Tensor Factorization for Relational Learning

Maximilian Nickel

München 2013

Tensor Factorization for Relational Learning

Maximilian Nickel

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Maximilian Nickel
aus Altötting

München, den 25. Juli 2013

Erstgutachter: Prof. Dr. Volker Tresp
Zweitgutachter: Prof. Dr. Gerhard Weikum
Tag der mündlichen Prüfung: 14. August 2013

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Nickel, Maximilian

Name, Vorname

Ort, Datum

Unterschrift Doktorand/in

Formular 3.2

To Anne, Susanne, and Regina

Contents

1. Introduction **1**
- 1.1. Notation . . . 3
- 1.2. An Introduction to Relational Learning . . . 3
  - 1.2.1. Learning in Relational Domains . . . 3
  - 1.2.2. Relational Learning Tasks . . . 8
  - 1.2.3. Relational Models . . . 10
- 1.3. An Introduction to Tensors and Tensor Factorizations . . . 16
  - 1.3.1. Tensors and the Tensor Product . . . 16
  - 1.3.2. Operations on Tensors . . . 19
  - 1.3.3. Tensor Factorizations . . . 21
- 1.4. Contributions of this Thesis . . . 24

2. A Three-Way Model for Relational Learning **29**
- 2.1. Introduction . . . 29
- 2.2. The RESCAL Factorization . . . 32
- 2.3. Solving Relational Learning Tasks . . . 41
- 2.4. Discussion and Related Work . . . 43
  - 2.4.1. Comparison to Statistical Relational Learning Models . . . 43
  - 2.4.2. Comparison to Tensor Models . . . 44
- 2.5. Computing the Factorization . . . 46
- 2.6. Experiments . . . 49
  - 2.6.1. Collective Learning . . . 50
  - 2.6.2. Entity Resolution . . . 54
  - 2.6.3. Link Prediction on Relational Learning Benchmark Datasets . . . 56
  - 2.6.4. Link-Based Clustering . . . 58
  - 2.6.5. Comparison to DEDICOM . . . 58
- 2.7. Summary . . . 61

3. Large-Scale Relational Learning and Application on Linked Data **63**
- 3.1. Introduction . . . 63
- 3.2. Modeling Linked Data . . . 65
- 3.3. Machine Learning Tasks on Linked Data . . . 66
- 3.4. Complexity Analysis of RESCAL-ALS . . . 69

3.5. Scalable Core Tensor Updates in RESCAL-ALS . . . 72
3.6. Learning from Attributes via Coupled Tensor Factorization . . . 75
3.7. Experiments . . . 79
3.7.1. Runtime Experiments on Synthetic Data . . . 80
3.7.2. Comparative Runtime Experiments . . . 81
3.7.3. Large-Scale Prediction of Unknown Triples . . . 83
3.7.4. Collective Learning on the Semantic Web . . . 85
3.7.5. Learning Taxonomies . . . 87
3.8. Discussion and Related Work . . . 88
3.8.1. Comparison to Tensor Decompositions . . . 89
3.8.2. Comparison to Semantic Web Machine Learning Methods . . . 90
3.9. Summary . . . 91

4. An Analysis of Tensor Models for Learning on Structured Data 93
4.1. Introduction . . . 93
4.2. Theory and Methods . . . 94
4.2.1. Structured Data, the Cartesian, and the Tensor Product . . . 95
4.2.2. Tensor Factorizations . . . 96
4.3. Generalization Bounds for Low-Rank Factorizations . . . 98
4.3.1. Bounds for Zero-One Sign Agreement Loss . . . 99
4.3.2. Bounds for Real-Valued Loss Functions . . . 100
4.3.3. Bounds on the Number of Sign Patterns . . . 101
4.4. The Effect of Structure and Constraints . . . 103
4.4.1. Comparable Tensors . . . 103
4.4.2. Experimental Results . . . 104
4.4.3. Discussion . . . 105
4.5. Related Work . . . 107
4.6. Conclusion . . . 108

5. Conclusion 109
5.1. Summary . . . 109
5.2. Outlook . . . 110

Appendix A. Block-Partitioned Matrix Multiplication 115
A.1. Tensor and Matrix View of RESCAL . . . 115
A.2. Simplification of RESCAL-ALS . . . 116

Appendix B. MLN on U.S. Presidents 117

Appendix C. Complexity Analysis of CP and TUCKER Algorithms 119

Appendix D. Number of Variables for Comparable Tensors 121

Acknowledgements

Over the last few years, many persons contributed greatly to this thesis through their guidance, their advice and their support for which I am very thankful.

