FB15K and YAGO3-10, contain respectively $15k/1.3k$ and $123k/37$ entities/predicates.

3.2.2 Tensor Factorization for Link Prediction

Relational data can be represented as a $\{0, 1\}$-valued third order tensor $\boldsymbol{Y} \in \{0, 1\}^{N \times L \times N}$, where N is the total number of entities and L the number of predicates, with $\boldsymbol{Y}_{i,j,k} = 1$ if the relation (i, j, k) is known. In the rest of the paper, the three modes will be called the subject mode, the predicate mode and the object mode respectively. Tensor factorization algorithms can thus be used to infer a predicted tensor $\hat{\boldsymbol{X}} \in \mathbb{R}^{N \times L \times N}$ that approximates $\boldsymbol{Y}$ in a sense that we describe in the next subsection. Validation/test queries $(?, j, k)$ are answered by ordering entities i' by decreasing values of $\hat{\boldsymbol{X}}_{i',j,k}$, whereas queries $(i, j, ?)$ are answered by ordering entities k' by decreasing values of

$\hat{\boldsymbol{X}}_{i,j,k'}$.

Several approaches have considered link prediction as a low-rank tensor learning problem. These models then differ only by structural constraints on the learned tensor. Three models of interest are:

CP. The low-rank parametrization, also called CANDECOMP/PARAFAC [53], represents a tensor $\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ as a sum of R rank one tensors $u_r^{(1)} \otimes u_r^{(2)} \otimes u_r^{(3)}$ (with $\otimes$ the tensor product) where $r \in \{1, ..., R\}$, and $u_r^{(m)} \in \mathbb{R}^{N_m}$:

$$\boldsymbol{X} = \sum_{r=1}^{R} u_r^{(1)} \otimes u_r^{(2)} \otimes u_r^{(3)} .$$

A representation of this parametrization, and the score of a specific triple is given in Figure 3-1 (a). Given $\boldsymbol{X}$, the smallest R for which this decomposition holds is called the canonical rank of $\boldsymbol{X}$.

DistMult. In the more specific context of link prediction, it has been suggested in Bordes et al. [12], Nickel et al. [91] that since both subject and object mode represent the same entities, they should have the same factors. DistMult [126] is a version of CP with this additional constraint. It represents a tensor $\boldsymbol{X} \in \mathbb{R}^{N \times L \times N}$ as a sum of rank-1 tensors $u_r^{(1)} \otimes u_r^{(2)} \otimes u_r^{(1)}$:

$$\boldsymbol{X} = \sum_{r=1}^{R} u_r^{(1)} \otimes u_r^{(2)} \otimes u_r^{(1)} .$$

ComplEx. By contrast with the first models that proposed to share the subject and object mode factors, DistMult yields a tensor that is symmetric in the object and subject modes. The assumption that the data tensor can be properly approximated by a symmetric tensor for Knowledge base completion is not satisfied in many practical cases (e.g., while $(Washington, capital\_of, USA)$ holds, $(USA, capital\_of, Washington)$ does not). ComplEx [118] proposes an alternative where the subject and object modes share the parameters of the factors, but are complex conjugate of each other. More precisely, this approach represents a real-valued

tensor $\boldsymbol{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ as the real part of a sum of R complex-valued rank one tensors $u_r^{(1)} \otimes u_r^{(2)} \otimes \overline{u}_r^{(1)}$ where $r \in \{1, ..., R\}$, and $u_r^{(m)} \in \mathbb{C}^{N_m}$

$$\boldsymbol{X} = \mathrm{Re}\big(\sum_{r=1}^{R} u_r^{(1)} \otimes u_r^{(2)} \otimes \overline{u}_r^{(1)} \big) \,,$$

where $\overline{u}_r^{(1)}$ is the complex conjugate of $u_r^{(1)}$. This parametrization can represent any real tensor [118].

The good performances of DistMult on notoriously non-symmetric datasets such as FB15K or WN18 are surprising. First, let us note that for the symmetricity to become an issue, one would have to evaluate queries $(i, j, ?)$ while also trying to answer correctly to queries of the form $(?, j, i)$ for a non-symmetric predicate j. The ranking for these two queries would be identical, and thus, we can expect issues with relations such as $capital\_of$. In FB15K, those type of problematic queries make up only 4% of the test set and thus, have a small impact. On WN18 however, they make up 60% of the test set. We describe in appendix 8.1 a simple strategy for DistMult to have a high filtered MRR on the hierarchical predicates of WN18 despite its symmetricity assumption.

3.2.3 Training

Previous work suggested ranking losses [13], binary logistic regression [118] or sampled multiclass log-loss [60]. Motivated by the solid results in Joulin et al. [59], our own experimental results, and with a satisfactory speed of about two minutes per epoch on FB15K, we decided to use the full multiclass log-loss.

Given a training triple (i, j, k) and a predicted tensor $\boldsymbol{X}$, the instantaneous multi-

class log-loss $\text{loss}_{i,j,k}(\boldsymbol{X})$ is

$$\begin{aligned}
\text{loss}_{i,j,k}(\boldsymbol{X}) &= \text{loss}^{(1)}_{i,j,k}(\boldsymbol{X}) + \text{loss}^{(3)}_{i,j,k}(\boldsymbol{X}) \\
\text{loss}^{(1)}_{i,j,k}(\boldsymbol{X}) &= -\boldsymbol{X}_{i,j,k} + \log\big(\sum_{k'} \exp(\boldsymbol{X}_{i,j,k'})\big) \\
\text{loss}^{(3)}_{i,j,k}(\boldsymbol{X}) &= -\boldsymbol{X}_{i,j,k} + \log\big(\sum_{i'} \exp(\boldsymbol{X}_{i',j,k})\big)\,.
\end{aligned} \tag{3.1}$$

These two partial losses are represented in Figure 3-1 (b). For CP, the final tensor is computed by finding a minimizer of a regularized empirical risk formulation, where the factors $u_r^{(d)}$ are weighted in a data-dependent manner by $w_{\mathcal{S}}^{(d)}$, which we describe below:

$$\begin{aligned}
\min_{\substack{(u_r^{(d)})_{\substack{d=1..3\\r=1..R}}}} \sum_{(i,j,k)\in\mathcal{S}} \text{loss}_{i,j,k}\Big(\sum_{r=1}^{R} u_r^{(1)} \otimes u_r^{(2)} \otimes u_r^{(3)}\Big) \\
+ \lambda \sum_{r=1}^{R}\sum_{d=1}^{3} \|w_{\mathcal{S}}^{(d)} \odot u_r^{(d)}\|_2^2\,,
\end{aligned} \tag{3.2}$$

where $\odot$ is the entry-wise multiplication of vectors. For DistMult and ComplEx, the learning objective is similar, up to the appropriate parameter sharing and computation of the tensor.

As discussed in Section 3.3.2, the weights $w_{\mathcal{S}}^{(d)}$ may improve performances when some rows/columns are sampled more than others. They appear naturally in optimization with stochastic gradient descent when the regularizer is applied only to the parameters that are involved in the computation of the instantaneous loss. For instance, in the case of the logistic loss with negative sampling used by Trouillon et al. [118], denoting by q_i^d the marginal probability (over $\mathcal{S}$) that index i appears in mode d of a data triple, these weights are $w_{\mathcal{S},i}^{(d)} = \sqrt{q_i^d + \alpha}$ for some $\alpha > 0$ that depends on the negative sampling scheme.

We focus on redefining the loss (3.1) and the regularizer (3.2).

3.3 Related Work

We discuss here in more details the work that has been done on link prediction in relational data and on regularizers for tensor completion.

3.3.1 Link Prediction in Relational Data

There has been extensive research on link prediction in relational data, especially in knowledge bases, and we review here only the prior work that is most relevant to this paper. While some approaches explicitly use the graph structure during inference [71], we focus here on representation learning and tensor factorization methods, which are the state-of-the-art on the benchmark datasets we use. We also restrict the discussion to approaches that only use relational information, even though some approaches have been proposed to leverage additional types [67, 80] or external word embeddings [117].

We can divide the first type of approaches into two broad categories. First, two-way approaches score a triple (i, j, k) depending only on bigram interaction terms of the form subject-object, subject-predicate, and predicate-object. Even though they are tensor approximation algorithms of limited expressivity, two-way models based on translations TransE, or on bag-of-word representations [59] have proved competitive on many benchmarks. Yet, methods using three-way multiplicative interactions, as described in the previous section, show the strongest performances [12, 40, 93, 118]. Compared to general-purpose tensor factorization methods such as CP, a common feature of these approaches is to share parameters between objects and subjects modes [91], an idea that has been widely accepted except for the two-way model of Joulin et al. [59]. DistMult [126] is the extreme case of this parameter sharing, in which the predicted tensor is symmetric in the subject and object modes.

3.3.2 Regularization for Matrix Completion

Norm-based regularization has been extensively studied in the context of matrix completion. The trace norm (or nuclear norm) has been proposed as a convex relaxation of the rank [114] for matrix completion in the setting of rating prediction,

with strong theoretical guarantees [21]. While efficient algorithms to solve the convex problems have been proposed [see e.g. 20, 55], the practice is still to use the matrix equivalent of the nonconvex formulation (3.2). For the trace norm (nuclear 2-norm), in the matrix case, the regularizer simply becomes the squared 2-norm of the factors and lends itself to alternating methods or SGD optimization [98, 66]. When the samples are not taken uniformly at random from a matrix, some other norms are preferable to the usual nuclear norm. The weighted trace norm reweights elements of the factors based on the marginal rows and columns sampling probabilities, which can improve sample complexity bounds when sampling is non-uniform [37, 88]. Direct SGD implementations on the nonconvex formulation implicitly take this reweighting rule into account and were used by the winners of the Netflix challenge [see 110, Section 5].

3.3.3 Tensor Completion and Parametrizations

There is a large body of literature on low-rank tensor parametrizations [see 64, for a comprehensive review]. Closely related to our work is the low-rank decomposition of a tensor (also called CANDECOMP/PARAFAC or CP) [53], which solves a problem similar to (3.5) without the regularization (i.e., $\lambda = 0$), and usually the square loss.

Several norm-based regularizations for tensors have been proposed. Some are based on unfolding a tensor along each of its modes to obtain matricizations, and either regularize by the sum of trace norms of the matricizations [116] or write the original tensor as a sum of tensors T_k, regularizing their respective kth matricizations with the trace norm [124]. However, in the large-scale setting, even rank-1 approximations of matricizations involve too many parameters to be tractable.

Recently, the tensor trace norm (nuclear 2-norm) was proposed as a regularizer for tensor completion Yuan and Zhang [127], and an algorithm based on the generalized conditional gradient has been developed by Cheng et al. [26]. This algorithm requires, in an inner loop, to compute a (constrained) rank-1 tensor that has largest dot-product with the gradient of the data-fitting term (gradient w.r.t. the tensor argument). This algorithm is efficient in our setup only with the square error loss (instead of the

multiclass log-loss), because the gradient is then a low-rank + sparse tensor when the argument is low-rank. However, on large-scale knowledge bases, the state of the art is to use a binary log-loss or a multiclass log-loss [118, 60]; in that case, the gradient is not adequately structured, thereby causing the approach of [26] to be too computationally costly.

3.4 Nuclear p-Norm Regularization

As discussed in Section 3.3, norm-based regularizers have proved useful for matrices. We aim to reproduce these successes with tensor norms. We use the nuclear p-norms defined by Friedland and Lim [38]. As shown in Equation (3.2), the community has favored so far a regularizer based on the square Frobenius norms of the factors [126, 118]. We first show that the unweighted version of this regularizer is not a tensor norm. Then, we propose a variational form of the nuclear 3-norm to replace the usual regularization at no additional computational cost when used with SGD. Finally, we discuss a weighting scheme analogous to the weighted trace-norm proposed in Srebro and Salakhutdinov [110].

3.4.1 From Matrix Trace-Norm to Tensor Nuclear Norms

To simplify notation, let us introduce the set of CP parametrizations of a tensor $\boldsymbol{X}$ of rank at most R:

$$\mathcal{U}_R(\boldsymbol{X}) = \Big\{ (u_r^{(d)})_{\substack{d=1..3 \\ r=1..R}} \;\Big|\; \boldsymbol{X} = \sum_{r=1}^{R} u_r^{(1)} \otimes u_r^{(2)} \otimes u_r^{(3)}, \forall r, d,\ u_r^{(d)} \in \mathbb{R}^{N_d} \Big\}.$$

We will study the family of regularizers defined for $u \in \mathcal{U}_R(\boldsymbol{X})$ by:

$$\Omega_p^\alpha(u) = \frac{1}{3} \sum_{r=1}^{R} \sum_{d=1}^{3} \|u_r^{(d)}\|_p^\alpha.$$

Note that with $p = \alpha = 2$, we recover the familiar squared Frobenius norm regularizer used in (3.2). Similar to showing that the squared Frobenius norm is a *variational*

form of the trace norm on matrices (i.e., its minimizers realize the trace norm, $\inf_{M=UV^T} \frac{1}{2}(\|U\|_F^2+\|V\|_F^2) = \|M\|_*$), we start with a technical lemma that links our regularizer with a function on the spectrum of our parametrizations.

Lemma 1.

$$\min_{u\in\mathcal{U}_R(\boldsymbol{X})} \frac{1}{3}\sum_{r=1}^{R}\sum_{d=1}^{3}\|u_r^{(d)}\|_p^{\alpha} = \min_{u\in\mathcal{U}_R(\boldsymbol{X})} \sum_{r=1}^{R}\prod_{d=1}^{3}\|u_r^{(d)}\|_p^{\alpha/3}. \tag{3.3}$$

Moreover, the minimizers of the left-hand side satisfy:

$$\|u_r^{(d)}\|_p = \sqrt[3]{\prod_{d'=1}^{3}\|u_r^{(d')}\|_p}.$$

Proof. See Appendix A.2. □

This Lemma motivates the introduction of the set of p-norm normalized tensor parametrizations:

$$\overline{\mathcal{U}}_R^p(\boldsymbol{X}) = \Big\{(\sigma_r,(\tilde{u}_r))_{r=1..R} \;\Big|\; \sigma_r = \prod_{d=1}^{3}\|u_r^{(d)}\|_p, \tilde{u}_r^{(d)} = \frac{u_r^{(d)}}{\|u_r^{(d)}\|_p}, \forall r,d,u\in\mathcal{U}_R(\boldsymbol{X})\Big\}.$$

Lemma (3.3), shows that Ω_p^{α} behaves as an $\ell_{\alpha/D}$ penalty over the CP *spectrum* for tensors of order D. We recover the nuclear norm for matrices when $\alpha = p = 2$.

Using Lemma (3.3), we have :

$$\min_{u\in\mathcal{U}_R(\boldsymbol{X})} \Omega_2^2(u) \leq \eta \iff \min_{(\sigma,\tilde{u})\in\overline{\mathcal{U}}_R^2(\boldsymbol{X})} \|\sigma\|_{2/3} \leq \eta^{3/2} \tag{3.4}$$

We show that the sub-level sets of the term on the right are not convex, which implies that Ω_2^2 is not the variational form of a tensor norm, and hence, is not the tensor analog to the matrix trace norm.

Proposition 2. *The function over third order-tensors of* $\mathbb{R}^{N_1\times N_2\times N_3}$ *defined as*

$$|||\boldsymbol{X}||| = \min\Big\{\|\sigma\|_{2/3} \;\Big|\; (\sigma,\tilde{u})\in\overline{\mathcal{U}}_R^2(\boldsymbol{X}), R\in\mathbb{N}\Big\}$$

is not convex.

Proof. See Appendix A.2. □

Remark 1. *Cheng et al. [26, Appendix D] already showed that regularizing with the square Frobenius norm of the factors is not related to the trace norm for tensors of order* 3 *and above, but their observation is that the regularizer is not positively homogeneous, i.e.,* $\min_{u\in\alpha\mathcal{U}_R(\boldsymbol{X})}\Omega_2^2(u) \neq |\alpha|\min_{u\in\mathcal{U}_R(\boldsymbol{X})}\Omega_2^2(u)$. *Our result in Proposition 2 is stronger in that we show that this regularizer is not a norm even after the rescaling* (3.4) *to make it homogeneous.*

The nuclear p-norm of $\boldsymbol{X} \in \mathbb{R}^{N_1\times N_2\times N_3}$ for $p \in [1, +\infty]$, is defined in Friedland and Lim [38] as

$$\|\boldsymbol{X}\|_{*,p} := \min\left\{\|\sigma\|_1 \;\middle|\; (\sigma,\tilde{u}) \in \overline{\mathcal{U}}_R^p(\boldsymbol{X}), R\in\mathbb{N}\right\}.$$

Given an estimated upper bound on the optimal R, the original problem (3.2) can then be re-written as a non-convex problem using the equivalence in Lemma (3.3):

$$\min_{\substack{(u_r^{(d)})_{d=1..3}\\ r=1..R}} \sum_{(i,j,k)\in\mathcal{S}} \text{loss}_{i,j,k}\Big(\sum_{r=1}^{R} u_r^{(1)}\otimes u_r^{(2)}\otimes u_r^{(3)}\Big) + \frac{\lambda}{3}\sum_{r=1}^{R}\sum_{d=1}^{3}\|u_r^{(d)}\|_p^3 . \tag{3.5}$$

This variational form suggests to use $p = 3$, as a means to make the regularizer separable in each coefficients, given that then $\|u_r^{(d)}\|_p^3 = \sum_{i=1}^{n_d}\left|u_{r,i}^{(d)}\right|^3$.

3.4.2 Weighted Nuclear p-Norm

Similar to the weighted trace-norm for matrices, the weighted nuclear 3-norm can be easily implemented by keeping the regularization terms corresponding to the sampled triplets only, as discussed in Section 3.3.2. This leads to a formulation of the form

$$\min_{\substack{(u_r^{(d)})_{d=1..3}\\ r=1..R}} \sum_{(i,j,k)\in\mathcal{S}} \Big[\text{loss}_{i,j,k}\big(\sum_{r=1}^{R} u_r^{(1)}\otimes u_r^{(2)}\otimes u_r^{(3)}\big) + \frac{\lambda}{3}\sum_{r=1}^{R}\Big(\left|u_{r,i}^{(1)}\right|^3 + \left|u_{r,j}^{(2)}\right|^3 + \left|u_{r,k}^{(3)}\right|^3\Big)\Big].$$

For an example (i, j, k), only the parameters involved in the computation of $\hat{X}_{i,j,k}$ are regularized. The computational complexity is thus the same as the currently

used weighted Frobenius norm regularizer. With $q^{(1)}$ (resp. $q^{(2)}$, $q^{(3)}$) the marginal probabilities of sampling a subject (resp. predicate, object), the weighting implied by this regularization scheme is

$$\|\boldsymbol{X}\|_{*,3,w} = \|(\sqrt[3]{q^{(1)}} \otimes \sqrt[3]{q^{(2)}} \otimes \sqrt[3]{q^{(3)}}) \odot \boldsymbol{X}\|_{*,3}$$

We justify this weighting only by analogy with the matrix case discussed by [110]: to make the weighted nuclear 3-norm of the all 1 tensor independent of its dimensions for a uniform sampling (since the nuclear 3-norm grows as $\sqrt[3]{MNP}$ for an (M, N, P) tensor).