First of all, I would like to express my deep gratitude to Prof. Volker Tresp for his supervision of my thesis. Volker provided invaluable guidance and advice over the last years, which helped me to introduce structure and focus into my research. I learned a lot from our long and regular discussions on numerous topics, where Volker's open-mindedness, his patience, and his habit to put things into perspective helped me to form a much deeper understanding of my own research and of machine learning in general. I feel very fortunate to have been advised by Volker and to still benefit from his continued support.

I would also like to thank Prof. Gerhard Weikum for very interesting and fruitful discussions about my research in the context of knowledge bases and for giving me the opportunity to visit his group at the Max-Planck Institute in Saarbrücken, what helped me to look at my work from a different perspective and introduced new thoughts and ideas. I am also very thankful that Gerhard kindly agreed to act as the second examiner of my thesis.

At the Ludwig Maximilian University, I would like to thank Prof. Hans-Peter Kriegel and Dr. Matthias Schubert, who already advised me during my Diploma and continued to provide advice and support during the completion of this thesis. I would also like to thank Prof. Hans Jürgen Ohlbach for agreeing to serve on my dissertation committee. At Siemens, I had the pleasure to work with Xueyan Jiang, Denis Krompass and Yi Huang as my lab mates and as my colleagues in numerous meetings. I am also thankful to Christa Singer at Siemens as well as Susanne Grienberger and Ulrike Robeck at LMU who helped me to navigate around the bureaucratic difficulties in industry and academia alike. I would also like to thank Rainer Gemulla, Pauli Miettinen, Florian Steinke, Fabian Suchaneck, Mohammed Yahya, Johannes Hoffart, Markus Bundschus and Siegmund Duell for interesting collaborations and stimulating thoughts.

More than anyone else, I owe thanks to my family for their love and continued support: My mother Gerda, my father Bernd, and my sisters Susanne and Regina. In particular, I would like to thank Susanne for teaching me to think creatively, Peter for introducing me to physics and science, and Regina for her support and valuable advice over the last years. Susanne and Peter – you know how grateful I am for all that you did for me throughout the years. I would also like to thank Irmgard, Renate, and Rosmarie for their dedication and dependability, what rendered difficult circumstances much easier. My deepest thanks go to my wife Anne. Not only did her sharp eye and tireless mind correct many of my careless errors and creative usages of grammar and spelling in this thesis, but her love, her patience, and her support allowed me to pursue my research freely and as a happy and content person throughout all these years.

Abstract

Relational learning is concerned with learning from data where information is primarily represented in form of relations between entities. In recent years, this branch of machine learning has become increasingly important, as relational data is generated in an unprecedented amount and has become ubiquitous in many fields of application such as bioinformatics, artificial intelligence and social network analysis. However, relational learning is a very challenging task, due to the network structure and the high dimensionality of relational data. In this thesis we propose that tensor factorization can be the basis for scalable solutions for learning from relational data and present novel tensor factorization algorithms that are particularly suited for this task.

In the first part of the thesis, we present the RESCAL model – a novel tensor factorization for relational learning – and discuss its capabilities for exploiting the idiosyncratic properties of relational data. In particular, we show that, unlike existing tensor factorizations, our proposed method is capable of exploiting contextual information that is more distant in the relational graph. Furthermore, we present an efficient algorithm for computing the factorization. We show that our method achieves better or on-par results on common benchmark data sets, when compared to current state-of-the-art relational learning methods, while being significantly faster to compute.

In the second part of the thesis, we focus on large-scale relational learning and its applications to Linked Data. By exploiting the inherent sparsity of relational data, an efficient computation of RESCAL can scale up to the size of large knowledge bases, consisting of millions of entities, hundreds of relations and billions of known facts. We show this analytically via a thorough analysis of the runtime and memory complexity of the algorithm as well as experimentally via the factorization of the YAGO2 core ontology and the prediction of relationships in this large knowledge base on a single desktop computer. Furthermore, we derive a new procedure to reduce the runtime complexity for regularized factorizations from $O(r^5)$ to $O(r^3)$ – where r denotes the number of latent components of the factorization – by exploiting special properties of the factorization. We also present an efficient method for including attributes of entities in the factorization through a novel coupled tensor-matrix factorization. Experimentally, we show that RESCAL allows us to approach several relational learning tasks that are important to Linked Data.