Comparatively, for the weighted version of the nuclear 2-norm analyzed in Yuan and Zhang [127], the nuclear 2-norm of the all 1 tensor scales like $\sqrt{NMP}$. This would imply a formulation of the form

$$\min_{\substack{(u_r^{(d)})_{\substack{d=1..3\\r=1..R}}}} \sum_{(i,j,k)\in\mathcal{S}} \text{loss}_{i,j,k}\Big(\sum_{r=1}^{R} u_r^{(1)} \otimes u_r^{(2)} \otimes u_r^{(3)}\Big) + \frac{\lambda}{3}\sum_{r=1}^{R}\sum_{d=1}^{3} \|\sqrt{q^{(d)}} \odot u_r^{(d)}\|_2^3 \,.$$

Contrary to formulation (3.4.2), the optimization of formulation (3.4.2) with a minibatch SGD leads to an update of every coefficients for each mini-batch considered. Depending on the implementation, and size of the factors, there might be a large difference in speed between the updates of the weighted nuclear $\{2, 3\}$-norm. In our implementation, this difference for CP is of about $1.5\times$ in favor of the nuclear 3-norm on FB15K.

3.5 A New CP Objective

Since our evaluation objective is to rank either the left-hand side or right-hand side of the predicates in our dataset, what we are trying to achieve is to model both predicates and their reciprocal. This suggests appending to our input the reciprocals of each predicates, thus factorizing $[\boldsymbol{Y};_2 \tilde{\boldsymbol{Y}}]$ rather than $\boldsymbol{Y}$, where $[\ ;_2\ ]$ is the mode-2 concatenation, and $\boldsymbol{Y}_{i,j,k} = \tilde{\boldsymbol{Y}}_{k,j,i}$. After that, we only need to model the object fibers

Dataset	N	P	Train	Valid	Test
WN18	41k	18	141k	5k	5k
WN18RR	41k	11	87k	3k	3k
FB15K	15k	1k	500k	50k	60k
FB15K-237	15k	237	272k	18k	20k
YAGO3-10	123k	37	1M	5k	5k

Table 3.1: Dataset statistics.

of this new tensor $\boldsymbol{Y}$. We represent this transformation in Figure 3-1 (c). This reformulation has an important side-effect: it makes our algorithm invariant to the arbitrary choice of including a predicate or its reciprocal in the dataset. This property was introduced as "Semantic Invariance" in Bailly et al. [5]. Another way of achieving this invariance property would be to find the flipping of predicates that lead to the smallest model. In the case of a CP parametrization, we would try to find the flipping that leads to lowest tensor rank. This seems hopeless, given the NP-hardness of computing the tensor rank.

More precisely, the instantaneous loss of a training triple (i, j, k) becomes :

$$\begin{aligned} \text{loss}_{i,j,k}(\boldsymbol{X}) = &- \boldsymbol{X}_{i,j,k} + \log\big(\sum_{k'} \exp(\boldsymbol{X}_{i,j,k'})\big) \\ &- \boldsymbol{X}_{k,j+P,i} + \log\big(\sum_{i'} \exp(\boldsymbol{X}_{k,j+P,i'})\big). \end{aligned} \tag{3.6}$$

At test time we use $\hat{\boldsymbol{X}}_{i,j,:}$ to rank possible right hand sides for query $(i, j, ?)$ and $\hat{\boldsymbol{X}}_{k,j+P,:}$ to rank possible left hand sides for query $(?, j, k)$.

Using CP to factor the tensor described in (3.6), we beat the previous state of the art on many benchmarks, as shown in Table 3.2. This reformulation seems to help even the ComplEx parametrization, for which parameters are shared between the entity embeddings of the first and third mode.

	Model	WN18		WN18RR		FB15K		FB15K-237		YAGO3-10	
		MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10	MRR	H@10
Past SOTA	CP	0.08	0.13	-	-	0.33	0.53	-	-	-	-
	ComplEx †	0.94	0.95	0.44	0.51	0.70	0.84	0.25	0.43	0.36	0.55
	DistMult *	0.82	0.94	0.43	0.49	0.80	0.89	0.24	0.42	0.34	0.54
	ConvE *	0.94	0.95	0.46	0.48	0.75	0.87	0.32	0.49	0.52	0.66
	Best Published⋆	0.94	**0.97**	0.46	0.51	0.84	**0.93**	0.32	0.49	0.52	0.66
Standard	CP-N3	0.20	0.33	0.12	0.20	0.46	0.65	0.33	0.51	0.38	0.65
	ComplEx-N3	**0.95**	**0.96**	0.47	0.54	0.80	0.89	0.35	0.54	0.49	0.68
Reciprocal	CP-FRO	**0.95**	0.95	0.46	0.48	**0.86**	0.91	0.34	0.51	0.54	0.68
	CP-N3	**0.95**	0.96	0.47	0.54	**0.86**	0.91	0.36	0.54	0.57	**0.71**
	ComplEx-FRO	**0.95**	0.96	0.47	0.54	**0.86**	0.91	0.35	0.53	0.57	**0.71**
	ComplEx-N3	**0.95**	0.96	**0.48**	**0.57**	**0.86**	0.91	**0.37**	**0.56**	**0.58**	**0.71**

Table 3.2: *Results taken as best from Dettmers et al. [30] and Kadlec et al. [60]. †Results taken as best from Dettmers et al. [30] and Trouillon et al. [118].⋆ We give the origin of each result on the Best Published row in appendix.

3.6 Experiments

We conducted all experiments on a Quadro GP 100 GPU. The code is available at `https://github.com/facebookresearch/kbc`.

3.6.1 Datasets and Experimental Setup

WN18 and FB15K are popular benchmarks in the Knowledge Base Completion community. The former comes from the WordNet database, was introduced in Bordes et al. [14] and describes relations between words. The most frequent types of relations are highly hierarchical (e.g., hypernym, hyponym). The latter is a subsampling of Freebase limited to 15k entities, introduced in Bordes et al. [13]. It contains predicates with different characteristics (e.g., one-to-one relations such as *capital_of* to many-to-many such as *actor_in_film*).

Toutanova and Chen [117] and Dettmers et al. [30] identified train to test leakage in both these datasets, in the form of test triplets, present in the train set for the

reciprocal predicates. Thus, both of these authors created two modified datasets: FB15K-237 and WN18RR. These datasets are harder to fit, so we expect regularization to have more impact. Dettmers et al. [30] also introduced the dataset YAGO3-10, which is larger in scale and doesn't suffer from leakage. All datasets statistics are shown in Table 3.1.

In all our experiments, we distinguish two settings: Reciprocal, in which we use the loss described in equation (3.6) and Standard, which uses the loss in equation (3.1). We compare our implementation of CP and ComplEx with the best published results, then the different performances between the two settings, and finally, the contribution of the regularizer in the reciprocal setting. In the Reciprocal setting, we compare the weighted nuclear 3-norm (N3) against the regularizer described in (3.2) (FRO). In preliminary experiments, the weighted nuclear 2-norm described in (3.4.2) did not seem to perform better than N3 and was slightly slower. We used Adagrad [32] as our optimizer, whereas Kadlec et al. [60] favored Adam [63], because preliminary experiments didn't show improvements.

We ran the same grid for all algorithms and regularizers on the FB15K, FB15K-237, WN18, WN18RR datasets, with a rank set to 2000 for ComplEx, and 4000 for CP. Our grid consisted of two learning rates: 10^{-1} and 10^{-2}, two batch-sizes: 25 and 100, and regularization coefficients in $[0, 10^{-3}, 5.10^{-3}, 10^{-2}, 5.10^{-2}, 10^{-1}, 5.10^{-1}]$. On YAGO3-10, we limited our models to rank 1000 and used batch-sizes 500 and 3000, the rest of the grid was identical. We used the train/valid/test splits provided with these datasets and measured the filtered Mean Reciprocal Rank (MRR) and Hits@10 (Bordes et al. [13]). We used the filtered MRR on the validation set for early stopping and report the corresponding test metrics. In this setting, an epoch for ComplEx with batch-size 100 on FB15K took about $110s$ and $325s$ for a batch-size of 25. We trained for 100 epochs to ensure convergence, reported performances were reached within the first 25 epochs.

All our results are reported in Table 3.2 and will be discussed in the next subsections. Besides our implementations of CP and ComplEx, we include the results of ConvE and DistMult in the baselines. The former because Dettmers et al. [30] includes

performances on the WN18RR and YAGO3-10 benchmarks, the latter because of the good performances on FB15K of DistMult and the extensive experiments on WN18 and FB15K reported in Kadlec et al. [60]. The performances of DistMult on FB15K-237, WN18RR and YAGO3-10 may be slightly underestimated, since our baseline CP results are better. To avoid clutter, we did not include in our table of results algorithms that make use of external data such as types [67], external word embeddings [117], or using path queries as regularizers [47]. The published results corresponding to these methods are subsumed in the "Best Published" line of Table 3.2, which is taken, for every single metric and dataset, as the best published result we were able to find.

3.6.2 Reimplementation of the Baselines

The performances of our reimplementation of CP and ComplEx appear in the middle rows of Table 3.2 (Standard setting). We only kept the results for the nuclear 3-norm, which didn't seem to differ from those with the Frobenius norm. Our results are slightly better than their published counterparts, going from 0.33 to 0.46 filtered MRR on FB15K for CP and 0.70 to 0.80 for ComplEx. This might be explained in part by the fact that in the Standard setting (3.2) we use a multi-class log-loss, whereas Trouillon et al. [118] used binomial negative log-likelihood. Another reason for this increase can be the large rank of 2000 that we chose, where previously published results used a rank of around 200; the more extensive search for optimization/regularization parameters and the use of nuclear 3-norm instead of the usual regularization are also most likely part of the explanation.

3.6.3 Standard vs Reciprocal

In this section, we compare the effect of reformulation (3.6), that is, the middle and bottom rows of Table 3.2. The largest differences are obtained for CP, which becomes a state of the art contender going from 0.2 to 0.95 filtered MRR on WN18, or from 0.46 to 0.86 filtered MRR on FB15K.For ComplEx, we notice a weaker, but consistent

	1-1	m-1	1-m	m-m
CP Standard	0.45	0.71	0.24	0.44
CP Reciprocal	0.77	**0.92**	**0.71**	0.86
ComplEx Standard	0.87	**0.92**	0.59	0.81
ComplEx Reciprocal	**0.88**	**0.92**	**0.71**	**0.87**

Table 3.3: Average MRR per relation type on FB15K.

improvement by using our reformulation, with the biggest improvements observed on FB15K and YAGO3-10. Following the analysis in Bordes et al. [13], we show in Table 3.3 the average filtered MRR as a function of the degree of the predicates. We compute the average in and out degrees on the training set, and separate the predicates in 4 categories : 1-1, 1-m, m-1 and m-m, with a cut-off at 1.5 on the average degree. We include reciprocal predicates in these statistics. That is, a predicate with an average in-degree of 1.2 and average out-degree of 3.2 will count as a 1-m when we predict its right-hand side, and as an m-1 when we predict its left-hand side. Most of our improvements come from the 1-m and m-m categories, both on ComplEx and CP.

3.6.4 Frobenius vs Nuclear 3

We focus now on the effect of our norm-based N3 regularizer, compared to the Frobenius norm regularizer favored by the community. Comparing the four last rows of Table 3.2, we notice a small but consistent performance gain across datasets. The biggest improvements appear on the harder datasets WN18RR, FB15K-237 and YAGO3-10. We checked on WN18RR the significance of that gain with a Signed Rank test on the rank pairs for CP.

3.6.5 Effect of Optimization Parameters

During these experiments, we noticed a heavy influence of optimization hyper-parameters on final results. This influence can account for as much as 0.1 filtered MRR and is illustrated in Figure 3-2.

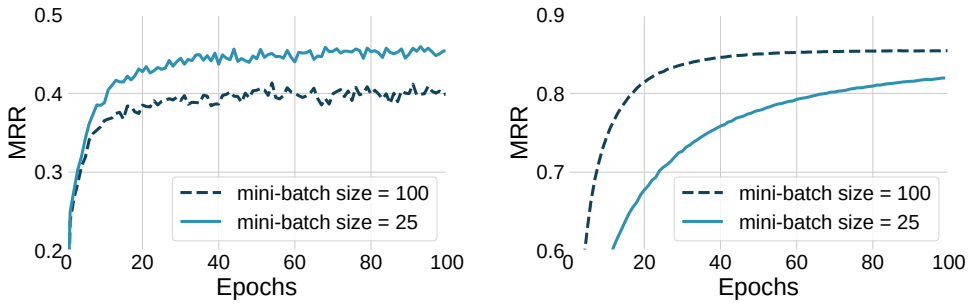


Figure 3-2: Effect of the batch-size on FB15K in the Standard (left) and Reciprocal (right) settings, other parameters being equal. The difference is large even after 100 epochs and the effect is inverted in the two settings, making it hard to choose the batch-size a priori.

3.7 Conclusion and Discussion

The main contribution of this paper is to isolate and systematically explore the effect of different factors for large-scale knowledge base completion. While the impact of optimization parameters was well known already, neither the effect of the formulation (adding reciprocals doubles the mean reciprocal rank on FB15K for CP) nor the impact of the regularization was properly assessed. The conclusion is that the CP model performs nearly as well as the competitors when each model is evaluated in its optimal configuration. We believe this observation is important to assess and prioritize directions for further research on the topic.

In addition, our proposal to use nuclear p-norm as regularizers with $p \neq 2$ for tensor factorization in general is of independent interest.

The results we present leave several questions open. Notably, whereas we give definite evidence that CP itself can perform extremely well on these datasets as long as the problem is formulated correctly, we do not have a strong theoretical justification as to why the differences in performances are so significant.

Chapter 4

Constrained Tucker parametrization

> When it is obvious that the goals cannot be reached, don't adjust the goals, adjust the action steps.
>
> Confucius

The leading approaches to tensor completion and link prediction are based on low-rank parametrization of tensors (CP). While these approaches were originally motivated by low rank approximations, the best performances are usually obtained for ranks as high as permitted by computation constraints. For large scale factorization problems where the factor dimensions have to be kept small, the performances of these approaches tend to drop drastically. The other main tensor factorization model, Tucker parametrization, is more flexible than CP for fixed factor dimensions, so we expect Tucker-based approaches to yield better performance under strong constraints on the number of parameters. However, as we show in this paper through experiments on standard benchmarks of link prediction in knowledge bases, ComplEx, [118], a variant of CP, achieves similar performances to recent approaches based on Tucker parametrization on *all* operating points in terms of number of parameters. In a control experiment we show that the adaptive optimization algorithms based on diagonal rescaling, such as Adagrad, may be too slow for practical application of Tucker parametrization to large-scale tensor completion. We present a new algorithm

for a constrained version of Tucker which implicitly applies Adagrad to a CP-based model with an additional projection of the embeddings onto a fixed lower dimensional subspace. The resulting Tucker-style extension of ComplEx obtains substantial gains on some datasets under constraints on the number of parameters.

4.1 Introduction

The problems of representation learning and link prediction in multi-relational data can be formulated as a binary tensor completion problem, where the tensor is obtained by stacking the adjacency matrices of every relations between entities. This tensor can then be intrepreted as a "knowledge base", and contains triples (subject, predicate, object) representing facts about the world. Link prediction in knowledge bases aims at automatically discovering missing facts [12, 91, 13, 92, 89].

State of the art methods use low-rank parametrization of tensors [53] or variants of it [118, 61, 70]. While initially motivated by low-rank assumptions on the underlying ground-truth tensor, the best performances are obtained by setting the rank as high as permitted by computational constraints, using tensor norms for regularization [70]. However, for large scale data where computational or memory constraints require ranks to be low [72], performances drop drastically.

Tucker parametrization is another multilinear model which allows richer interactions between entities and predicate vectors. A special case of Tucker parametrization is RESCAL [91], in which the relations are represented by matrices and entities factors are shared for subjects and objects. However, an evaluation of this model in Nickel et al. [93] shows that RESCAL lags behind other methods on several benchmarks of interest. Recent work have obtained more competitive results with similar models [7, 121], using different regularizers or deep learning heuristics such as dropout and label smoothing. Despite these recent efforts, learning Tucker parametrizations remains mostly unresolved. Wang et al. [121] does not achieve state of the art results on standard benchmarks, and we show (see Figure 4-4) that the performances reported by Balazevic et al. [7] are actually matched by ComplEx [118, 70] optimized with

Adam, which has less hyperparameters.

In this work, we overcome some of the difficulties associated with learning a Tucker model for knowledge base completion. Balazevic et al. [7] use deep-learning mechanisms such as batch normalization [54], dropout [115] or learning-rate annealing to address both regularization and optimization issues. Our approach is different: We factorize the core tensor of the Tucker parametrizations with CP to obtain a formulation which is closer to CP and better understand what difficulties appear. This yields a simple approach, which has a single regularization hyperparameter to tune for a fixed model specification.

The main novelty of our approach is a more careful application of adaptive gradient techniques. State-of-the-art methods for tensor completion use optimization algorithms with adaptive diagonal rescaling such as Adagrad [32] or Adam [63]. Through control experiments in which our model is equivalent to CP up to a fixed rotation of the embeddings, we show that one of the difficulties in training Tucker-style parametrizations can be attributed to the lack of invariance to rotation of the diagonal rescaling. Focusing on Adagrad, we propose a different update rule that is equivalent to implicitely applying Adagrad to a CP model with a projection of the embedding to a lower dimensional subspace.

Combining the Tucker formulation and the implicit Adagrad update, we obtain performances that match state-of-the-art methods on the standard benchmarks and achieve significanly better results for small embedding sizes on several datasets. Compared to the best current algorithm for Tucker parametrization of Balazevic et al. [7], our approach has less hyperparameters, and we effectively report better performances than the implementation of ComplEx of Lacroix et al. [70] in the regime of small embedding dimensions.

We discuss the related work in the next section. In Section 4.3, we present a variant of the Tucker parametrization which allows to interpolate between Tucker and CP. The extreme case of this variant, which is equivalent to CP up to a fixed rotation of the embedding, serves as control model to highlight the deficiency of the diagonal rescaling of Adagrad for Tucker-style parametrizations in experiments reported in

Section 4.4 . We present the modified version of Adagrad in Section 4.5 and present experimental results on standard benchmarks of knowledge base completion in Section 4.7.

4.2 Link prediction in knowledge bases

Notation Tensors and matrices are denoted by uppercase letters. For a matrix U, u_i is the vector corresponding to the i-th row of U. The tensor product is written $\otimes$ and the Hadamard product (i.e., elementwise product) is written $\odot$.

4.2.1 Learning setup

A knowledge base consists of a set S of triples (subject, predicate, object) that represent (true) known facts. The goal of link prediction is to recover facts that are true but not in the database. The data is represented as a tensor $\tilde{X} \in \{0,1\}^{N \times L \times N}$ for N the number of entities and L the number of predicates. Given a training set of triples, the goal is to provide a ranking of entities for queries of the type (subject, predicate, ?) and (?, predicate, object). Following Lacroix et al. [70], we use the cross-entropy as a surrogate of the ranking loss. As proposed by Lacroix et al. [70] and Kazemi and Poole [61], we include reciprocal predicates: for each predicate P in the original dataset, and given an item o, each query of the form (?, P, o) is reformulated as a query (o, P^{-1}, ?), where o is now the subject of P^{-1}. This doubles the effective number of predicates but reduces the problem to queries of the type (subject, predicate, ?) only.

For a given triple $(i, j, k) \in S$, the training loss function for a tensor X is then

$$\ell_{i,j,k}(X) = -X_{i,j,k} + \log\Big(\sum_{k' \neq k} \exp(X_{i,j,k'})\Big). \tag{4.1}$$

For a tensor $X(\theta)$ parameterized by θ, the parameters $\hat{\theta}$ are found by minimizing

the regularized empirical risk with regularizer Λ:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \frac{1}{|S|} \sum_{(i,j,k)\in S} \ell_{i,j,k}(X(\theta)) + \nu\Lambda(\theta). \tag{4.2}$$

This work studies specific models for $X(\theta)$, inspired by CP and Tucker parametrizations. We discuss the related work on tensor parametrizations and link prediction in knowledge bases below.

4.2.2 Related work

CP and variants

The low-rank parametrization (CP) of a tensor X is defined entrywise by

$$\forall i, j, k, \quad X_{i,j,k} = \langle u_i, v_j, w_k \rangle := \sum_{r=1}^{d} u_{ir} v_{jr} w_{kr}.$$

The smallest value of d for which this parametrization exists is the rank of X. Each element $X_{i,j,k}$ is thus represented as a multi-linear product of the 3 embeddings in $\mathbb{R}^d$ associated respectively to the ith subject, the jth predictate and the kth object.

CP currently achieves near state-of-the-art performances on standard benchmarks of knowledge base completion [61, 70]. Nonetheless, the best reported results are with the ComplEx model [118], which learns complex-valued embeddings and sets the embeddings of the objects to be the complex conjugate of the embeddings of subjects, i.e., $w_k = \bar{u}_k$. Prior to ComplEx, Dismult was proposed [126] as a variant of CP with $w_k = u_k$. While this model obtained good performances [60], it can only model symmetric relations and does not perform as well as ComplEx. CP-based models are optimized with vanilla Adam or Adagrad and a single regularization parameter [118, 60, 70] and do not require additional heuristics for training.

Tucker Parametrization and its variants

Given a tensor X of size $N \times L \times N$, the Tucker parametrization of X is defined entrywise by

$$\begin{aligned} \forall i, j, k, \quad X_{i,j,k} &= \langle u_i \otimes v_j \otimes w_k, C \rangle \\ &:= \sum_{r_1=1}^{d_1} \sum_{r_2=1}^{d_2} \sum_{r_3=1}^{d_3} C_{r_1,r_2,r_3} u_{ir_1} v_{jr_2} w_{kr_3}. \end{aligned}$$

The triple (d_1, d_2, d_3) are the rank parameters of the parametrization. We also use a multilinear product notation $X = [\![C; U, V, W]\!]$, where U, V, W are the matrices whose rows are respectively u_j, v_k, w_l and C the three dimensional $d_1 \times d_2 \times d_3$ *core tensor*. Note that the CP parametrization is a Tucker parametrizatioin in which $d_1 = d_2 = d_3 = d$ and C is the identity, which we write $[\![U, V, W]\!]$. With a non-trivial core tensor, Tucker parametrization is thus more flexible than CP for fixed embedding size. In knowledge base applications, we typically have $d \leq L \ll N$, so the vast majority of the model parameters are in the embedding matrices of the entities U and W. When constraints on the number of model parameters arise (e.g., memory constraints), Tucker models appear as natural candidates to increase the expressivity of the parametrization compared to CP with limited impact on the total number of parameters.

While many variants of the Tucker parametrization have been proposed in the literature on tensor factorization [see e.g., 64], the first approach based on Tucker for link prediction in knowledge bases is RESCAL [91]. RESCAL uses a special form of Tucker parametrization in which the object and subject embeddings are shared, i.e., $U = W$, and it does not compress the relation matrices. In the multilinear product notation above, a RESCAL model is thus written as $X = [\![C; U, I, U]\!]$. Despite some success on a few smaller datasets, RESCAL performances drop on larger datasets [93]. This decrease in performances has been attributed either to improper regularization [91] or optimization issues [125]. Balazevic et al. [7] revisits Tucker parametrization in the context of large-scale knowledge bases and resolves some of the optimization

and regularization issues using learning rate annealing, batch-normalization and dropout. It comes at the price of more hyperparameters to tune for each dataset (label smoothing, three different dropouts and a learning rate decay), and as we discuss in our experiments, the results they report are not better than ComplEx for the same number of parameters.

Two methods were previously proposed to interpolate between the expressivity of RESCAL and CP. Xue et al. [125] expands the HolE model [93] (and thus the ComplEx model [51]) based on cross-correlation of embeddings to close the gap in expressivity with the Tucker parametrization for a fixed embedding size. Jenatton et al. [56] express the relation matrices in RESCAL as low-rank combination of a family of matrices. We describe the link between these approaches and ours in Appendix B.1.3. None of these approach however studied the effect of their formulation on optimization, and reported results inferior to ours.

Other approaches

(Graph) neural networks for link prediction Several methods have introduced models that go beyond the form of Tucker and canonical parametrizations. ConvE [30] uses a convolution on a 2D tiling of the subject and relation embeddings as input to a 2-layer neural net that produces a new embedding for the pair, then compares to the object embedding. Graph neural networks [101, 95, 74, 19] have recently gained popularity and have been applied to link prediction in knowledge bases by Schlichtkrull et al. [102]. This model uses a graph convolutional architecture to generate a variant of CP.

Poincaré embeddings Poincaré embeddings have been proposed as an alternative to usual tensor parametrization approaches to learn smaller embeddings when the relations are hierarchical [94]. The method has recently been extended to link prediction in relational data with very good performance trade-offs for small dimensional embeddings on the benchmark using WordNet [6], which contains relationships such as hypernyms and hyponyms which are purely hierarchical. However, such good results

do not extend to other benchmarks.

Approximate adaptive optimization methods

The Adagrad [32] optimization algorithm is used in practice only in its diagonal approximated form. This scheme is particularly efficient when large embedding tables are involved, since sparse gradients are naturally handled. In this paper, we start from the diagonal approximation of Adagrad and make it invariant to an arbitrary rotation of the parameters. However, other efficient approximations of Adagrad have been proposed that are naturally robust to such rotations. Krummenacher et al. [68] suggests to use low-dimensional random sketches of the gradient and Agarwal et al. [2] only uses a small window of gradients to compute the preconditioning. These updates scale with the full number of parameters even for sparse updates. On the contrary, our algorithm remains as efficient in practice for very sparse gradients. Gupta et al. [46] has memory requirements which scale with the square of the number of entities, which is not applicable in our case.

4.3 Interpolating between CP and Tucker

In order to better understand the underlying difficulties in learning (variants of) Tucker parametrizations compared to CP, our analysis starts from a Tucker model in which the core tensor is itself decomposed with CP. Given a $N \times L \times N$ tensor, a fixed d and assuming a (d, d, d) Tucker parametrization to simplify notation, a Tucker model where the core tensor is itself decomposed with a rank-D CP can be written as (details are given in Appendix B.1.1):

$$X_{ijk} = \langle u_i \otimes v_j \otimes w_k, C \rangle = \langle P_1 u_i, P_2 v_j, P_3 w_k \rangle$$
$$\text{or equivalently } X = [\![U P_1^\top, V P_2^\top, W P_3^\top]\!],$$

where P_1, P_2, P_3 are all $D \times d$ matrices. Since most knowledge bases have much fewer predicates than entities ($L \ll N$), the dimension of the predictate factors has little

impact on the overall number of model parameters. So in the remainder of the paper, we always consider $P_2 = I$. Learning matrices U, V, W, P_1, P_3 of this parametrizatioin simultaneously leads to the following model, which we call CP-Tucker (CPT):

$$\begin{aligned}&\text{(CPT) } X_{ijk} = \langle P_1 u_i, v_j, P_3 w_k \rangle \\ &u_i, w_k \in \mathbb{R}^d, v_j \in \mathbb{R}^D, P_i \in \mathbb{R}^{D \times d}.\end{aligned}$$

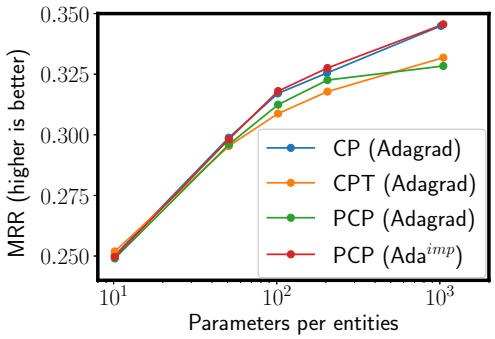
The CPT model is similar to a CP model except that the embedding matrices U and W have an additional low-rank constraint (d instead of D). We say that the model *interpolates* between CP and Tucker because for $D = d$ it is equivalent to CP (as long as P_1 and P_3 are full rank), whereas for $D = d^2$ we recover a full Tucker model because the matrices P_1 and P_3 can be chosen such that $\langle P_1 u_i, v_j, P_3 w_k \rangle = u_i \text{Mat}(v_j) w_k^T$, where Mat is the operator that maps a d^2 vector to a $d \times d$ matrix (see Appendix B.1.2).

CPT is similar to CANDELINC [23], except that in CANDELINC the factors U, V and W are fixed and used to compress the data in order to efficiently learn the P_i. Closer to CPT, Bro and Andersson [17] first learn a Tucker3 parametrization of X before applying CANDELINC using the learned factors. These methods are only applicable to least-square estimation, and for tensors of smaller scale than knowledge bases.

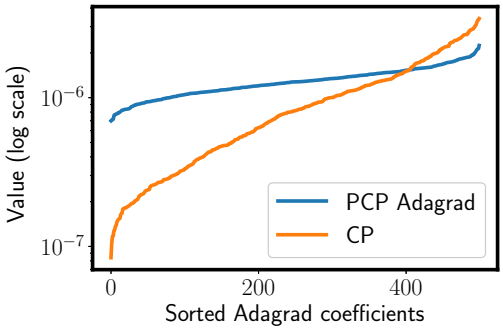
Fixed projection matrices: The Constrained Tucker (PCP)

Parametrization In order to clarify the difficulty that arise when learning a CPT model compared to a CP model, we study a simpler model in which the matrices P_1 and P_3 are not learned but rather fixed during training and taken as random matrices with orthonormal columns. We call the resulting model the constrained tucker (PCP) parametrization, since P_1, P_3 project the embeddings of dimension d into a higher dimension D:

$$\begin{aligned}&\text{(PCP) } X_{ijk} = \langle P_1 u_i, v_j, P_3 w_k \rangle \\ &u_i, w_k \in \mathbb{R}^d, v_j \in \mathbb{R}^D, \text{ fixed } P_1, P_3 \in \mathbb{R}^{D \times d}\end{aligned}$$



(a) Performances on FB15K-237 in the control experiments for $D = d$, i.e., when PCP is a reparameterization of CP. We observe that PCP and CPT with vanilla Adagrad, which are variants of Tucker, underperform compared to CP. As expected in this case $D = d$, our modification of the Adagrad update leads to the same performances for PCP as for CP.



(b) Adagrad coefficients for subject/object embedding matrices in the control experiments ($D = d$) for CP and PCP, averaged by columns (i.e., embedding dimension) and sorted by values. Adagrad coefficients decay exponentially for CP, but the values are similar across most dimensions in PCP: the fixed unitary transform in PCP removes the benefit of Adagrad.

Figure 4-1: Results of the control experiment of Section 4.4.

When $D = d$ the matrices P_i are then fixed unitary transformations. The PCP (or CPT) model in this case $D = d$ is then equivalent to a CP model, up to a fixed invertible transformation of the embeddings. The capacity of the model grows beyond that of CP as D increases up to d^2.

4.4 Optimization issues with CPT and PCP

As discussed in the related works, previous results suggest that Tucker models are more difficult to train than CP models. The goal of this section is to isolate an issue faced with CPT/PCP models when trained with vanilla adaptive gradient methods such as Adagrad or Adam.

4.4.1 Control experiment: Unitary P_1 and P_3 in PCP

When $D = d$ in PCP, the model becomes equivalent to CP. Indeed, the matrices P_1 and P_3 are unitary ($P_1 P_1^\top = P_3 P_3^\top = I$) and so $[\![(UP_1)P_1^\top, V, (WP_3)P_3^\top]\!] = [\![U, V, W]\!]$. There is no practical interest in considering this degenerate case of PCP, we only use

it in the following toy experiment to exhibit one of the difficulties encountered when training PCP.

We perform a simple control experiment in which we take one of the standard benchmarks of link prediction in knowledge bases, called FB15K-237, and train a CP model for different values of the rank D and a PCP model with $D = d$ with vanilla Adagrad. The full experimental protocol, including hyperparameter tuning, is similar to our main experiments and is described in Section 4.7.2. Figure 4-1a plots the performances in terms of the standard metric mean reciprocal rank (higher is better) as a function of D of CP (blue curve) and PCP (red curve, called PCP (Adagrad)).

We observe that, after a fixed amount of epochs, CP obtains significantly better performances than CPT for larger embedding dimension D. Since in this toy experiment CP and PCP can represent exactly the same tensors and have equivalent regularizations, the only difference between the algorithms that can explain the difference in performances is in how the optimization is carried out, namely the diagonal rescaling performed by Adagrad: Adagrad adapts the learning rate on a per-parameter basis, depending on previous and current gradients, and is therefore not invariant by the addition of the matrices P_1 and P_2 even if these are unitary (we provide the formal justification in the next section). This is shown experimentally in Figure 4-1b where we plot the average Adagrad coefficients for each embedding dimensions (i.e., adagrad coefficients of subject/object embedding matrices averaged by column). The addition of the random P_1 and P_2 flattens the Adagrad weights, which in turn removes all the benefit of the adaptive rescaling of the algorithm.

For reference, we also tried to directly learn all parameters including P_1 and P_3 (i.e., learn a CPT model) with vanilla Adagrad. The performances obtained are also lower than those of CP, as shown in Figure 4-1a (orange curve).

4.5 A rotation invariant AdaGrad: Ada$^{\mathbf{imp}}$

In this section, we study the optimization problem in more details, and more precisely the effect of the diagonal rescaling performed by Adagrad. As a remainder, given a

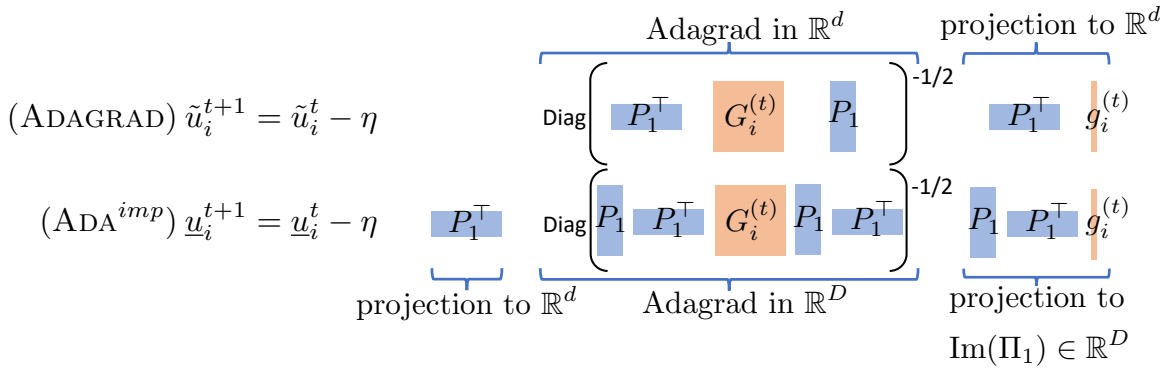


Figure 4-2: Comparison of the updates in Ada$^{\text{imp}}$ and Adagrad.

sequence of stochastic gradients $g^{(t)}$ of $\mathcal{L}$ and denoting $G^{(t)} = \epsilon I + \sum_{\tau=1}^{t} g^{(\tau)} g^{(\tau)\top}$, the (practical) AdaGrad update is:

$$\theta_p^{(t+1)} = \theta_p^{(t)} - \eta g_p^{(t)} / \sqrt{G_{pp}^{(t)}} \quad \text{or equivalently}$$
$$\theta^{(t+1)} = \theta^{(t)} - \eta \, \mathrm{Diag}(G^{(t)})^{-1/2} g^{(t)}$$

where $\mathrm{Diag}(G)$ is the diagonal matrix obtained by extracting the diagonal elements of G.

4.5.1 Two equivalent parametrizations of PCP

The following parametrization is equivalent to PCP, but its embeddings are expressed in $\mathbb{R}^D$:

$$(\mathrm{PCP}_{\mathrm{FULL}})\ X_{i,j,k} = \langle P_1 P_1^\top u_i, v_j, P_3 P_3^\top w_k \rangle$$
$$\text{with } u_i, v_j, w_k \in \mathbb{R}^D, \text{ fixed } P_1, P_2 \in \mathbb{R}^{D \times d}.$$

Note that with $\underline{u}_i = P_1^\top u_i$ and $\underline{w}_k = P_3^\top w_k$, we go from $\mathrm{PCP}_{\mathrm{full}}$ to PCP. The practical differences between PCP and $\mathrm{PCP}_{\mathrm{full}}$ are that $\mathrm{PCP}_{\mathrm{full}}$ learn embeddings in the high-dimensional space, maintaining the low-rank structure of the overall entity embeddings through the orthogonal projections $P_1 P_1^T$ and $P_3 P_3^T$. The practical interest of $\mathrm{PCP}_{\mathrm{full}}$ is not in terms of modeling but rather from the optimization perspective with Adagrad because it has a structure that is closer to that of CP.