In the third part of this thesis, we focus on the theoretical analysis of learning with tensor factorizations. Although tensor factorizations have become increasingly popular for solving

machine learning tasks on various forms of structured data, there exist only very few theoretical results on the generalization abilities of these methods. Here, we present the first known generalization error bounds for tensor factorizations. To derive these bounds, we extend known bounds for matrix factorizations to the tensor case. Furthermore, we analyze how these bounds behave for learning on over- and understructured representations, for instance, when matrix factorizations are applied to tensor data. In the course of deriving generalization bounds, we also discuss the tensor product as a principled way to represent structured data in vector spaces for machine learning tasks. In addition, we evaluate our theoretical discussion with experiments on synthetic data, which support our analysis.

Zusammenfassung

Relationale Daten werden in bisher unbekanntem Ausmaß generiert und sind in vielen Anwendungsgebieten, wie zum Beispiel der Bioinformatik, der Künstlichen Intelligenz oder Sozialen Netzwerken allgegenwärtig, so dass das Relationale Lernen, welches das Maschinelle Lernen auf diesen Daten behandelt, sehr stark an Bedeutung gewonnen hat. Effiziente sowie skalierbare Methoden für das Relationale Lernen sind somit eine wichtige und aufgrund der besonderen Charakteristiken von Relationalen Daten herausfordernde Aufgabe. Diese Dissertation widmet sich der Anwendung von Tensor Faktorisierung auf das Relationale Lernen.

Im ersten Teil dieser Arbeit liegt der Fokus auf dem relationalen Aspekt des Maschinellem Lernens auf Relationalen Daten mittels Tensor Faktorisierung. Hierfür präsentieren wir das RESCAL Modell, eine neuartige Tensor Faktorisierung für das Relationale Lernen, welche der inhärenten Struktur von Multi-Relationalen Daten Rechnung trägt. Wir zeigen, dass diese Methode, im Gegensatz zu existierenden Tensor Faktorisierungen, dadurch in der Lage ist kontextuelle Information effizient zu nutzen auch wenn sich diese weiter entfernt ein einem Relationalen Graphen befindet. Des Weiteren präsentieren wir einen effizienten Algorithmus zur Berechnung der Faktorisierung. Wir zeigen auf Benchmark-Datensätzen, dass unsere Methode vergleichbare oder bessere Ergebnisse liefert als "state-of-the-art" Relationale Lernmethoden und gleichzeitig, wesentlich schneller zu berechnen ist.

Im zweiten Teil dieser Arbeit behandeln wir Relationales Lernen auf großen Datenmengen und Anwendungen auf Linked Data. Wir zeigen, dass eine effiziente Berechnung des RESCAL Modells, welche der Dünn-Besetztheit von Relationalen Daten Rechnung trägt, bis zu großen Wissensbasen skaliert, welche aus Millionen von Entitäten, Hunderten von Relationen und Milliarden von bekannten Fakten bestehen. Wir zeigen dies einerseits analytisch durch die Analyse der Laufzeit- und Speicherkomplexität des entwickelten Algorithmus. Wir zeigen dies andererseits auch experimentell durch die Faktorisierung der YAGO2 core Ontologie und durch die globale Vorhersage von Beziehungen auf dieser großen Wissensbasis. Des Weiteren präsentieren wir ein neues Verfahren zur Berechnung der Faktorisierung, welches die Laufzeitkomplexität von $O(r^5)$ auf $O(r^3)$ verringert – wobei r für die Anzahl der latenten Komponenten der Faktorisierung steht. Wir zeigen zudem experimentell, dass mit Hilfe unseres Ansatzes wichtige Probleme des Maschinellen Lernens auf Linked Data effizient gelöst werden können.