Algorithm 1 Step of PComplEx training

$(i, j, k) \leftarrow$ sample from S
$g_i, g_j, g_k \leftarrow$ gradients in (Pu_i, Pu_j, w_k)
$u_i, \tilde{G}_i \leftarrow \text{Ada}^{\text{imp}}\ (\eta, u_i, g_u, \tilde{G}_i, P)$
$u_j, \tilde{G}_j \leftarrow \text{Ada}^{\text{imp}}\ (\eta, u_j, g_j, \tilde{G}_j, P)$
$w_k, G_k \leftarrow \text{AdaGrad}(\eta, w_k, g_k, G_k)$

Algorithm 2 Ada^{imp}

Input: $\eta, x^{(t)}, g^{(t)}, \tilde{G}^{(t-1)}, P$
$\tilde{g}^{(t)} \leftarrow P(P^\top g^{(t)})$
$\tilde{G}^{(t)} \leftarrow \tilde{G}^{(t-1)} + \tilde{g}^{(t)} \odot \tilde{g}^{(t)}$
$x^{(t+1)} \leftarrow x^{(t)} - \eta P^\top \text{Diag}(\tilde{G}^{(t)})^{-1/2} \tilde{g}^{(t)}$
return $x^{(t+1)}, \tilde{G}^{(t)}$

Figure 4-3: The two algorithms used in the training of PComplEx

Indeed, for $d = D$, P_1 and P_3 disappear in PCP_{full}, so that optimizing a PCP_{full} model with Adagrad is equivalent to optimizing a CP model with Adagrad. This property suggests an alternative to the vanilla PCP + Adagrad algorithm, which we call *implicit Adagrad*:

Implicit Adagrad: Ada^{imp} The approach we propose is to effectively optimize PCP_{full} with Adagrad. However, when $d < D$, which is the interesting case for PCP, we notice that we do not need to maintain embeddings in $\mathbb{R}^D$. Our approach, called Ada^{imp}, computes the gradients and Adagrad coefficients with respect to $u_i, w_k \in \mathbb{R}^D$, but the full dimension factor matrices U and W are never explicitly stored in memory. Rather, we store $\underline{u_i} = P_1^\top u_i$ and $\underline{w_k} = P_3^\top w_k \in \mathbb{R}^d$, which is all that is required for any model computation in PCP_{full} since P_1 and P_3 are fixed. Overall, the effective model parameters are exactly the same as in PCP, and we call this approach PCP +Ada^{imp}.

An Ada^{imp} update is described in Algorithm 2. While PCP +Adagrad and PCP +Ada^{imp} work with the same number of *model* parameters, the fundamental difference is the computation of Adagrad coefficients. Since Ada^{imp} effectively applies Adagrad to PCP_{full}, we need to maintain the Adagrad coefficients in $\mathbb{R}^D$ even when $d < D$: the overall update is first computed in $\mathbb{R}^D$ and projected back to $\mathbb{R}^d$ *after* the application of Adagrad rescaling. In constrat, in vanilla PCP +Adagrad, the gradient is projected to $\mathbb{R}^d$ *before* Adagrad rescaling.

4.5.2 Implicit optimization of PCP_{full}: Ada^{imp}

In this section, we discuss more formally the Ada^{imp} updates, and how they compare to PCP +Adagrad. In the following, $\tilde{U}, \tilde{W}$ are in $\mathbb{R}^{N \times d}$, whereas U, V and W are

in $\mathbb{R}^{N\times D}$. Using $d \leq D$, $P_1, P_3 \in \mathbb{R}^{D\times d}$, and we use the notation $\Pi_1 = P_1 P_1^\top$ and $\Pi_3 = P_3 P_3^\top$.

The empirical risk $\mathcal{L}$ can be expressed as a function of three matrices $M^{(1)}$, $M^{(2)}$ and $M^{(3)}$ corresponding to factors of a CP parametrization. We study the following problems :

$$\text{(PCP)} \ \operatorname*{argmin}_{\tilde{U},V,\tilde{W}} \mathcal{L}(\tilde{U}P_1^\top, V, \tilde{W}P_3^\top)$$

$$\text{(PCP}_{\text{FULL}}\text{)} \ \operatorname*{argmin}_{U,V,W} \mathcal{L}(U\Pi_1^\top, V, W\Pi_3^\top)$$

We focus on a step at time (t) on vectors $\tilde{u}_i$ and $\underline{u}_i$. We assume that at this time t, the tensor iterates are the same, that is $\tilde{U} = UP_1^\top$ (resp. $\tilde{W} = WP_1^\top$) so that $[\![\tilde{U}P_1^\top, V, \tilde{W}P_3^\top]\!] = [\![U\Pi_1^\top, V, W\Pi_3^\top]\!]$. In this case, the gradient $\nabla_{M_i^{(1)}}\mathcal{L}$ is the same in both optimization problems, we denote it by $g_i^{(t)}$. Let $G_i^{(t)} = \epsilon I_d + \sum_{\tau=1}^t g_i^{(\tau)} g_i^{(\tau)^\top}$. The updates for (PCP) are:

$$\tilde{u}_i^{t+1} = \tilde{u}_i^t - \eta \text{Diag}(P_1^\top G_i^{(t)} P_1)^{-1/2} P_1^\top g_i^{(t)}. \tag{4.3}$$

Note that due to the presence of P_1 inside the Diag operator, the update (4.3) is not rotation invariant. Moreover, for random P_1, the matrix $P_1^\top G_i^{(t)} P_1$ will be far from diagonal with high probability, making its diagonal meaningless. This is visualized in Figure 4-1b.

Similar updates can be derived for (PCP_{full}):

$$u_i^{t+1} = u_i^t - \eta \text{Diag}(\Pi_1^\top G_i^{(t)} \Pi_1)^{-1/2} \Pi_1^\top g_i^{(t)}. \tag{4.4}$$

As a sanity check, clearly, for $d = D$ and $\Pi_1 = I$, the update (4.4) is equivalent to the Adagrad update for the CP model. In the general case $d \leq D$, in order to avoid storing $U \in \mathbb{R}^{N\times D}$, we apply these updates *implicitly* with Ada^{imp}, by storing $\underline{u}_i^{(t)} = P_1^\top u_i^{(t)}$ in $\mathbb{R}^d$. We compare the updates in Figure 4-2.

Going back to our control experiment, we note on Figure 4-1a that PCP $+\text{Ada}^{\text{imp}}$

matches the performances of CP+Adagrad for all D, indicating that we fixed this optimization issue. Further experiments on the convergence speed of Ada$^{\text{imp}}$ for other datasets are available in Appendix B.3.7.

Convergence guarantees Note that the tensor iterates obtained by applying Ada$^{\text{imp}}$ are exactly those that we would obtain by applying Adagrad on the PCP$_{\text{full}}$ formulation. Thus, theoretical guarantees for our algorithm are exactly the same as those of Adagrad.

4.5.3 Alternatives to Ada$^{\text{imp}}$

Another solution would be to use Adagrad projected on the column space of Π, but we show in Appendix B.2.1 that even with the diagonal approximation, this is impractical. Note that the version of Adagrad which uses the full matrix $G^{(t)}$ is rotation invariant (see Appendix B.2.2 for details), but it cannot be used at the scale of our problems.

It could be argued that the strength of the AdaGrad algorithm in our context mostly comes from its adaptation to the different frequencies of updates of each embedding. In fact, this is one of the examples chosen in Duchi et al. [32] to display the advantages of AdaGrad compared to stochastic gradient descent. A version of AdaGrad that would only keep one coefficient per embedding (we call this version Ada$^{\text{row}}$) would be invariant to unitary transforms by design and would adapt to the various update frequencies of different embeddings. In fact, this version of AdaGrad is used to save memory in Lerer et al. [72]. We test this algorithm in Appendix B.3.2. The difference in performances shows that this adaptation is not sufficient to recover the performances of the finer diagonal AdaGrad in our setting.

4.5.4 Complexity

The time complexity of our Ada$^{\text{imp}}$ update for a batch of size B is $\mathcal{O}(D \cdot d \cdot B)$ which is similar, up to constants, to the complexity of updates for the AdaGrad algorithm. We do not notice any runtime differences between our algorithm applied in dimensions

(d, D) and a CP parametrization of rank D (see Section 4.7). The runtime for large enough D is dominated by the matrix product ($\mathcal{O}(D^2 \cdot B)$) required to compute the cross-entropy in Equation (4.1).

4.6 Projected ComplEx

As the state-of-the-art variant of CP is ComplEx [118, 70], we propose the following alternative to PCP base on ComplEx with Ada$^{\text{imp}}$ in practice. Given the ComplEx parametrization $X = Re([U, V, \overline{U}])$, a low-rank parametrization of the entity factor U as $P\tilde{U}$ leads to the model PComplEx we use in the experiments of Section 4.7:

$$\begin{aligned}
\text{(PComplEx) } X_{ijk} &= Re\big(\langle Pu_i, v_j, \overline{Pu_k}\rangle\big) \\
&= Re\big(\langle u_i, P^\top \text{Diag}(v_j) P, \overline{u_k}\rangle\big) \\
u_i, w_k \in \mathbb{C}^d, v_j \in \mathbb{C}^D,& \text{ fixed } P \in \mathbb{R}^{D \times d}
\end{aligned}$$

PComplEx is similar to ComplEx but with interactions described by full matrices of rank D that share a same basis. We learn this parametrization with Algorithms 1 and 2.

4.7 Experiments

In this section, we compare ComplEx optimized with AdaGrad and PComplEx optimized with Ada$^{\text{imp}}$. We optimize the regularized empirical risk of Equation (4.2). Following Lacroix et al. [70], we regularize ComplEx with the weighted nuclear-3 norm, which is equivalent to regularizing $\|u_i\|_3^3 + \|u_j\|_3^3 + \|w_k\|_3^3$ for each training example (i, j, k). For PComplEx based models, we regularize $\|Pu_i\|_3^3 + \|v_j\|_3^3 + \|Pu_k\|_3^3$ by analogy.

We conduct all experiments on a Quadro GP 100 GPU. The code for PComplEx and Ada$^{\text{imp}}$ is available in the supplementary materials, experiments on ComplEx use the

code[1] from [70]. We include results from TuckER [7], DRT and SRT which are the two models considered in Wang et al. [121], ConvE [30], HolEx [125], LFM [56] and MurP [6] without re-implementation on our parts. All the parameters for the experiments are reported in Appendix B.3.6. Variance of performances in PComplEx due to random choice of P is similar to the variance of ComplEx. We present experiments on the WN18 dataset for 5 different seeds in Appendix B.3.3.

4.7.1 Datasets

WN18 [13] is taken from the Wordnet database which contains words and relations between them. WN18RR [30] is a filtering of this dataset which removes train/test leakage. YAGO3-10 [30] is taken from the eponymous knowledge base. Finally, SVO [56] contains observations of Subject, Verb, Object triples. All statistics for these datasets can be found in Appendix B.3.1. Experiments on the FB15K and FB15K-237 datasets are deferred to Appendix B.3.4.

4.7.2 Results

We report the filtered Mean Reciprocal Rank [93] on Figure 4-4. For SVO, we report the only figure available in previous work which is the filtered hits at 5% [56]. These measures are detailed in Appendix B.3.5. Only the grid-search parameters were given for LFM, so we were not able to obtain a precise number of parameters for the number they report.

On WN18, SVO and YAGO3-10 we observe sizable performance gains for low embedding sizes : up to 0.14 MRR points on WN18, 0.05 MRR points on YAGO and 0.03 H@5% points on SVO.

The TuckER [7] model performs similarly to PComplEx and ComplEx except on FB15K and WN18 where it underperforms (see Appendix B.3.4). We expect this discrepancy to come from a less extensive grid-search rather than any intrinsic differences in the models that are both based on the Tucker parametrization. The

[1] `https://github.com/facebookresearch/kbc`

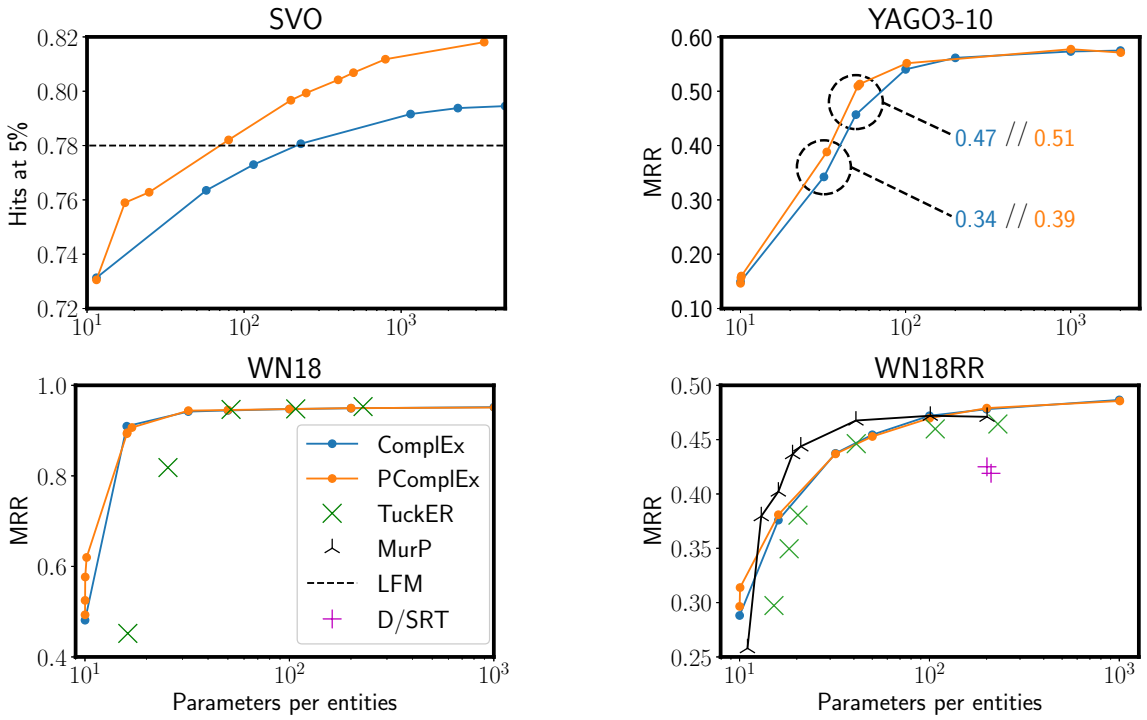


Figure 4-4: MRR as a function of #floats / entities (see Appendix B.3.5) on four knowledge bases. We plot the convex envelope of various operating points we tested, varying D for several values of d. For some datasets (WN18, WN18RR), Adam [63] is beneficial, in which case we use the implicit adaptation of Adam detailed in Appendix B.2.4.

consistency on all operating points of our method with ComplEx shows an advantage of our method, which enjoys the same learning rate robustness as AdaGrad, and does not require choosing a learning-rate decay, leading to easier experiments with only the regularization strength to tune. The MurP model [6] provides good performances for low embedding sizes on WN18RR, but underperforms on FB15K-237 (see Appendix B.3.4). All other models fail to match the performances of ComplEx and PComplEx with equivalent number of parameters (see Appendix B.3.5 for details on the number of parameters for each methods).

The Tucker parametrization seems well adapted to the SVO dataset since this is where our approach is most beneficial, even allowing us to outperform ComplEx for high dimensions.

4.8 Conclusion

We use a formulation of the Tucker parametrization expressed as a CP parametrization with low-rank factors. A controlled experiment shows that a fixed random core tensor is detrimental to the speed of Adagrad. We propose a faster rotation-invariant algorithm which stores learning rates in the uncompressed space. Our model, despite using a fixed random core tensor, provides better performances than ComplEx in the low-rank regime, and matches its performances in the other regimes. On the SVO dataset, where ranking is done on the uncompressed mode, our method yields better performances in all regimes, allowing us to outperform ComplEx and previous state of the art.

Chapter 5

Temporal Tensor Completion

> The easiest way to solve a problem is to deny it exists.
>
> Isaac Asimov

Most algorithms for representation learning and link prediction in relational data have been designed for static data. However, the data they are applied to usually evolves with time, such as friend graphs in social networks or user interactions with items in recommender systems. This is also the case for knowledge bases, which contain facts such as (US, has president, B. Obama, [2009-2017]) that are valid only at certain points in time. For the problem of link prediction under temporal constraints, i.e., answering queries such as (US, has president, ?, 2012), we propose a solution inspired by the CP parametrization of tensors of order 4. We introduce new regularization schemes and present an extension of ComplEx [118] that achieves state-of-the-art performance. Additionally, we propose a new dataset for knowledge base completion constructed from Wikidata, larger than previous benchmarks by an order of magnitude, as a new reference for evaluating temporal and non-temporal link prediction methods.

5.1 Introduction

Link prediction in relational data has been the subject of interest, given the widespread availability of such data and the breadth of its use in bioinformatics [128], recommender systems [66] or Knowledge Base completion [92]. Relational data is often temporal, for example, the action of buying an item or watching a movie is associated to a timestamp. Some medicines might not have the same adverse side effects depending on the subject's age. The task of *temporal* link prediction is to find missing links in graphs at precise points in time.

In this work, we study temporal link prediction through the lens of temporal knowledge base completion, which provides varied benchmarks both in terms of the underlying data they represent, but also in terms of scale. A knowledge base is a set of facts (subject, predicate, object) about the world that are known to be true. Link prediction in a knowledge base amounts to answer incomplete queries of the form (subject, predicate, ?) by providing an accurate ranking of potential objects. In temporal knowledge bases, these facts have some temporal metadata attached. For example, facts might only hold for a certain time interval, in which case they will be annotated as such. Other facts might be event that happened at a certain point in time. Temporal link prediction amounts to answering queries of the form (subject, predicate, ?, timestamp). For example, we expect the ranking of queries (USA, president, ?, timestamp) to vary with the timestamps.

As tensor factorization methods have proved successful for Knowledge Base Completion [92, 118, 70], we express our Temporal Knowledge Base Completion problem as an order 4 tensor completion problem. That is, timestamps are discretized and used to index a 4-th mode in the binary tensor holding (subject, predicate, object, timestamps) facts.

First, we introduce a ComplEx [118] parametrization of this order 4 tensor, and link it with previous work on temporal Knowledge Base completion. This parametrization yields embeddings for each timestamps. A natural prior is for these timestamps representation to evolve slowly over time. We are able to introduce this prior as a

regularizer for which the optimum is a variation on the nuclear p-norm. In order to deal with heterogeneous temporal knowledge bases where a significant amount of relations might be non-temporal, we add a non-temporal component to our parametrization.

Experiments on available benchmarks show that our method outperforms the state of the art for similar number of parameters. We run additional experiments for larger, regularized models and obtain improvements of up to 0.07 absolute Mean Reciprocal Rank (MRR).

Finally, we propose a dataset of $400k$ entities, based on Wikidata, with $7M$ train triples, of which 10% contain temporal validity information. This dataset is larger than usual benchmarks in the Knowledge Base completion community and could help bridge the gap between the method designed and the envisaged web-scale applications.

5.2 Related Work

Matrices and tensors are upper case letters. The i-th row of U is denoted by u_i while it's $j-th$ column is denoted by $U_{:,j}$. The tensor product of two vectors is written $\otimes$ and the hadamard (elementwise) product $\odot$.

Static link prediction methods Standard tensor parametrizations have lead to good results [126, 118, 70, 7] in Knowledge Base completion. The CP parametrization [53] is the tensor equivalent to the low-rank parametrization of a matrix. A tensor X of canonical rank R can be written as:

$$X = \sum_{r=1}^{R} U_{:,r} \otimes V_{:,r} \otimes W_{:,r} = [\![U, V, W]\!] \iff \forall (i,j,k),\ X_{i,j,k} = \sum_{r=1}^{R} u_{i,r} v_{j,r} w_{k,r} = \langle u_i, v_j, w_k \rangle$$

Setting $U = W$ leads to the Distmult [126] model, which has been successful, despite only being able to represent symmetric score functions. In order to keep the parameter sharing scheme but go beyond symmetric relations, Trouillon et al. [118] use complex parameters and set W to the complex conjugate of U, $\overline{U}$. Regularizing this algorithm with the variational form of the tensor nuclear norm as well as a slight transformation

to the learning objective (also proposed in Kazemi and Poole [61]) leads to state of the art results in Lacroix et al. [70].

Other methods are not directly inspired from classical low-rank tensor parametrizations. For example, TransE [13] models the score as a distance of the translated subject to an object representation. This method has lead to many variations [57, 90, 122], but is limited in the relation systems it can model [61] and does not lead to state of the art performances on current benchmarks. Finally Schlichtkrull et al. [102] propose to generate the entity embeddings of a CP-like tensor parametrization by running a forward pass of a Graph Neural Network over the training Knowledge Base. The experiments included in this work did not lead to better link prediction performances than the same parametrization (Distmult) directly optimized [60].

Temporal link prediction methods Sarkar and Moore [100] describes a bayesian model and learning method for representing temporal relations. The temporal smoothness prior used in this work is similar to the gradient penalty we describe in Section 5.3.3. However, learning one embedding matrix per timestamp is not applicable to the scales considered in this work. Bader et al. [4] uses a tensor parametrization called ASALSAN to express temporal relations. This parametrization is related to RESCAL [91] which underperforms on recent benchmarks due to overfitting [93].

For temporal knowledge base completion, Goel et al. [43] learns entity embeddings that change over time, by masking a fraction of the embedding weights with an activation function of learned frequencies. Based on the Tucker parametrization, ConT [81] learns one new core tensor for each timestamp. Finally, viewing the time dimension as a sequence to be predicted, Garcia-Duran et al. [41] use recurrent neural nets to transform the embeddings of standard models such as TransE or Distmult to accomodate the temporal data.

This work follows Lacroix et al. [70] by studying and extending a regularized CP parametrization of the training set seen as an order 4 tensor. We propose and study several regularizer suited to our parametrizations.

DE-SimplE	$2r\left((3\gamma + (1-\gamma))|E| + |P|\right)$
TComplEx	$2r(|E| + |T| + 2|P|)$
TNTComplEx	$2r(|E| + |T| + 4|P|)$

Table 5.1: Number of parameters for each models considered

5.3 Model

In this section, we are given facts (subject, predicate, object) annotated with timestamps, we discretize the timestamp range (eg. by reducing timestamps to years) in order to obtain a training set of 4-tuple (subject, predicate, object, time) indexing an order 4 tensor. We will show in Section 5.5.1 how we reduce each datasets to this setting. Following Lacroix et al. [70], we minimize, for each of the train tuples (i, j, k, l), the instantaneous multiclass loss :

$$\ell(\hat{X}; (i, j, k, l)) = -\hat{X}_{i,j,k,l} + \log\left(\sum_{k'} \exp\left(\hat{X}_{i,j,k',l}\right)\right). \tag{5.1}$$

Note that this loss is only suited to queries of the type (subject, predicate, ?, time), which is the queries that were considered in related work. We consider another auxiliary loss in Section 5.6 which we will use on our Wikidata dataset. For a training set S (augmented with reciprocal relations [70, 61]), and parametric tensor estimate $\hat{X}(\theta)$, we minimize the following objective, with a *weighted* regularizer Ω:

$$\mathcal{L}(\hat{X}(\theta)) = \frac{1}{|S|} \sum_{(i,j,k,l)\in S} \left[\ell(\hat{X}(\theta); (i, j, k, l)) + \lambda\Omega(\theta; (i, j, k, l))\right].$$

The ComplEx [118] parametrization can naturally be extended to this setting by adding a new factor T, we then have:

$$\hat{X}(U, V, T) = \mathrm{Re}\left(\llbracket U, V, \overline{U}, T\rrbracket\right) \iff \hat{X}(U, V, T)_{i,j,k,l} = \mathrm{Re}\left(\langle u_i, v_j, \overline{u_k}, t_l\rangle\right) \tag{5.2}$$

We call this parametrization TComplEx. Intuitively, we added timestamps embedding that modulate the multi-linear dot product. Notice that the timestamp

can be used to equivalently modulate the objects, predicates or subjects to obtain time-dependent representation:

$$\langle u_i, v_j, \overline{u_k}, t_l \rangle = \langle u_i \odot t_l, v_j, \overline{u_k} \rangle = \langle u_i, v_j \odot t_l, \overline{u_k} \rangle = \langle u_i, v_j, \overline{u_k} \odot t_l \rangle.$$

Contrary to DE-SimplE [43], we do not learn temporal embeddings that scale with the number of entities (as frequencies and biases), but rather embeddings that scale with the number of timestamps. The number of parameters for the two models are compared in Table 5.1.

5.3.1 Non-Temporal predicates

Some predicates might not be affected by timestamps. For example, Malia and Sasha will always be the daughters of Barack and Michelle Obama, whereas the "has occupation" predicate between two entities might very well change over time. In heterogeneous knowledge bases, where some predicates might be temporal and some might not be, we propose to decompose the tensor $\hat{X}$ as the sum of two tensors, one temporal, and the other non-temporal:

$$\hat{X} = \text{Re}\left(\llbracket U, V^t, \overline{U}, T \rrbracket + \llbracket U, V, \overline{U}, \mathbf{1} \rrbracket\right) \iff \hat{X}_{i,j,k,l} = Re\left(\langle u_i, v_j^t \odot t_l + v_j, \overline{u_k} \rangle\right) \quad (5.3)$$

We call this parametrization TNTComplEx. Goel et al. [43] suggests another way of introducing a non-temporal component, by only allowing a fraction γ of components of the embeddings to be modulated in time. By allowing this sharing of parameters between the temporal and non-temporal part of the tensor, our model removes one hyperparameter. Moreover, preliminary experiments showed that this model outperforms one without parameter sharing.

5.3.2 Regularization

Any order 4 tensor can be considered as an order 3 tensor by *unfolding* modes together. For a tensor $X \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times N_4}$, unfolding modes 3 and 4 together will lead to tensor $\tilde{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3 N_4}$ [64].

We can see both parametrizations ((5.2) and (5.3)) as order 3 tensors by unfolding the temporal and predicate modes together. Considering the parametrization implied by these unfoldings (see Appendix C.1.1) leads us to the following weighted regularizers [70]:

$$\begin{aligned} \Omega^3(U, V, T; (i, j, k, l)) &= \frac{1}{3}\left(\|u_i\|_3^3 + \|u_k\|_3^3 + \|v_k \odot t_l\|_3^3\right) \\ \Omega^3(U, V^t, V, T; (i, j, k, l)) &= \frac{1}{3}\left(2\|u_i\|_3^3 + 2\|u_k\|_3^3 + \|v_j^t \odot t_l\|_3^3 + \|v_j\|_3^3\right) \end{aligned} \tag{5.4}$$

The first regularizer weights objects, predicates and pairs (predicate, timestamp) according to their respective marginal probabilities. This regularizer is a variational form of the weighted nuclear 3-norm on an order 4 tensor (see subsection 5.3.4 and Appendix C.1.3 for details and proof). The second regularizer is the sum of the nuclear 3 penalties on tensors $[\![U, V^t, \overline{U}, T]\!]$ and $[\![U, V, \overline{U}]\!]$.

5.3.3 Smoothness of temporal embeddings

We have more a priori structure on the temporal mode than on others. Notably, we expect smoothness of the application $i \mapsto t_i$. In words, we expect neighboring timestamps to have close representations. Thus, we penalize the norm of the discrete derivative of the temporal embeddings :

$$\Lambda_p(T) = \frac{1}{|T| - 1} \sum_{i=1}^{|T|-1} \|t_{i+1} - t_i\|_p^p. \tag{5.5}$$

We show in Appendix C.1.2 that the sum of Λ_p and the variational form of the nuclear p norm (5.6) lead to a variational form of a new tensor atomic norm.

5.3.4 Nuclear p-norms of tensors and their variational forms

As was done in Lacroix et al. [70], we aim to use tensor nuclear p-norms as regularizers. The definition of the nuclear p-norm of a tensor [38] of order D is:

$$\|X\|_{p*} = \inf_{\alpha, R, U^{(1)}, \ldots, U^{(D)}} \left\{ \|\alpha\|_1 \mid X = \sum_{r=1}^{R} \alpha_r U^{(1)}_{:,r} \otimes \cdots \otimes U^{(D)}_{:,r}, \forall r, d \ \|U^{(d)}_{:,r}\|_p = 1 \right\}.$$

This formulation of the nuclear p-norm writes a tensor as a sum over *atoms* which are the rank-1 tensors of unit p-norm factors. The nuclear p-norm is NP-hard to compute [38]. Following Lacroix et al. [70], a practical solution is to use the equivalent formulation of nuclear p-norm using their *variational form*, which can be conveniently written for $p = D$:

$$\|X\|_{D*} = \frac{1}{D} \inf_{X = [\![U^{(1)}, \ldots, U^{(D)}]\!]} \sum_{d=1}^{D} \sum_{r=1}^{R} \|U^{(d)}_{:,r}\|_D^D. \tag{5.6}$$

For the equality above to hold, the infimum should be over all possible R. The practical solution is to fix R to the desired dimension of the parametrization. Using this variational formulation as a regularizer leads to state of the art results for order-3 tensors [70] and is convenient in a stochastic gradient setting because it separates over each model coefficient.

In addition, this formulation makes it easy to introduce a weighting as recommended in Srebro and Salakhutdinov [110], Foygel et al. [37]. In order to learn under non-uniform sampling distributions, one should penalize the weighted norm : $\| \left(\sqrt{M^{(1)}} \otimes \sqrt{M^{(2)}} \right) \odot X \|_{2*}$, where $M^{(1)}$ and $M^{(2)}$ are the empirical row and column marginal of the distribution. The variational form (5.6) makes this easy, by simply penalizing rows $U^{(1)}_{i_1}, \ldots, U^{(D)}_{i_D}$ for observed triple $(i_1, \ldots, i_D)$ in stochastic gradient descent. More precisely for $D = 2$ and $N^{(d)}$ the vectors holding the observed count of each index over each mode d:

$$\frac{1}{|S|} \sum_{(i,j) \in S} \|u_i\|_2^2 + \|v_j\|_2^2 = \sum_i \frac{N^{(1)}_i}{S} \|u_i\|_2^2 + \sum_j \frac{N^{(2)}_j}{S} \|v_j\|_2^2 = \sum_i M^{(1)}_i \|u_i\|_2^2 + \sum_j M^{(2)}_j \|v_j\|_2^2.$$

In subsection 5.3.3, we add another penalty in Equation (5.5) which changes the norm of our atoms.In subsection 5.3.2, we introduced another variational form in Equation (5.4) which allows to easily penalize the nuclear 3-norm of an order 4 tensor. This regularizer leads to different weighting. By considering the unfolding of the timestamp and predicate modes, we are able to weight according to the joint marginal of timestamps and predicates, rather than by the product of the marginals. This can be an important distinction if the two are not independent.

5.3.5 Experimental impact of the regularizers

We study the impact of regularization on the ICEWS05-15 dataset, for the TNTComplEx model. For details on the experimental set-up, see Section 5.5.1. The first effect we want to quantify is the effect of the regularizer Λ_p. We run a grid search for the strength of both Λ_p and Ω^3 and plot the convex hull as a function of the temporal regularization strength. As shown in Figure 5-1, imposing smoothness along the time mode brings an improvement of over 2 MRR point.

The second effect we wish to quantify is the effect of the choice of regularizer Ω. A natural regularizer for TNTComplEx would be :

$$\Delta^p(U, V, T; (i, j, k, l)) = \frac{1}{p}\left(2\|u_i\|_p^p + 2\|u_k\|_p^p + \|v_j^t\|_p^p + \|t_l\|_p^p + \|v_j\|_p^p\right).$$

We compare Δ^4, Δ^3 and Δ^2 with Ω^3. The comparison is done with a temporal regularizer of 0 to reduce the experimental space.

Δ^2 is the common weight-decay frequently used in deep-learning. Such regularizers have been used in knowledge base completion [91, 93, 118], however, Lacroix et al. [70] showed that the infimum of this penalty is non-convex over tensors.

Δ^3 matches the order used in the Ω^3 regularizer, and in previous work on knowledge base completion [70]. However, by the same arguments, its minimization does not lead to a convex penalty over tensors.

Δ^4 is the sum of the variational forms of the Nuclear 4-norm for the two tensors of order 4 in the TNTComplEx model according to equation (5.6).

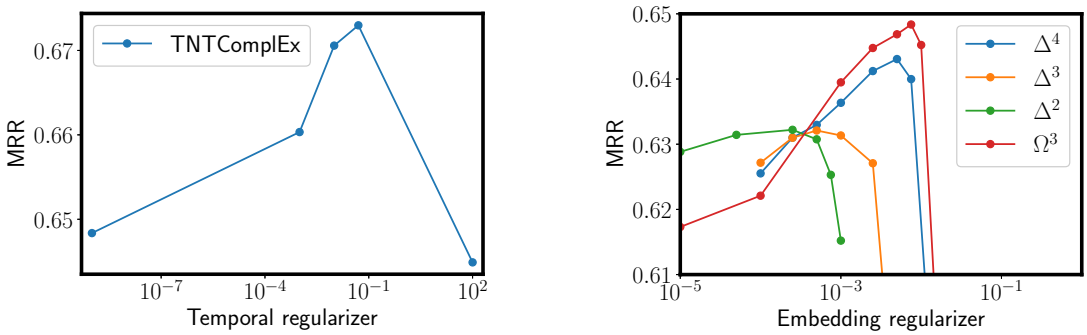


Figure 5-1: Impact of the temporal (left) regularizer and embeddings (right) regularizer on a TNTComplEx model trained on ICEWS05-15.

Detailed results of the impact of regularization on the performances of the model are given in Figure 5-1. The two regularizers Δ^4 and Ω^3 are the only regularizers that can be interpreted as sums of tensor norm variational forms and perform better than their lower order counterparts.

There are two differences between Δ^4 and Ω^3. First, whereas the first is a variational form of the nuclear 4-norm, the second is a variational form of the nuclear 3-norm which is closer to the nuclear 2-norm. Results for exact recovery of tensors have been generalized to the nuclear 2-norm, and to the extent of our knowledge, there has been no formal study of generalization properties or exact recovery under the nuclear p-norm for p greater than two.

Second, the weighting in Δ^4 is done separately over timestamps and predicates, whereas it is done jointly for Ω^3. This leads to using the joint empirical marginal as a weighting over timestamps and predicates. The impact of weighting on the guarantees that can be obtained are described more precisely in Foygel et al. [37].

The contribution of all these regularizers over a non-regularized model are summarized in Table 5.3. Note that careful regularization leads to a 0.05 MRR increase.

5.4 A new dataset for Temporal and non-Temporal Knowledge Base Completion

A dataset based on Wikidata was proposed by Garcia-Duran et al. [41]. However, upon inspection, this dataset contains numerical data as entities, such as ELO rankings of chess players, which are not representative of practically useful link prediction problems. Also, in this dataset, temporal informations is specified in the form of "OccursSince" and "OccursUntil" statements appended to triples, which becomes unwieldy when a predicate holds for several intervals in time. Moreover, this dataset contains only $11k$ entities and $150k$ which is insufficient to benchmark methods at scale.

The GDelt dataset described in Ma et al. [81], Goel et al. [43] holds many triples ($2M$), but does not describe enough entities (500). In order to adress these limitations, we created our own dataset from Wikidata, which we make available at `dataseturl`.

Starting from Wikidata, we removed all entities that were instance of scholarly articles, proteins and others. We also removed disambiguation, template, category and project pages from wikipedia. Then, we removed all facts for which the object was not an entity. We iteratively filtered out entities that had degree at least 5 and predicates that had at least 50 occurrences. With this method, we obtained a dataset of 432715 entities, 407 predicates and 1724 timestamps (we only kept the years). Each datum is a triple (subject, predicate, object) together a timestamp range (begin, end) where begin, end or both can be unspecified. Our train set contains $7M$ such tuples, with about 10% partially specified temporal tuples. We kept a validation and test set of size $50k$ each.

At train and test time, for a given datum (subject, predicate, object, [begin, end]), we sample a timestamp (appearing in the dataset) uniformly at random, in the range [begin, end]. For datum without a temporal range, we sample over the maximum date range. Then, we rank the objects for the partial query (subject, predicate, ?, timestamp).

5.5 Experimental Results

5.5.1 Experimental Set-Up

We follow the experimental set-up in Garcia-Duran et al. [41], Goel et al. [43]. We use models from Garcia-Duran et al. [41] and Goel et al. [43] as baselines since they are the best performing algorithms on the datasets considered. We report the filtered Mean Reciprocal Rank (MRR) defined in Nickel et al. [93]. In order to obtaiqn comparable results, we use Table 5.1 and dataset statistics to compute the rank for each (model, dataset) pair that matches the number of parameters used in Goel et al. [43]. We also report results at ranks 10 times higher. This higher rank set-up gives an estimation of the best possible performance attainable on these datasets, even though the dimension used might be impractical for applied systems. All our models are optimized with Adagrad [32], with a learning rate of 0.1, a batch-size of 1000. More details on the grid-search, actual ranks used and hyper-parameters are given in Appendix C.2.2.

We give results on 3 datasets previously used in the literature : ICEWS14, ICEWS15-05 and Yago15k. The ICEWS datasets are samplings from the Integrated Conflict Early Warning System (ICEWS)[15][1].Garcia-Duran et al. [41] introduced two subsampling of this data, ICEWS14 which contains all events occuring in 2014 and ICEWS05-15 which contains events occuring between 2005 and 2015. These datasets immediately fit in our framework, since the timestamps are already discretized.

The Yago15K dataset [41] is a modification of FB15k [13] which adds "occursSince" and "occursUntil" timestamps to each triples. Following the evaluation setting of Garcia-Duran et al. [41], during evaluation, the incomplete triples to complete are of the form (subject, predicate, ?, occursSince | occursUntil, timestamp) (with reciprocal predicates). Rather than deal with tensors of order 5, we choose to unfold the (occursSince, occursUntil) and the predicate mode together, multiplying its size by two.

Some relations in Wikidata are highly unbalanced (eg. (?, InstanceOf, Human)). For such relations, a ranking evaluation would not make much sense. Instead, we only

[1]More information can be found at `http://www.icews.com`

	ICEWS14	ICEWS15-05	Yago15k
TA	0.48	0.47	0.32
DE-SimplE	0.53	0.51	-
ComplEx	0.47 (0.47)	0.49 (0.49)	**0.35** (0.36)
TComplEx	**0.56** (0.61)	0.58 (0.66)	**0.35** (0.36)
TNTComplEx	**0.56** (**0.62**)	**0.60** (**0.67**)	**0.35** (**0.37**)

Table 5.2: Results for TA [41] and DE-SimplE [43] are the best numbers reported in the respective papers. Our models have as many parameters as DE-SimplE. Numbers in parentheses are for ranks multiplied by 10.