Im dritten Teil dieser Arbeit liegt der Fokus auf der theoretischen Analyse von Tensor Fakto-

risierungen. Obwohl Tensor Faktorisierungen stark an Beliebtheit gewonnen haben, um Aufgaben des Maschinellen Lernen auf strukturierten Daten zu lösen, existieren nur sehr wenige theoretische Resultate, welche Einsicht in die Generalisierungsfähigkeiten dieser Methoden bieten. Dieser Teil der Arbeit untersucht, wie das Tensor Produkt im Maschinellen Lernen als genereller Ansatz verwendet werden kann um strukturierte Daten in Vektorräumen zu repräsentieren. Um Schranken für den Generalisierungsfehler von Tensor Faktorisierungen abzuleiten, werden bekannte Schranken von Matrix Faktorisierungen erweitert. Des Weiteren wird analysiert, wie sich diese Schranken im Falle von über- oder unterstrukturierten Repräsentationen verhalten, zum Beispiel wenn Matrix Faktorisierung auf Tensor-Daten angewandt wird. Wir evaluieren zusätzlich unsere theoretischen Erwägungen durch Experimente auf synthetischen Daten, welche unsere Analyse unterstützen.

Chapter 1

Introduction

> I am convinced that the crux of the problem of learning is recognizing relationships and being able to use them
>
> *Christopher Strachey in a letter to Alan Turing, 1954*

> Classification of mathematical problems as linear and non-linear is like classification of the universe as bananas and non-bananas
>
> *Unknown Source*

The relationships between entities are a rich source of information, whose exploitation has been essential for a number of important scientific and technological advances in recent years. For instance, social networking services, which have revolutionized the way that people interact and communicate, are largely based on the relationships between persons whose analysis is of great interest to the social sciences and to commercial entities alike. Similarly, bioinformatics and molecular biology, which have contributed fundamental insights into the life sciences, make extensive use of the relationships between proteins, genes and chemical compounds and are considered enabling technologies for new approaches to health care such as translational medicine. Furthermore, the World Wide Web, which has enabled the access to information on an unprecedented scale and which can be considered as one of the most disruptive technologies in recent years, is primarily based on the linkage of related documents and information.

An important development with regard to knowledge representation is also the advent of the Semantic Web and the Linked cloud, which aim to create a web of semantically structured data. For the first time in history, these projects made large quantities of knowledge publicly

available in a relational and across different domains interlinked form. By the time of this writing, over 60 billion[1] known facts have been published in relational form in hundreds of interlinked databases in the Linked Data cloud (Cyganiak and Jentzsch, 2011; Auer et al., 2013) and its size is still growing steadily. While the Semantic Web as envisioned by Berners-Lee and Fischetti (2008, Chapter 12) has yet to be realized, efforts and projects surrounding the Semantic Web have already made tremendous impact. The abundance of interlinked semantic information has, for instance, enabled new ways to access biomedical knowledge in the life sciences (Ruttenberg et al., 2009) and is also expected to drive next generation information retrieval methods such as Google's Knowledge Graph.

It follows immediately from this great variety of use-cases and data sources that the ability to learn from relational data has a significant impact on many applications. Important tasks like the prioritization of unknown gene-disease associations, the prediction of links in social networks, or the answering of queries in incomplete knowledge bases can all benefit greatly from an efficient and reliable method to predict unknown relationships. Moreover, relational data, as all forms of data, can not only be incomplete but can be uncertain, noisy and include false information; a problem that is being aggravated as knowledge bases are being created increasingly via automatic information extraction methods. Seminal projects like NELL (Carlson et al., 2010), REVERB (Fader et al., 2011), or PATTY (Nakashole et al., 2012a; Nakashole et al., 2012b) aim to create large knowledge bases via the extraction of relational information from natural language; a very difficult task in which these projects achieve good but not perfect precision. To improve the quality of the extracted relational data, it is therefore important to identify objects that are likely to refer to identical entities or to determine which relationships are likely and unlikely to exist. *Relational learning* is a branch of machine learning that is concerned with all of these tasks, i.e. to learn efficiently from relational information for tasks like link prediction, entity resolution or collective classification. It has been shown numerous times that learning methods which model a relational domain truthfully and that take the relationships of entities into consideration, can improve learning results significantly over non-relational methods (Dzeroski, 2001; Singh and Gordon, 2008b; Taskar et al., 2004; Davis et al., 2005; Getoor and Taskar, 2007; Raedt, 2008). This underlines both the importance of relational data for knowledge representation and the importance of learning methods that can exploit this representation. However, despite the success of relational learning in specific applications, wider adoption has been hindered by multiple

[1] In this thesis, we adopt the *short-scale naming system* and use the term "billion" to refer to a thousand millions (10^9).

factors, such as the complexity of existing methods, their need for extensive prior knowledge, and their considerable scalability issues. Learning from relational data, and in particular learning from relational data on a large scale, is therefore one of the important tasks and one of the great challenges for today's machine learning.