Reg.	MRR
No regularizer	0.62
Δ^2	0.63
Δ^3	0.63
Δ^4	0.64
Ω^3	0.65
$\Omega^3 + \Lambda_4$	**0.67**

Table 5.3: Impact of regularizers on ICEWS05-15 for TNTComplEx.

compute the Mean Reciprocal Rank for missing right hand sides, since the data is such that highly unbalanced relations happen on the left-hand side. However, we follow the same training scheme as for all the other dataset, including reciprocal relations in the training set. The cross-entropy loss evaluated on $400k$ entities puts a restriction on the dimensionality of embeddings at about $d = 100$ for a batch-size of 1000. We leave sampling of this loss, which would allow for higher dimensions to future work.

5.5.2 Results

We compare ComplEx with the temporal versions described in this paper. We report results in Table 5.2. Note that ComplEx has performances that are stable through a tenfold increase of its number of parameters, a rank of 100 is enough to capture the static information of these datasets. For temporal models however, the performance increases a lot with the number of parameters. It is always beneficial to allow a separate modeling of non-temporal predicates, as the performances of TNTComplex

	MRR	NT-MRR	T-MRR
ComplEx	**0.45**	**0.48**	0.29
TComplEx	0.42	0.45	0.30
TNTComplEx	0.44	0.47	**0.32**

Table 5.4: Results on wikidata for entity dimension $d = 100$.

show. Finally, our model match or beat the state of the art on all datasets, even at identical number of parameters. Since these datasets are small, we also report results for higher ranks (10 times the number of parameters used for DE-SimplE).

On Wikidata, 90% of the triples have no temporal data attached. This leads to ComplEx outperforming all temporal models in term of average MRR, since the Non-Temporal MRR (NT-MRR) far outweighs the Temporal MRR (T-MRR). A breakdown of the performances is available in table 5.4. TNTComplEx obtains performances that are comparable to ComplEx on non-temporal triples, but are better on temporal triples. Moreover, TNTComplEx can minimize the temporal cross-entropy (5.7) and is thus more flexible on the queries it can answer.

Training TNTComplEx on Wikidata with a rank of $d = 100$ with the full cross-entropy on a Quadro GP 100, we obtain a speed of $5.6k$ triples per second, leading to experiments time of 7.2 hours. This is to be compared with $5.8k$ triples per second when training ComplEx for experiments time of 6.9 hours. The additional complexity of our model does not lead to any real impact on runtime, which is dominated by the computation of the cross-entropy over $400k$ entities.

5.6 Qualitative study

The instantaneous loss described in equation (5.1), along with the timestamp sampling scheme described in the previous section only enforces correct rankings along the "object" tubes of our order-4 tensor. In order to enforce a stronger temporal consistency, and be able to answer queries of the type (subject, predicate, object, ?), we propose another cross-entropy loss along the temporal tubes:

$$\tilde{\ell}(\hat{X}; (i, j, k, l)) = -\hat{X}_{i,j,k,l} + \log\Big(\sum_{l'} \exp\big(\hat{X}_{i,j,k,l'}\big)\Big). \tag{5.7}$$

We optimize the sum of ℓ defined in Equation (5.1) and $\tilde{\ell}$ defined in Equation (5.7). Doing so, we only lose 1 MRR point. However, by adding the loss $\tilde{\ell}$ we make our model better at answering queries along the time axis. Computing the macro area

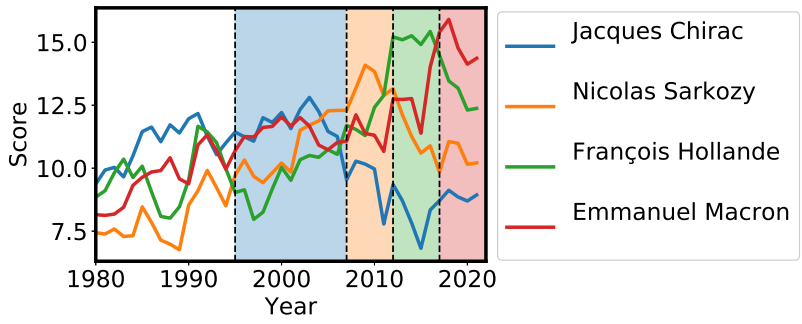


Figure 5-2: Scores for triples (President of the French republic, office holder, {Jacques Chirac | Nicolas Sarkozy | François Hollande | Emmanuel Macron}, $[1980, 2020]$)

under the precision recall curve gives a score of 0.92 for a TNTComplEx model learned with ℓ alone and 0.98 for a TNTComplEx model trained with $\ell + \tilde{\ell}$.

We plot in Figure 5-2 the scores along time for train triples (president of the french republic, office holder, {Jacques Chirac | Nicolas Sarkozy | François Hollande | Emmanuel Macron}, $[1980, 2020]$). The periods where a score is highest matches closely the ground truth of start and end dates of these presidents mandates which is represented as a colored background. This shows that our models are able to learn rankings that are correct along time intervals despite our training method only ever sampling timestamps within these intervals.

5.7 Conclusion

Tensor methods have been successful for Knowledge Base completion. In this work, we suggest an extension of these methods to Temporal Knowledge Bases. Our methodology adapts well to the various form of these datasets : point-in-time, beginning and endings or intervals. We show that our methods reach higher performances than the state of the art for similar number of parameters. For several datasets, we also provide performances for higher dimensions. We hope that the gap between low-dimensional and high-dimensional models can motivate further research in models that have increased expressivity at lower number of parameters per entity. Finally, we propose a large scale temporal dataset which we believe to be more realistic than the current benchmarks. We give performances of our methods for low-ranks on this dataset.

We believe that, given its scale, this dataset could also be an interesting addition to non-temporal knowledge base completion.

Chapter 6

Conclusion

The starting point of this thesis was the transfer of methods that worked well for single relation recommender systems [66] to multi-relational link prediction. As shown in Chapter 3, our use of the CP parametrization and a variational form of the weighted nuclear 3-norm have lead to large improvements on standard benchmarks. However, best results are not obtained for "low ranks": on benchmarks with $15k$ entities, we use ranks up to 2000 in order to obtain our best results. Such ranks cannot be used at scales of hundreds of millions of entities, which are the scales of large relational databases such as Wikidata [120].

In order to obtain better performances for low ranks, we turn to the more expressive Tucker parametrization. As we show in Chapter 4, this parametrization is difficult to learn when using Adagrad with a diagonal rescaling of the learning rates. Using links between the Tucker parametrization and a CP parametrization with low-rank but high-dimensional factors, we design a control experiments on which we can study the failures of this diagonal rescaling. This leads to a new algorithm, with which we obtain improved results on several datasets with low embedding sizes.

Finally, the last Chapter 5 shows the flexibility of our approach. Adding timestamps to the data is easily interpreted as a new mode in the tensor, leading to a straightforward generalization of ComplEx. This model beats the previous state of the art in temporal knowledge base completion. By adding a component for non-temporal predicate, we obtain even better performances. Searching for regularizers that are variational forms

of weighted nuclear norms leads to improved performances of these models.

In this thesis, we advanced the state of the art on various knowledge base completion benchmarks by combining ComplEx [118] with a variational form of the tensor nuclear norm. This variational form is versatile and can be easily adapted to other problems, as we show by adding a temporal mode to our models. Our approach is conceptually simple and has few hyper-parameters. The rank of the model can be set to the computational limits, and all capacity control defered to the regularization coefficient. On all benchmarks currently in use in the knowledge base completion community, our method reaches the state of the art in a matter of minutes. However, these benchmarks are not representative of the difficulties of working with web-scale multi-relational data, with millions of entities and billions of positive examples. Future research should be focused towards learning on these larger scale datasets. Two issues arise with larger scale, first, the linear growth of model size makes the operational points investigated in our work inaccessible. Second, computing the full cross-entropy over all entities for each positive example becomes infeasible. The trade-offs of approximations of this loss or pairwise ranking loss have not yet been sufficiently investigated at these scales to yield a clear solution. The dataset we released with our latest work can be used to benchmark advances in these two directions, since its scale makes our approach more difficult (with training times of about 7 hours for small embedding sizes) but not impossible.

Appendix A

Appendix for chapter 3

A.1 DistMult on hierarchical predicates

In this section, we justify the observed performances of the Distmult algorithm when used on the hierarchical Wordnet dataset.

Suppose we are trying to embed a single hierarchical predicate p which is a n-ary tree of depth d. This tree will have n^d leaves, 1 root and $K_n^d = \frac{n^d - n}{n-1} \sim n^{d-1}$ internal nodes. We forget any modeling issues we might have, but focus on the symmetricity assumption in `Distmult`.

Leaves and the root only appear on one side of the queries (i, p, j) and hence won't have any problems with the symmetricity. We now focus on an internal node i. It has n children $(c_k^i)_{k=1..n}$ and one ancestor a_i. Assuming $n > 2$, the MRR associated with this node will be higher if the query $(i, p, ?)$ yields the ranked list $[c_1^i, ..., c_n^i, a_i]$. Indeed, the filtered rank of the n queries (i, p, c_k^i) will be 1 while the filtered rank of the query (a_i, p, i) will be $n + 1$.

Counting the number of queries for which the filtered rank is 1, we see that they far outweigh the queries for which the filtered rank is $n + 1$ in the final filtered MRR. For each internal nodes, n queries lead to a rank of 1, and only 1 to a rank of $n + 1$. For the root, n queries with a rank of 1, for the leaves, n^d queries with a rank of 1.

Our final filtered MRR is :

$$mrr = \frac{n^d + n + K_n^d \frac{1}{n+1} + K_n^d n}{n^d + n + (n+1)K_n^d} = 1 - \underbrace{K_n^d \frac{n/(n+1)}{n^d + n + (n+1)K_n^d}}_{\sim \frac{1}{2n}} \to 1$$

Hence for big hierarchies such as hyponym or hypernym in `WN`, we expect the filtered MRR of `DistMult` to be high even though its modeling assumptions are incorrect.

A.2 Proofs

Lemma 2.

$$\min_{u \in \mathcal{U}_R(\boldsymbol{X})} \frac{1}{3} \sum_{r=1}^{R} \sum_{d=1}^{3} \|u_r^{(d)}\|_p^\alpha = \min_{u \in \mathcal{U}_R(\boldsymbol{X})} \sum_{r=1}^{R} \sqrt[3]{\prod_{d=1}^{3} \|u_r^{(d)}\|_p^\alpha}.$$

Moreover, the minimizers are the same and satisfy:

$$\|u_r^{(d)}\|_p = \sqrt[3]{\prod_{d=1}^{3} \|u_r^{(d)}\|_p}.$$

Proof. First, we characterize the minima :

$$\begin{aligned}
&\min \left\{ \Omega_p^\alpha(u) \;\middle|\; u \in \mathcal{U}_R(\boldsymbol{X}) \right\} \\
=& \min \left\{ \Omega_p^\alpha(\tilde{u}) \;\middle|\; \tilde{u}_r^d = c_r^d u_r^d, u \in \mathcal{U}_R(\boldsymbol{X}), \prod_{d=1}^{3} c_r^d = 1 \right\} \\
=& \min \left\{ \frac{1}{3} \sum_{r=1}^{R} \sum_{d=1}^{3} (|c_r^d| \|u_r^{(d)}\|_p)^\alpha \;\middle|\; \tilde{u}_r^d = c_r^d u_r^d, u \in \mathcal{U}_R(\boldsymbol{X}), \prod_{d=1}^{3} c_r^d = 1 \right\}
\end{aligned}$$

We study a summand, for $c_i, a_i > 0$:

$$\min_{\prod_{d=1}^{3} c^d = 1} \frac{1}{3} \sum_{d=1}^{3} (c_d a_d)^\alpha$$

Using constrained optimization techniques, we obtain that this minimum is obtained

for :

$$c_i = \frac{\sqrt[3]{\prod_{d=1}^{3} a_d}}{a_i}$$

and has value $(\prod_{d=1}^{3} a_d)^{\alpha/3}$, which completes the proof. □

Proposition 3. *The function over third order-tensors of* $\mathbb{R}^{N_1 \times N_2 \times N_3}$ *defined as*

$$|||\boldsymbol{X}||| = \min \left\{ \|\sigma\|_{2/3} \;\middle|\; (\sigma, \tilde{u}) \in \mathcal{U}_R(\boldsymbol{X}), R \in \mathbb{N} \right\}$$

is not convex.

Proof. We first study elements of $\mathbb{R}^{2\times 2\times 1}$, tensors of order 3 associated with matrices of size 2×2. We have that

$$\left|\left|\left| \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \right|\right|\right| = \left|\left|\left| \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \right|\right|\right| = 1$$

Let $A = \frac{1}{2} I_2$, the mean of these two matrices. Identifying A with a $2 \times 2 \times 1$ tensor $\boldsymbol{A}$ to obtain the decomposition (σ, u) yielding $|||\boldsymbol{A}|||$, we have that the matrix A can be written as $\sum_{r=1}^{R} \sigma_r u_r^{(1)} \otimes u_r^{(2)}$. This comes from the fact that $u_r^{(3)}$ is a normalized 1×1 vector, so its only entry is equal to 1. We then write that trace $\mathrm{Tr}(A) = \sum_{r=1}^{R} \sigma_r \mathrm{Tr}(u_r^{(1)} \otimes u_r^{(2)}) \leq \sum_{r=1}^{R} \sigma_r$ by Cauchy-Schwarz. Hence $\|\sigma\|_1 \geq \mathrm{Tr}(A) = 1$. Moreover, we have $\|\sigma\|_{2/3} \geq \|\sigma\|_1$ with equality only for σ with at most one non-zero coordinate. Since A is of rank 2, its representation has at least 2 non-zero coordinates, hence $|||A||| = \|\sigma\|_{2/3} > 1$, which contradicts convexity. This proof can naturally be extended to tensors of any sizes. □

A.3 Best Published results

We report in Table A.1 the references for each of the results in Table 2 in the article.

Model	Metric	Result	Reference
WN18	MRR	0.94	Trouillon et al. [118]
	H@10	**0.97**	Ma et al. [80]
WN18RR	MRR	0.46	Dettmers et al. [30]
	H@10	**0.51**	Dettmers et al. [30]
FB15K	MRR	0.84	Kadlec et al. [60]
	H@10	**0.93**	Shen et al. [104]
FB15K-237	MRR	0.32	Dettmers et al. [30]
	H@10	0.49	Dettmers et al. [30]
YAGO3-10	MRR	0.52	Dettmers et al. [30]
	H@10	0.66	Dettmers et al. [30]

Table A.1: References for the *Best Published* row in Table 3.2

Appendix B

Appendix for chapter 4

B.1 Tensor parametrizations

B.1.1 CP-Tucker

We have $X = [\![C; U, V, W]\!]$ and $C = [\![P_1, P_2, P_3]\!]$. For all i, j, k, we have:

$$\begin{aligned}
X_{i,j,k} &= \sum_{r_1,r_2,r_3}^{d} C_{r_1,r_2,r_3} U_{i,r_1} V_{i,r_2} W_{k,r_3} \\
&= \sum_{r_1,r_2,r_3}^{d} \left(\sum_{s=0}^{D} [P_1]_{r_1,s} [P_2]_{r_2,s} [P_3]_{r_3,s} \right) U_{i,r_1} V_{i,r_2} W_{k,r_3} \\
&= \sum_{s=0}^{D} \left(\sum_{r_1}^{d} U_{i,r_1} [P_1]_{r_1,s} \right) \left(\sum_{r_2}^{d} V_{j,r_2} [P_2]_{r_2,s} \right) \left(\sum_{r_3}^{d} W_{k,r_3} [P_3]_{r_3,s} \right) \quad = \langle P_1 u_i, P_2 v_j, P_3 w_k \rangle
\end{aligned}$$

B.1.2 Tucker2 with CP-Tucker

Let for all $1 \leq r \leq d$, $M^{(r)}$ be a matrix of zeros except its $r - th$ column which is all one. Let P_1 be the vertical concatenation of all $(M^{(r)})_{r=1..d}$ and P_2 the vertical concatenation of d identity matrix in $\mathbb{R}^d$. Remember that for all k, w_k is an element of $\mathbb{R}^{d^2}$. For all $0 \leq r < d$, let w_k^r be the restriction of w_k to its $[rd, (r+1)d]$ coordinates.

Then P_1 and P_2 are elements of $\mathbb{R}^{d^2 \times d}$ and we have for all i, j, k:

$$\begin{aligned}
\langle P_1 u_i, P_2 v_j, w_k \rangle &= \sum_{r_1=1}^{d} \langle M^{(r_1)} u_i, I_d v_j, w_k^{r_1} \rangle \\
&= \sum_{r_1=1}^{d} u_{i,r_1} \langle v_j, w_k^{r_1} \rangle \quad \text{by definition of } M^{(r_1)} \\
&= \sum_{r_1=1}^{d} u_{i,r_1} \left(\sum_{r_2=1}^{d} v_{j,r_2} w_{k,r_1 r_2} \right) \quad \text{by definition of } w_k^{r_1} \\
&= u_i^\top \mathrm{Mat}(w_k) v_j
\end{aligned}$$

B.1.3 HolEx and Latent factor model

HolEx

The HolEx model [125] writes $X_{i,j,k} = \sum_{r=1}^{R} \langle (c_r \odot u_i) \star u_j, w_k^r \rangle$, where $\star$ denotes the circular correlation[1]. Exploiting the equivalence between HolE and ComplEx [51], denoting by $\mathcal{F}$ the discrete Fourier transform (with values in $\mathbb{C}$), we can write for embeddings of size d:

$$\begin{aligned}
\sum_{r=1}^{R} \langle (c_r \odot u_i) \star u_j, w_k^r \rangle &= \frac{1}{d} Re \left(\sum_{r=1}^{R} \langle \overline{\mathcal{F}(c_r \odot u_i)}, \mathcal{F}(u_j), \mathcal{F}(w_k^r) \rangle \right) \\
&= \frac{1}{d} Re \left(\sum_{r=1}^{R} \langle \overline{\mathcal{F}(c_r) \star \mathcal{F}(u_i)}, \mathcal{F}(u_j), \mathcal{F}(w_k^r) \rangle \right)
\end{aligned}$$

For all vectors, we write $\hat{u}_i = \mathcal{F}(u_i) \in \mathbb{C}^d$. We can re-write the circular correlation $\mathcal{F}(c_r) \star \mathcal{F}(u_i)$ as $C_r \hat{u}_i$ where $C_r \in \mathbb{C}^{d \times d}$ is the circulant matrix associated with $c_r \in \mathbb{R}^d$. We have :

$$\sum_{r=1}^{R} \langle (c_r \odot u_i) \star v_j, w_k^r \rangle = \frac{1}{d} Re \left(\sum_{r=1}^{R} \langle C_r \overline{\hat{u}_i}, \hat{u}_j, \hat{w}_k^r \rangle \right).$$

[1] $[a \star b]_k = \sum_{i=0}^{d-1} a_i b_{(i+k) \bmod d}$.

Finally, with $C^1 = [C_1, \ldots, C_R] \in \mathbb{R}^{Rd \times d}$ the vertical stacking of all C_r, $C^2 = [I_d, \ldots, I_d]$ and $\hat{w}_k = [\hat{w}_k^1, \ldots, \hat{w}_k^R]$:

$$\sum_{r=1}^{R} \langle (c_r \odot u_i) \star u_j, w_k^r \rangle = \frac{1}{d} Re\left(\langle C^1 \overline{\hat{u}_i}, C^2 \hat{u}_j, \hat{w}_k \rangle \right)$$

HolEx with embeddings of size d is close to the CPT with $D = Rd$, allowing for two different complex matrices to act on left and right hand side embeddings.

Latent Factor model

The latent factor model defines the score for a triple (i, j, k) as:

$$X_{i,j,k} = \langle (s_i + z), R_j(o_k + z') \rangle, \quad \text{with } R_j = \sum_{r=1}^{D} \alpha_r^j u_r v_r^\top.$$

Removing the bias terms z and z' and gathering u_r and v_r into matrices P_1 and P_3 leads to the model CPT. In the PCP model, we fix P_1 and P_3 instead of learning them. We do not use a sparsity inducing penalty on α but rather a variational form of the nuclear norm on the whole tensor.

B.2 The Adagrad algorithm

In subsections B.2.1 and B.2.2 where we discuss the Adagrad algorithm, we do so in a general setting, where we study, for fixed P with orthonormal columns, the problem $\min_\theta f(P\theta)$.

B.2.1 Projected Adagrad update

Let D denote $\mathrm{Diag}(G)^{1/2}$ for the classical version of Adagrad and $G^{1/2}$ itself for the "full" version of the update. When the parameter θ is constrained to a set Θ, the update proposed in Eq.(1) in [32] the one obtained by solving

$$\min_{\theta \in \Theta} (\tilde{\theta} - z)^\top D (\tilde{\theta} - z) \quad \text{with} \quad z = \tilde{\theta}^{(t)} - \eta D^{-1} g^{(t)}$$

To enforce the constraint that $\tilde{\theta} = P\theta$, we can consider the Lagrangian

$$\mathcal{L}(\tilde{\theta}, \theta; \lambda) = (\tilde{\theta} - z)^\top D(\tilde{\theta} - z) - \lambda^\top(\tilde{\theta} - P\theta)$$

whose stationary points satisfy $D(\tilde{\theta} - z) = \lambda \quad \text{and} \quad P^\top \lambda = 0$. So this entails $P^\top D(\tilde{\theta} - \tilde{\theta}^{(t)}) = \eta P^\top g^{(t)}$ and finally using $\tilde{\theta} = P\theta$ we obtain an update in θ as follows

$$\theta^{(t+1)} = \theta^{(t)} - \eta(P^\top D P^\top)^{-1} P g^{(t)}.$$

Clearly, $PDP^\top$ is in general non-diagonal whether D is diagonal or not, and so this approach does not provide a computationally efficient update.

If $D = G^{1/2}$, then since $PG^{1/2}P^\top = (PGP^\top)^{1/2}$ the update is the same as the full Adagrad update (B.1) that we derive in the following section and replacing $(PGP^\top)^{1/2}$ by its diagonal approximation recovers update (4.3).

B.2.2 The two Full Adagrad updates and the quality of approximations of the Diag versions

If we consider the full versions of the Adagrad updates then (letting again $\Pi = PP^\top$) its application to $\theta \mapsto f(P\theta)$ and $\tilde{\theta} \mapsto f(\Pi\tilde{\theta})$ yield respectively

$$\tilde{\theta}^{(t+1)} = \tilde{\theta}^{(t)} - \eta\, P\big(P^\top G^{(t)} P\big)^{-1/2} P^\top g^{(t)} \quad \text{and} \tag{B.1}$$

$$\tilde{\theta}^{(t+1)} = \tilde{\theta}^{(t)} - \eta\, \Pi\, \big(\Pi\, G^{(t)}\, \Pi\big)^{-\dagger/2}\, \Pi\, g^{(t)}, \tag{B.2}$$

where $M^\dagger$ notes the pseudo-inverse of a matrix M. As it turns out, the two updates are equivalent:

Indeed, first $(\Pi G \Pi)^{1/2} = P(P^\top G P)^{1/2} P^\top$ because $P^\top P = I$ implies that

$$P(P^\top G P)^{1/2} P^\top P (P^\top G P)^{1/2} P^\top = \Pi G \Pi,$$

and the p.s.d. squareroot is unique. Second, taking the pseudo-inverse of this identity,

we have

$$(\Pi G \Pi)^{\dagger/2} = \left(P(P^\top G P)^{1/2} P^\top\right)^\dagger = P(P^\top G P)^{-1/2} P^\top. \tag{B.3}$$

because, if $H = (P^\top G P)^{1/2}$ is an invertible matrix, then $(PHP^\top)^\dagger = PH^{-1}P^\top$ given that $PHP^\top\, PH^{-1}P^\top = PP^\top$. Finally multiplying both sides of Eq. (B.3) by Π shows that

$$\Pi(\Pi G^{(t)} \Pi)^{\dagger/2} \Pi = P(P^\top G^{(t)} P)^{-1/2} P^\top.$$

This shows that although Adagrad is not invariant for any invertible P, it is invariant for any P such that $P^\top P = I$. Eq.(B.1) seems in general simpler than (B.2), but note that if $D = d$, then Π is the identity and (B.2) shows that both full updates are in that case actually equivalent to the full update of Adagrad applied to plain CP.

Finally, the equivalence of the full updates discussed above strongly suggests that if our proposed update performs better than the naive application of Adagrad to PCP, it is because $P\mathrm{Diag}(\Pi G^{(t)} \Pi)^{-1/2} P^\top$ is a better approximation of $(G^{(t)})^{-1/2}$ than $\mathrm{Diag}(PG^{(t)}P^\top)^{-1/2}$ while not being much more computationally expensive.

B.2.3 Complete PComplEx Algorithm

Let $U \in \mathbb{C}^{N \times d}$ be the the entity embedding matrix and $V \in \mathbb{C}^{2L \times D}$ be the predicate embedding matrix. Let $\mathcal{P} : \mathbb{R}^{N \times d} \mapsto \mathbb{R}^{N \times D}$ such that $\mathcal{P}(U_i) = PU_i$. Let G_t^U, G_t^V be the stochastic gradients with respect to U, V at time t.

Algorithm 3 PComplEx optimized with $\mathrm{Ada}^{\mathrm{imp}}$

Input: learning rate η, (random) matrix P with orthogonal columns, ϵ
  $C_U, C_V \leftarrow 0$
  while U, V not converged **do**
    $\tilde{G}_t^U \leftarrow \mathcal{P}(G_t^U)$
    $C_U \leftarrow C_U + \tilde{G}_t^U \odot \tilde{G}_t^U$
    $C_V \leftarrow C_V + G_t^V \odot G_t^V$
    $U \leftarrow U - \eta \cdot \mathcal{P}^\top(\tilde{G}_t^U / (\sqrt{C_U + \epsilon}))$
    $V \leftarrow V - \eta \cdot G_t^V / (\sqrt{C_V + \epsilon}))$
  end while
  return U

B.2.4 Adam - Implicit

Algorithm 4 Adam^{imp} applied to U

Input: $\eta, \beta_1, \beta_2, \mathcal{P}, \epsilon$
  $m_0, v_0, t \leftarrow 0$
  while U not converged **do**
    $t \leftarrow t + 1$
    $\tilde{G}_t \leftarrow \mathcal{P}(G_t)$
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot \tilde{G}_t$
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot \tilde{G}_t \odot \tilde{G}_t$
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
    $U \leftarrow U - \eta \cdot \mathcal{P}^\top (\hat{m}_t / (\sqrt{\hat{v}_t + \epsilon}))$
  end while
  return U

For WN18RR, we used the ideas presented in this paper, but adapted to the Adam [63] optimization algorithm. Similar to Ada$^{\text{imp}}$, first and second moment estimates are accumulated in $\mathbb{R}^D$ and we project back to $\mathbb{R}^d$ only for the parameter update. For simplicity, we present in algorithm 4 the dense version of the algorithm applied to the entity embeddings $U \in \mathbb{R}^{N \times d}$. Let $\mathcal{P} : \mathbb{R}^{N \times d} \mapsto \mathbb{R}^{N \times D}$ such that $\mathcal{P}(U_i) = PU_i$. Let G_t be the stochastic gradient with respect to U at time t.

B.3 Experiments

B.3.1 Dataset statistics

B.3.2 Ada$^{\text{row}}$

For the same control experiment as in Section 4.5, we observe the performances of Ada$^{\text{row}}$ which is rotation invariant by design.

Dataset	N	P	Train
FB15K	15k	1k	500k
FB15K-237	15k	237	272k
WN18	41k	18	141k
WN18	41k	11	141k
YAGO3-10	123k	37	1M
SVO	30k	4.5k	1M

Table B.1: Dataset statistics.

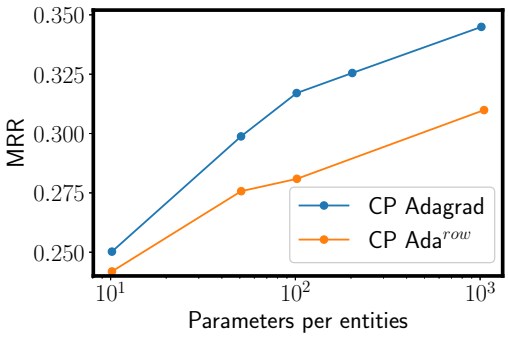
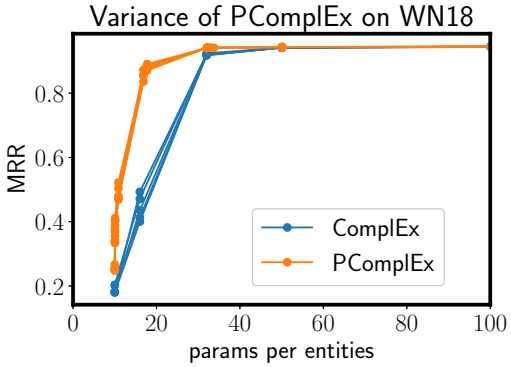


Figure B-1: Ada$^{\text{row}}$ vs AdaGrad on FB15K-237.

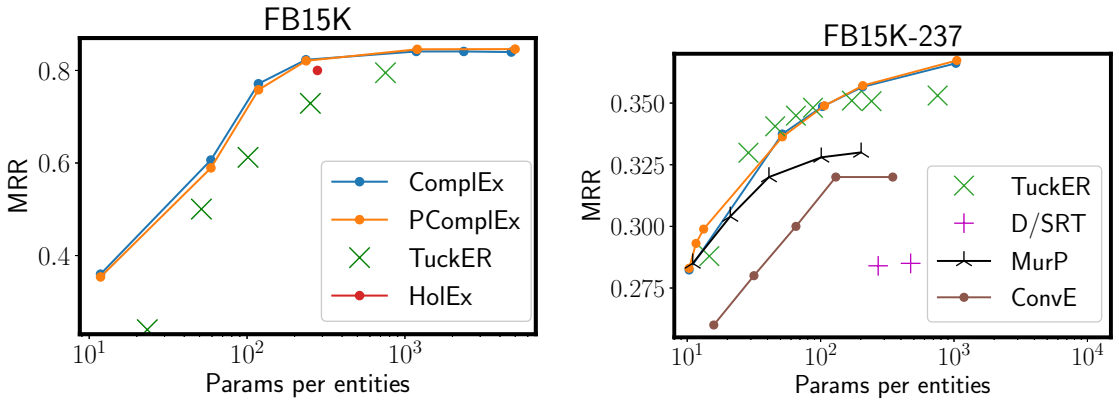
B.3.3 Variance of PComplEx

We run 5 grid search and plot the 5 associated convex-hulls on the WN18 dataset. Note that despite the added randomness of the choice of P, there is not more variance in PComplEx than in ComplEx.



B.3.4 FB15k datasets

We use two subsets of the Freebase knowledge base : FB15K [13] and FB15K-237 [117]. FB15K-237 is a harder version of FB15K, where some triples have been removed to avoid leakage between the train and test set. There is no difference of performances between PComplEx and ComplEx on these datasets.



B.3.5 Experimental details

Metrics Let $rank(\hat{X}_{i,j,:}; k)$ be the rank of $\hat{X}_{i,j,k}$ in the sorted list of values of $\hat{X}_{i,j,:}$. We report the MRR for most datasets :

$$MRR(X) = \frac{1}{|S|} \sum_{(i,j,k) \in S} \frac{1}{rank(X_{i,j,:}; k)}.$$

For SVO, the task is slightly different as the ranking happens on the predicate mode. The metric reported in Jenatton et al. [56] is Hits@5% defined as :

$$H@5\%(X) = \frac{1}{|S|} \sum_{(i,j,k) \in S} \mathbb{1}(rank(X_{i,:,k}; j) \leq 227).$$

The metrics we report are filtered by ignoring other true positives when computing the ranks, as done in Bordes et al. [13].

Number of parameters per entities We count the number of floats in the model, and divide by the number of entities in the dataset. For different methods, the number

of parameters are :

- ComplEx: $2 \cdot d \cdot (N + 2L)$
- PComplEx: $2 \cdot d \cdot N + 2 \cdot D \cdot 2L + d \cdot D$
- TuckEr: $N \cdot d_e + L \cdot d_p + d_e^2 \cdot d_p$
- MurP: $N \cdot (d + 1) + 2 \cdot L \cdot d$
- ConvE: taken from Dettmers et al. [30]
- D/SRT: taken from Wang et al. [121]
- HolEx: $d \cdot (N + L)$

B.3.6 Grid Search

Grid Search For SVO:

- For Complex vary d in $[5, 25, 50, 100, 500, 1000, 2000]$. For PComplEx, we vary d in $[5, 25, 50, 100, 500]$.
- The strength of the regularizer, ν varies in $[5e{-}4, 1e{-}3, 5e{-}3, 1e{-}2, 5e{-}2, 1e{-}1]$.
- Finally, for PComplEx, we vary the dimension D in $[5, 25, 50, 100, 500, 1000, 2000, 4000, 8000]$.

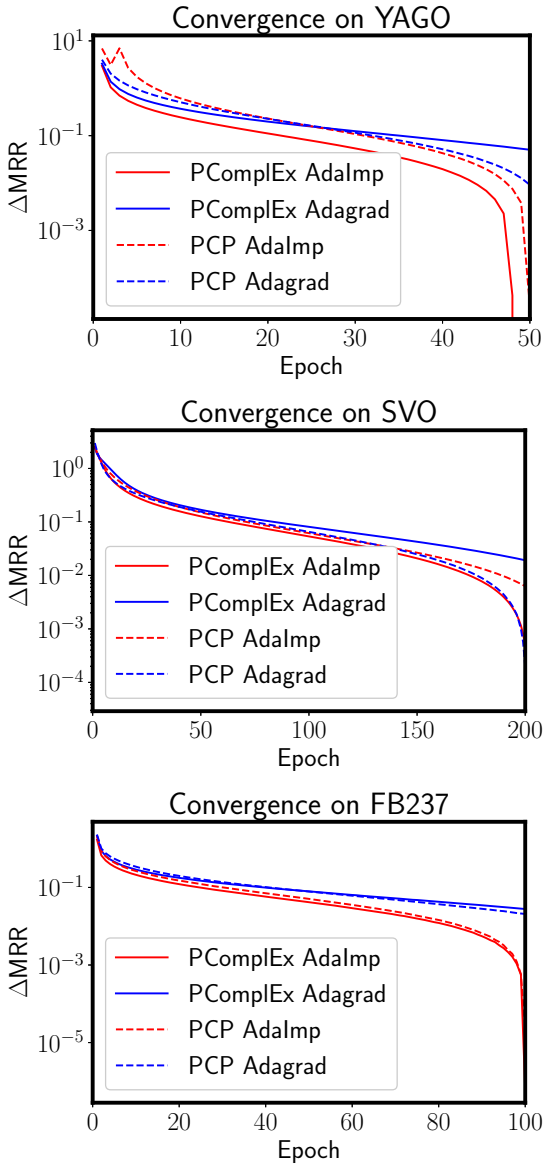
For all other datasets :

- For FB15K and FB15K-237, we vary d in $[5, 25, 50, 100, 500, 1000, 2000]$. For YAGO3-10, WN18 and WN18RR, we add ranks 8 and 16 to that list.
- The strength of the regularizer, ν varies in $[5e{-}4, 1e{-}3, 5e{-}3, 1e{-}2, 5e{-}2, 1e{-}1]$.
- Finally, for PComplEx, we vary the dimension D in $[5, 25, 50, 100, 500, 1000, 2000]$.

Other details Batch-size is fixed to 1000, the learning-rate is $1e-1$ both for Adagrad and for Ada$^{\text{imp}}$, we run 100 epochs to ensure convergence. The best model based on validation MRR is selected and we report the corresponding test MRR.

B.3.7 Convergence speed

We run a grid-search on learning rates in $[10, 5, 1, 0.5, 0.1, 0.05]$ and select the best final performance for each model and each method. For each model, we plot in log-scale the difference between the loss and the best final loss.



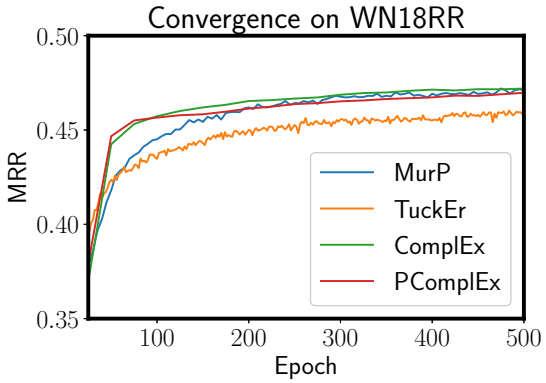
B.3.8 Running times

We give here the running times of each method per epoch, on the WN18RR dataset for comparable dimensionalities and run on a P100 GPU. We use the original imple-

mentation for MurP [6] and TuckEr [7]. For ComplEx, we use the implementation from Lacroix et al. [70].

- ComplEx ($d = 200$) : 4s/epoch.
- PComplEx ($d = D = 200$, Ada$^{\text{imp}}$) : 5s/epoch.
- MurP ($d = 200$): 38s/epoch.
- TuckEr ($d_e = 200,\ d_r = 200$): 24s/epoch

Note that these running times are implementation dependent. In the figure below, we give the learning-curves of MurP, TuckEr, ComplEx and PComplEx for one operating point on the WN18RR dataset. The convergence speed of these methods is given in the figure below at an operating point on WN18RR where all methods are close in final performances.