The remainder of this chapter is structured as follows: Section 1.2 will provide a brief introduction into relational learning and existing relational learning methods. This section will underline again the importance of relational learning and discuss problems of existing relational models in more detail. Since this thesis will make use of many concepts related to tensors and tensor factorizations, we will provide a short introduction into these concepts in section 1.3. At last, in section 1.4 we will provide an overview of the contributions of this thesis.

1.1. Notation

In the following, scalars will be denoted by lowercase letters x; vectors will be denoted by bold lowercase letters $\boldsymbol{x}, \boldsymbol{y}$ with elements x_i, y_j. Vectors are assumed to be column vectors. Matrices will be denoted by uppercase letters X, Y with elements x_{ij}. Tensors will be indicated by upright bold uppercase letters $\mathbf{X}, \mathbf{Y}$ with elements $x_{i_1,\dots,i_n}$. For notational convenience, we will often group tensor indices into a vector $\boldsymbol{i} = [i_1, \cdots, i_n]^T$ and write $x_{\boldsymbol{i}}$ instead of $x_{i_1,\dots,i_n}$. Sets will be denoted by calligraphic letters $\mathcal{S}$ and their cardinality will be denoted by $|\mathcal{S}|$.

1.2. An Introduction to Relational Learning

Relational learning is largely characterized by the properties of relational data, the assumptions made about the data, and the learning tasks which are sought to be solved. In this short introduction, we will therefore begin with a definition of relational data in section 1.2.1 and discuss the consequences of its properties for machine learning. In section 1.2.2 we will review important tasks in relational learning tasks. In section 1.2.3 we will discuss influential approaches to relational learning and evaluate some of their properties that have hindered their wider adoption.

1.2.1. Learning in Relational Domains

In relational learning, various, mostly equivalent definitions exist on the nature of relational data, which are usually based on entries in relational databases or ground predicates in first-order logic (Friedman et al., 1999; Heckerman et al., 2007; Richardson and Domingos,

2006). In this thesis, we adopt a mathematical definition of relational data, based on set theory and n-tuples, which can be interpreted as underlying both, the relational database model and the first-order logic interpretation of relational data.

Generally speaking, relations describe connections that exist between entities, e.g. whether two persons are friends, whether a person likes a movie, or whether a university is located in a specific country. Mathematically, or more precisely, in set theory and logic, an n-ary relation is simply defined as a set of n-tuples. Let $\cdot \times \cdot$ denote the *Cartesian product* of sets. An n-ary relation $\mathcal{R}$ is then defined as a subset of the Cartesian product of n sets (Halmos, 1998, Chapter 7), which is formally expressed as:

$$\mathcal{R} \subseteq \mathcal{V}_1 \times \cdots \times \mathcal{V}_n,$$

In turn, the Cartesian product of n sets $\mathcal{V}_1, \ldots, \mathcal{V}_n$ is defined as the set of all possible n-tuples over $\mathcal{V}_1, \ldots, \mathcal{V}_n$ (Halmos, 1998, Chapter 6), i.e. as the set

$$\mathcal{V}_1 \times \cdots \times \mathcal{V}_n := \{(v_1, \ldots, v_n) \mid v_1 \in \mathcal{V}_1 \wedge \ldots \wedge v_n \in \mathcal{V}_n\},$$

such that a relation $\mathcal{R}$ can be interpreted as being the set of all *existing* relationships, while the Cartesian product can be interpreted as being the set of all *possible* relationships over the entities in the domains $\mathcal{V}_1, \ldots, \mathcal{V}_n$. In the following, we will also use $\text{dom}(\mathcal{R})$ to denote the Cartesian product $\mathcal{V}_1 \times \cdots \times \mathcal{V}_n$ associated with a relation $\mathcal{R} \subseteq \mathcal{V}_1 \times \cdots \times \mathcal{V}_n$. Furthermore, we will refer to a single n-tuple $(v_1, \ldots, v_n) \in \text{dom}(\mathcal{R})$ as a possible *relationship* between the entities $v_1, \ldots, v_n$.