Appendix C

Appendix for chapter 5

C.1 Tensor norms

C.1.1 Unfolding and the CP parametrization

Let $X = [\![U, V, W, T]\!]$, that is $X_{i,j,k,l} = \langle u_i, v_j, w_k, t_l \rangle$. Then according to Kolda and Bader [64], unfolding along modes 3 and 4 leads to an order three tensor of decomposition $\tilde{X} = [\![U, V, W \circ T]\!]$. Where $\circ$ is the Khatri-Rao product [106], which is the column-wise Kronecker product : $W \circ T = (W_{:,1} \otimes T_{:,1}, \ldots, W_{:,R} \otimes T_{:,R})$.

Note that for a fourth mode of size L: $(W \circ T)_{L(k-1)+l} = w_k \odot t_l$. This justifies the regularizers used in Section 5.3.2.

C.1.2 Temporal regularizer and Nuclear norms

Consider the penalty:

$$\Omega(U, V, W, T) = \frac{1}{4} \left(\|U\|_4^4 + \|V\|_4^4 + \|W\|_4^4 + \|T\|_4^4 + \alpha \|T_{1:} - T_{:-1}\|_4^4 \right)$$

Let us define a new norm on vectors:

$$\|t\|_{\tau 4} = \left(\|t\|_4^4 + \alpha \|t_{1:} - t_{:-1}\|_4^4 \right)^{1/4}$$

$\|\cdot\|_{\tau 4}$ is a norm and lets us rewrite:

$$\Omega(U,V,W,T) = \sum_{r=1}^{R} \frac{1}{4}\left(\|u_r\|_4^4 + \|v_r\|_4^4 + \|w_r\|_4^4 + \|t_r\|_{\tau 4}^4\right).$$

Following the proof in Lacroix et al. [70] which only uses homogeneity of the norms, we can show that $\Omega(U,V,W,T)$ is a variational form of an atomic norm with atoms :

$$\mathcal{A} = \{u \otimes v \otimes w \otimes t \mid \|u\|_4, \|v\|_4, \|w\|_4 \leq 1 \text{ and } \|t\|_{\tau 4} \leq 1\}$$

C.1.3 Nuclear norms on unfoldings

We consider the regularizer :

$$\Omega^{N3}(U,V,T;(i,j,k,l)) = \frac{1}{3}\left(\|u_i\|_3^3 + \|u_k\|_3^3 + \|v_k \odot t_l\|_3^3\right).$$

Let D^{subj} (resp. obj, pred/time) the diagonal matrix containing the cubic-roots of the marginal probabilities of each subject (resp. obj, pred/time) in the dataset. We denote by $\circ$ the Kathri-Rao product between two matrices (the columnwise Kronecker product). Summing over the entire dataset, we obtain the penalty:

$$\frac{1}{|S|}\sum_{(i,j,k,l)\in S} \Omega^{N3}(U,V,T;(i,j,k,l)) = \frac{1}{3}\left(\|D^{\text{subj}}U\|_3^3 + \|D^{\text{obj}}U\|_3^3 + \|D^{\text{pred/time}}(V \circ T)\|_3^3\right).$$

Dropping the weightings to simplify notations, we state the equivalence between this regularizer and a variational form of the nuclear 3-norm of an order 4 tensor:

$$\inf_{[U_1,U_2,U_3,U_4]=X} \frac{1}{3}\left(\sum_{r=1}^{R} \|u_r^{(1)}\|_3^3 + \|u_r^{(2)}\|_3^3 + \|u_r^{(3)} \otimes u_r^{(4)}\|_3^3\right) = \inf_{[U_1,U_2,U_3,U_4]=X} \frac{1}{3}\left(\sum_{r=1}^{R}\prod_{d=1}^{4} \|u_r^{(d)}\|_3\right).$$

The proof follows Lacroix et al. [70], noting that $\|u_r^{(3)} \otimes u_r^{(4)}\|_3^3 = \|u_r^{(3)}\|_3^3 \|u_r^{(4)}\|_3^3$. Note that for $D^{\text{pred/time}} = D^{\text{pred}} D^{\text{time}}$, there would also be equality of the weighted norms. However, in the application considered, time and predicate are most likely not

independent, leading to different weightings of the norms.

C.2 Experiments

C.2.1 Dataset statistics

Statistics of all the datasets used in this work are gathered in Table 3.1.

	ICEWS14	ICEWS05-15	Yago15k	Wikidata
Entities	6869	10094	15403	432715
Predicates	460	502	102	814
Timestamps	365	4017	170	1726
|S|	72826	368962	110441	7224361

Table C.1: Dataset statistics

C.2.2 Grid Search

For ICEWS14, ICEWS05-15 and Yago15k, we follow the grid-search below :

Using Table 5.1 to compute the number of parameters and the dataset statistics in Table 3.1, we use the following ranks to match the number of parameters of DE-SimplE in dimension 100:

	ICEWS14	ICEWS05-15	Yago15k
DE-SimplE	100	100	100
ComplEx	182	186	196
TComplEx	174	136	194
TTComplEx	156	128	189

Bibliography

[1] Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 670–688. IEEE, 2015.

[2] Naman Agarwal, Brian Bullins, Xinyi Chen, Elad Hazan, Karan Singh, Cyril Zhang, and Yi Zhang. The case for full-matrix adaptive regularization. *arXiv preprint arXiv:1806.02958*, 2018.

[3] Noga Alon. Tools from higher algebra. *Handbook of combinatorics*, 2:1749–1783, 1995.

[4] Brett W Bader, Richard A Harshman, and Tamara G Kolda. Temporal analysis of semantic graphs using asalsan. In *Seventh IEEE international conference on data mining (ICDM 2007)*, pages 33–42. IEEE, 2007.

[5] Raphaël Bailly, Antoine Bordes, and Nicolas Usunier. Semantically Invariant Tensor Factorization. 2015.

[6] Ivana Balazevic, Carl Allen, and Timothy Hospedales. Multi-relational poincaré graph embeddings. In *Advances in Neural Information Processing Systems*, pages 4465–4475, 2019.

[7] Ivana Balazevic, Carl Allen, and Timothy Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5188–5197, 2019.

[8] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *arXiv preprint arXiv:1811.04820*, 2018.

[9] James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, NY, USA., 2007.

[10] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Semi-supervised novelty detection. *Journal of Machine Learning Research*, 11(Nov):2973–3009, 2010.

[11] Ron Blei. *The Grothendieck inequality revisited*, volume 232. American Mathematical Society, 2014.

[12] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. Learning structured embeddings of knowledge bases. In *Conference on artificial intelligence*, 2011.

[13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating Embeddings for Modeling Multi-relational Data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc., 2013.

[14] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[15] Elizabeth Boschee, Jennifer Lautenschlager, Sean O'Brien, Steve Shellman, James Starz, and Michael Ward. Icews coded event data. *Harvard Dataverse*, 12, 2015.

[16] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Reviews*, 60(2):223–311, 2018. URL `http://leon.bottou.org/papers/bottou-curtis-nocedal-2018`.

[17] Rasmus Bro and Claus A Andersson. Improving the speed of multiway algorithms: Part ii: Compression. *Chemometrics and intelligent laboratory systems*, 42(1-2): 105–113, 1998.

[18] Rasmus Bro, ND Sidiropoulos, and GB Giannakis. A fast least squares algorithm for separating trilinear mixtures. In *Int. Workshop Independent Component and Blind Signal Separation Anal*, pages 11–15, 1999.

[19] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014)*, 2014.

[20] Jian-Feng Cai, Emmanuel J Candès, and Zuowei Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4): 1956–1982, 2010.

[21] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717, 2009.

[22] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[23] J Douglas Carroll, Sandra Pruzansky, and Joseph B Kruskal. Candelinc: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. *Psychometrika*, 45(1):3–24, 1980.

[24] Andrew Chatr-Aryamontri, Bobby-Joe Breitkreutz, Rose Oughtred, Lorrie Boucher, Sven Heinicke, Daici Chen, Chris Stark, Ashton Breitkreutz, Nadine Kolas, Lara O'Donnell, et al. The biogrid interaction database: 2015 update. *Nucleic acids research*, 43(D1):D470–D478, 2014.

[25] Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *Journal of Machine Learning Research*, 11(Mar):1109–1135, 2010.

[26] Hao Cheng, Yaoliang Yu, Xinhua Zhang, Eric Xing, and Dale Schuurmans. Scalable and sound low-rank tensor learning. In *Artificial Intelligence and Statistics*, pages 1114–1123, 2016.

[27] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, pages 160–167. ACM, 2008.

[28] Vin De Silva and Lek-Heng Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.

[29] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84(6):066106, 2011.

[30] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2D knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[31] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.

[32] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[33] Takuma Ebisu and Ryutaro Ichise. Toruse: Knowledge graph embedding on a lie group. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[34] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220. ACM, 2008.

[35] Maryam Fazel, Haitham Hindi, Stephen P Boyd, et al. A rank minimization heuristic with application to minimum order system approximation. In *Proceedings of the American control conference*, volume 6, pages 4734–4739. Citeseer, 2001.

[36] Jun Feng, Minlie Huang, Mingdong Wang, Mantong Zhou, Yu Hao, and Xiaoyan Zhu. Knowledge graph embedding by flexible translation. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*, 2016.

[37] Rina Foygel, Ohad Shamir, Nati Srebro, and Ruslan R Salakhutdinov. Learning with the weighted trace-norm under arbitrary sampling distributions. In *Advances in Neural Information Processing Systems*, pages 2133–2141, 2011.

[38] Shmuel Friedland and Lek-Heng Lim. Nuclear norm of higher-order tensors. *Mathematics of Computation*, 87(311):1255–1281, 2018.

[39] Silvia Gandy, Benjamin Recht, and Isao Yamada. Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems*, 27(2):025010, 2011.

[40] Alberto Garcia-Duran, Antoine Bordes, Nicolas Usunier, and Yves Grandvalet. Combining two and three-way embedding models for link prediction in knowledge bases. *Journal of Artificial Intelligence Research*, 55:715–742, 2016.

[41] Alberto Garcia-Duran, Sebastijan Dumančić, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, 2018.

[42] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011.

[43] Rishab Goel, Seyed Mehran Kazemi, Marcus Brubaker, and Pascal Poupart. Diachronic embedding for temporal knowledge graph completion. *arXiv preprint arXiv:1907.03143*, 2019.

[44] Google. Freebase data dumps. `https://developers.google.com/freebase/data`, 2019.

[45] David Gross. Recovering low-rank matrices from few coefficients in any basis. *IEEE Transactions on Information Theory*, 57(3):1548–1566, 2011.

[46] Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *35th International Conference on Machine Learning, ICML 2018*, pages 2956–2964. International Machine Learning Society (IMLS), 2018.

[47] Kelvin Guu, John Miller, and Percy Liang. Traversing knowledge graphs in vector space. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 318–327, 2015.

[48] William L Hamilton, Jure Leskovec, and Dan Jurafsky. Diachronic word embeddings reveal statistical laws of semantic change. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1489–1501, 2016.

[49] Richard A Harshman. Models for analysis of asymmetrical relationships among n objects or stimuli. In *First Joint Meeting of the Psychometric Society and the Society of Mathematical Psychology, Hamilton, Ontario, 1978*, 1978.

[50] Richard A Harshman et al. Foundations of the parafac procedure: Models and conditions for an" explanatory" multimodal factor analysis. 1970.

[51] Katsuhiko Hayashi and Masashi Shimbo. On the equivalence of holographic and complex embeddings for link prediction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 554–559, 2017.

[52] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 60(6):45, 2013.

[53] Frank L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics*, 6(1-4):164–189, 1927.

[54] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[55] Martin Jaggi, Marek Sulovsk, and others. A simple algorithm for nuclear norm regularized problems. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 471–478, 2010.

[56] Rodolphe Jenatton, Nicolas L Roux, Antoine Bordes, and Guillaume R Obozinski. A latent factor model for highly multi-relational data. In *Advances in Neural Information Processing Systems*, pages 3167–3175, 2012.

[57] Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, 2015.

[58] Armand Joulin, Édouard Grave, Piotr Bojanowski, and Tomáš Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, 2017.

[59] Armand Joulin, Edouard Grave, Piotr Bojanowski, Maximilian Nickel, and Tomas Mikolov. Fast linear model for knowledge graph embeddings. *arXiv preprint arXiv:1710.10881*, 2017.

[60] Rudolf Kadlec, Ondrej Bajgar, and Jan Kleindienst. Knowledge base completion: Baselines strike back. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 69–74, 2017.

[61] Seyed Mehran Kazemi and David Poole. Simple embedding for link prediction in knowledge graphs. In *Advances in Neural Information Processing Systems 31*, pages 4289–4300. 2018.

[62] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, page 5, 2006.

[63] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[64] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[65] Tamara Gibson Kolda. Multilinear operators for higher-order decompositions. Technical report, Sandia National Laboratories, 2006.

[66] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[67] Denis Krompass, Stephan Baier, and Volker Tresp. Type-constrained representation learning in knowledge graphs. In *International Semantic Web Conference*, pages 640–655. Springer, 2015.

[68] Gabriel Krummenacher, Brian McWilliams, Yannic Kilcher, Joachim M Buhmann, and Nicolai Meinshausen. Scalable adaptive stochastic optimization using random projections. In *Advances in Neural Information Processing Systems*, pages 1750–1758, 2016.

[69] Michael Kuhn, Ivica Letunic, Lars Juhl Jensen, and Peer Bork. The sider database of drugs and side effects. *Nucleic acids research*, 44(D1):D1075–D1079, 2015.

[70] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *Proceedings of the 35th International Conference on Machine Learning (ICML-18)*, pages 2863–2872, 2018.

[71] Ni Lao, Tom Mitchell, and William W Cohen. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics, 2011.

[72] Adam Lerer, Ledell Wu, Jiajun Shen, Timothee Lacroix, Luca Wehrstedt, Abhijit Bose, and Alex Peysakhovich. Pytorch-biggraph: A large-scale graph embedding system. *arXiv preprint arXiv:1903.12287*, 2019.

[73] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pages 2177–2185, 2014.

[74] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations (ICLR)*, 2016.

[75] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[76] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[77] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[78] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):208–220, 2012.

[79] Canyi Lu, Jiashi Feng, Yudong Chen, Wei Liu, Zhouchen Lin, and Shuicheng Yan. Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5249–5257, 2016.

[80] Shiheng Ma, Jianhui Ding, Weijia Jia, Kun Wang, and Minyi Guo. Transt: Type-based multiple embedding representations for knowledge graph completion. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 717–733. Springer, 2017.

[81] Yunpu Ma, Volker Tresp, and Erik A Daxberger. Embedding models for episodic knowledge graphs. *Journal of Web Semantics*, page 100490, 2018.

[82] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M Suchanek. Yago3: A knowledge base from multilingual wikipedias. 2013.

[83] Gerry McKiernan. arxiv. org: the los alamos national laboratory e-print server. *International Journal on Grey Literature*, 1(3):127–138, 2000.

[84] Jörg Menche, Amitabh Sharma, Maksim Kitsak, Susan Dina Ghiassian, Marc Vidal, Joseph Loscalzo, and Albert-László Barabási. Uncovering disease-disease relationships through the incomplete interactome. *Science*, 347(6224):1257601, 2015.

[85] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[86] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[87] Cun Mu, Bo Huang, John Wright, and Donald Goldfarb. Square deal: Lower bounds and improved relaxations for tensor recovery. In *International conference on machine learning*, pages 73–81, 2014.

[88] Sahand Negahban and Martin J Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *Journal of Machine Learning Research*, 13(May):1665–1697, 2012.

[89] Dat Quoc Nguyen. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv preprint arXiv:1703.08098*, 2017.

[90] Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of NAACL-HLT*, pages 460–466, 2016.

[91] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 809–816, 2011.

[92] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1):11–33, 2016.

[93] Maximilian Nickel, Lorenzo Rosasco, Tomaso A Poggio, et al. Holographic embeddings of knowledge graphs. 2016.

[94] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in Neural Information Processing Systems*, pages 6338–6347, 2017.

[95] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International Conference on Machine Learning*, pages 2014–2023, 2016.

[96] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5): 1–17, 1964.

[97] Benjamin Recht. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(Dec):3413–3430, 2011.

[98] Jasson DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 713–719. ACM, 2005.

[99] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[100] Purnamrita Sarkar and Andrew W Moore. Dynamic social network analysis using latent space models. In *Advances in Neural Information Processing Systems*, pages 1145–1152, 2006.

[101] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

[102] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[103] Yoav Seginer. The expected norm of random matrices. *Combinatorics, Probability and Computing*, 9(2):149–166, 2000.

[104] Yelong Shen, Po-Sen Huang, Ming-Wei Chang, and Jianfeng Gao. Implicit reasonet: Modeling large-scale structured relationships with shared memory. 2016.

[105] Marco Signoretto, Lieven De Lathauwer, and Johan AK Suykens. Nuclear norms for tensors and their use for convex multilinear estimation. *Submitted to Linear Algebra and Its Applications*, 43, 2010.

[106] Age Smilde, Rasmus Bro, and Paul Geladi. *Multi-way analysis: applications in the chemical sciences*. John Wiley & Sons, 2005.

[107] Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

[108] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*, pages 926–934, 2013.

[109] Nathan Srebro. Learning with matrix factorizations. 2004.

[110] Nathan Srebro and Ruslan R Salakhutdinov. Collaborative filtering in a non-uniform world: Learning with the weighted trace norm. In *Advances in Neural Information Processing Systems*, pages 2056–2064, 2010.

[111] Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *International Conference on Computational Learning Theory*, pages 545–560. Springer, 2005.

[112] Nathan Srebro, Noga Alon, and Tommi S Jaakkola. Generalization error bounds for collaborative prediction with low-rank matrices. In *Advances In Neural Information Processing Systems*, pages 1321–1328, 2005.

[113] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336, 2005.

[114] Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. Maximum-margin matrix factorization. In *Advances in Neural Information Processing Systems*, pages 1329–1336, 2005.

[115] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[116] Ryota Tomioka, Kohei Hayashi, and Hisashi Kashima. Estimation of low-rank tensors via convex optimization. *arXiv preprint arXiv:1010.0789*, 2010.

[117] Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66, 2015.

[118] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080, 2016.

[119] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[120] Denny Vrandečić and Markus Krötzsch. Wikidata: a free collaborative knowledge base. 2014.

[121] Yanjie Wang, Samuel Broscheit, and Rainer Gemulla. A relational tucker decomposition for multi-relational link prediction. *arXiv preprint arXiv:1902.00898*, 2019.

[122] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*, 2014.

[123] Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine learning*, 81(1): 21–35, 2010.

[124] Kishan Wimalawarne, Masashi Sugiyama, and Ryota Tomioka. Multitask learning meets tensor factorization: task imputation via convex optimization. In *Advances in Neural Information Processing Systems*, pages 2825–2833, 2014.

[125] Yexiang Xue, Yang Yuan, Zhitian Xu, and Ashish Sabharwal. Expanding holographic embeddings for knowledge completion. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2018.

[126] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*, 2014.

[127] Ming Yuan and Cun-Hui Zhang. On tensor completion via nuclear norm minimization. *Foundations of Computational Mathematics*, 16(4):1031–1068, 2016.

[128] Marinka Zitnik, Monica Agrawal, and Jure Leskovec. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics*, 34(13):457–466, 2018.