This tuple-based definition of relational data serves as a basis for the relational model in database theory (Codd, 2001) and has also a close connection to first-order logic: For a set $\mathcal{X}$ and a subset $\mathcal{Y} \subseteq \mathcal{X}$, the *characteristic function* of $\mathcal{Y}$ is a boolean-valued function $\phi_{\mathcal{Y}} : \mathcal{X} \mapsto \{0, 1\}$ which indicates for all elements in $\mathcal{X}$, whether they are also an element of the subset $\mathcal{Y}$ (Halmos, 1998), i.e.

$$\forall x \in \mathcal{X} : \phi_{\mathcal{Y}}(x) := \begin{cases} 1 & \text{if } x \in \mathcal{Y} \\ 0 & \text{otherwise} \end{cases}$$

Hence, for a relation $\mathcal{R} \subseteq \mathcal{V}_1 \times \ldots \times \mathcal{V}_n$, its characteristic function is a function

$$\phi_{\mathcal{R}} : \mathcal{V}_1 \times \ldots \times \mathcal{V}_n \mapsto \{0, 1\}.$$

which is true if and only if a particular relationship exists. Along the lines of Fregean

logic, a characteristic function of a relation is also referred to as a *predicate*, which are the building blocks of predicate logic. For notational convenience, we will usually denote the existence of a particular relation ship by $\mathcal{R}(v_1, \ldots, v_n)$ instead of the more cumbersome terms $(v_1, \ldots, v_n) \in \mathcal{R}$ or $\phi_{\mathcal{R}}(v_1, \ldots, v_n) = 1$.

This thesis will mostly focus on *dyadic relational data*, which is an important subclass of relational data where all relations are binary, i.e. where $\mathcal{R}_k \subseteq \mathcal{V}_i \times \mathcal{V}_j$ for all relations $\mathcal{R}_k$. Modeling data in form of dyadic relations has proven to be very versatile. It is, for instance, used in the Semantic Web's Resource Description Framework (RDF) to represent knowledge on the scale of the World Wide Web and is a common way to store information in knowledge bases. N-ary relations, which involve more than two sets of entities, can be converted to dyadic relations by introducing auxiliary entities, such as blank nodes in RDF (Antoniou and Harmelen, 2004, Section 3.2.7). *In the remainder of this thesis, we will assume that relational data is in dyadic form, unless clearly noted otherwise.* For a relationship $\mathcal{R}_k(a, b)$ we will, similar to RDF, refer to a as the *subject* and to b as the *object* of the relationship. Furthermore, we will assume that relational data is in the following normalized form: Let $\mathcal{E}_m$ denote a set of entities of a particular type, e.g. a set of persons, countries or Ph.D. students and let $\mathcal{A}_n$ denote the possible values for the n-th attribute of an entity, e.g. the age of a person or the national product of a country. Then, any relation $\mathcal{R}$ is either a subset of $\mathcal{E}_i \times \mathcal{E}_j$, which we will also refer to as an entity relation or it is a subset of $\mathcal{E}_i \times \mathcal{A}_j$ which we will also refer to as an attribute relation.[1] This corresponds to the Semantic Web's distinction between object- and datatype properties in its Web Ontology Language (OWL). In the following, we will restrict the term relationship to refer to tuples from entity relations, while tuples from attribute relations will be called attribute values.

Learning in relational domains is in many ways connected to learning the characteristic function of relations, to predict, for instance, which relationships are true in a given domain. One of the key insights and motivations for relational learning is that the relationships of entities introduce rich patterns in a data set which can be exploited to improve the learning and prediction process significantly. In a non-relational machine learning setting, data is usually assumed to range over a single type of entities and to be attribute-valued, i.e. to consist only of relations of the form $\mathcal{E} \times \mathcal{A}_j$, where $\mathcal{E}$ denotes the set of all entities and the sets $\mathcal{A}_j$

[1] The distinction between entities and attribute values is largely domain-specific and difficult to generalize. For instance, in one domain the number 42 can occur as an attribute value as the age of a person, while it can occur in a different domain as an object in a relation like $23 < 42$. However, for all practical purposes considered in this thesis, it is usually clear what constitutes an entity and what constitutes an attribute value and we will assume that the distinction is given.

correspond to the different attributes of these entities. For instance, $\mathcal{E}$ could consist of all Ph.D. students in a university and the sets $\mathcal{A}_j$ could reflect attributes like gender, age, intelligence etc. An important assumption that is typically made in non-relational machine learning is that attribute values of *different* entities are independent. For instance, the age at which a student $\mathtt{a} \in \mathcal{E}$ pursues his Ph.D. studies might depend on other attributes of this particular student, like his intelligence, but it is assumed to be independent from the attributes of a second student $\mathtt{b} \in \mathcal{E}$, when $\mathtt{a} \neq \mathtt{b}$. The setting considered in relational learning however, is different: Relational data can not only consist of multiple types of entities, but more importantly, it also includes relationships between these entities, i.e. relations of the form $\mathcal{E}_i \times \mathcal{E}_j$. For instance, in addition to a set of students $\mathcal{E}_i$ and their attributes, relational data could consist of a set of professors $\mathcal{E}_j$, a relation $\mathtt{advisedBy} \subseteq \mathcal{E}_i \times \mathcal{E}_j$ that indicates which student is advised by which professor, and a relation $\mathtt{friendOf} \subseteq \mathcal{E}_i \times \mathcal{E}_j$ that indicates which students are friends. It has been shown that this kind of relational information introduces patterns from which dependencies can be derived across entity boundaries. Important examples of such patterns include:

Homophily It is well-known from the analysis of social networks that a commonly occurring pattern in these networks is *homophily*, i.e. the tendency of persons to be associated with persons that share similar characteristics. In relational learning, this type of pattern is also called *autocorrelation* and has shown to be present in many relational data sets (Jensen and Neville, 2002). Homophily in relational data can be exploited to predict unknown relationships as well as unknown attributes. For instance, a good covariate to predict the age of a student might be the age of his friends in a social network.

Stochastic Equivalence Another pattern which is often present in relational data is *stochastic equivalence*. It refers to the fact that entities in a data set can be partitioned into groups such that the observed relationships can be explained via relationships between these groups (Hoff, 2008). Stochastic equivalence can be exploited for the analysis of relational data, e.g. to cluster entities according to their relationships or to predict unknown relationships when the cluster memberships of entities are known.

Global Dependencies An important and very general form of patterns commonly occurring in relational data are *global dependencies* between relationships, i.e. dependencies that affect different types of relations and that can possibly range over chains of multiple relationships. For instance, whether a student completes a course successfully might

depend on the teaching ability of the professor who teaches the particular course. Such a pattern ranges over multiple connected relationships of different types, what becomes evident when the pattern is expressed in a logical formalism

$$\texttt{holdsCourse}(p,c) \wedge \texttt{teachingAbility}(p,\texttt{high}) \Rightarrow \texttt{completedCourse}(s,c), \quad (1.1)$$

where s ranges over all students, p ranges over all professors, and c ranges over all courses. Similar to homophily, global dependencies can be exploited to predict relationships and attributes of entities from the properties of related entities. However, in the case of global dependencies the related entities do not have to be directly connected to an entity.

The presence of these patterns in relational data illustrates the benefits of combining a relational representation of knowledge with learning methods that can fully exploit this representation: It allows for the efficient exploitation of patterns and dependencies which would otherwise not be accessible and can aid the learning process tremendously. However, all of these dependencies can occur between attributes and relationships of different entities. It is therefore necessary to remove the independence assumptions of traditional machine learning, to be able to exploit these relational patterns. This is one of the main differences between relational and non-relational machine learning. If it is necessary to clearly distinguish between methods that truly exploit relational information from methods that learn *on* relational data but do not fully exploit the relationships between entities, we will also use the term *collective learning* to refer to the former learning methods.

The objective of statistical relational learning (SRL) is to derive a complete model of a relational domain from uncertain data, i.e. relational data that can be incomplete, noisy, and contain false information. It is also assumed that dependencies in the data are rather statistical in nature than deterministic, e.g. that patterns such as equation (1.1) hold with varying degrees of probability rather then being either true or false. Statistical relational learning is the general setting considered in this thesis. In SRL, data is usually modelled in the following way: For each relationship $\mathcal{R}_k \subseteq \mathcal{V}_a \times \mathcal{V}_b$ and each possible relationship $\mathcal{R}_k(v_i, v_j)$, a binary random variable X_{ijk} is created which represents the existence of the associated relationship, meaning that it is set to the value of the characteristic function $\phi_{\mathcal{R}_k}(v_i, v_j)$. Let $\mathcal{X}$ denote the set of all these random variables. To derive a complete model of the relational domain, we are then interested in estimating the joint distribution $\mathrm{P}(\mathcal{X})$. A desirable property of relational models is that they do not only model the joint probability well, but also that

they represent and answer queries on $\mathrm{P}(\mathcal{X})$ efficiently. The additional complexity that is introduced through a relational model becomes apparent when examining the number of variable states that have to be considered to specify the joint distribution in this setting. For instance, consider a very simple data set with n entities and m binary attributes and let A_{ij} denote the j-th attribute of the i-th entity. The probability table of the joint distribution of k binary variables consist of 2^k entries, namely of one entry for each possible assignment of values to these random variables. In a non-relational learning setting, the joint distribution over the variables A_{ij} factors nicely due to the independence assumption of non-relational learning, such that

$$\mathrm{P}(A_{11},\ldots,A_{nm}) = \prod_{i=1}^{n} \mathrm{P}(A_{i1},\ldots,A_{im}) \ .$$

It would therefore only be necessary to specify n times the joint distribution of m binary variables in a non-relational setting, meaning that n times 2^m different states have to be considered. However, relational learning removes exactly this independence assumption, such that it would be necessary to consider the joint distribution of all nm binary variables, what leads to 2^{nm} possible states. And this is only a very simple case where data consists of a few binary variables. For instance, for dyadic relational data consisting of n entities and m entity relations, the number of possible states would already rise up to $2^{n^2 m}$, what becomes intractable very quickly for any reasonable amount of entities. This illustrates the hardness and complexity of relational learning. An important objective of relational learning methods is therefore to reduce this complexity to a manageable size. The strategies employed to achieve this task will be discussed in section 1.2.3.

1.2.2. Relational Learning Tasks

Relational learning is not only concerned with learning efficiently from relational data, but also focuses on special tasks that arise with this kind of data. In the following, we will briefly outline the most common and important of these tasks and in doing so illustrate again the benefits of a relational approach.

Link Prediction *Link prediction* is central to relational learning. Its objective is to determine whether a particular relationship exists, i.e. whether $\phi_{\mathcal{R}}(t_k) = 1$ for a particular relation $\mathcal{R}$ and any valid n-tuple $t_k \in \mathrm{dom}(\mathcal{R})$. Typical applications of link prediction are to predict friendships in social networks, protein-protein interactions in bioinformatics, or general relationships in a knowledge base. It has been shown that non-relational methods, i.e. methods

that predict the existence of a link solely on the attributes of the involved entities, can be significantly outperformed by a relational approach on this task (Taskar et al., 2004; Getoor and Diehl, 2005). Link prediction is not only central to relational learning because relationships form the basis of relational data, but also because many relational learning tasks can be cast as a link prediction problem as it will be shown in the following paragraphs.

Collective Classification *Collective classification* is the extension of classification to relational learning. In a standard classification setting, the classes of entities are inferred from their attributes. The underlying idea of collective classification is that relational data provides valuable information for classification, as related entities often share identical classes. Similar to link prediction, it has been shown that collective classification can improve learning results significantly over its non-relational counterpart (Sen et al., 2008; Jensen et al., 2004; Neville and Jensen, 2003; Taskar et al., 2002). Collective classification can be cast as a link prediction problem, by introducing classes as entities to the data, introducing a relation `isClassOf` and inferring the probability of relationships `isClassOf`(i`-th entity`,j`-th class`).

Entity Resolution *Entity resolution*, which is also known as object identification, record linkage, instance matching, and deduplication amongst others, is the problem of identifying which objects in the data refer to the identical underlying entity. Entity resolution is an important task in many fields of application such as database deduplication, linked data or natural language processing and can benefit significantly from relational information. In a relational setting, the decisions about which objects are assumed to be identical can propagate via the relationships between these objects, such that the matching decisions are performed collectively for all objects rather then independently for each object pair. It has also been shown for this task that relational learning can improve learning results significantly over a non-relational approach (Singla and Domingos, 2006a; Bhattacharya and Getoor, 2007; Whang and Garcia-Molina, 2012). Entity resolution can be cast as a link prediction problem by introducing a new relation `isEqual` and inferring the probability of relationships `isEqual`(i`-th entity`,j`-th entity`).

Link-Based Clustering *Link-based clustering* is the extension of clustering to a relational learning setting. Similar to feature-based clustering, entities are partitioned into groups based on their similarity. However, in link-based clustering, entities are not only grouped by the similarity of their attributes but also by similarity of their relationships. As in entity resolution,

this similarity of entities can propagate in a relational setting through relationships, such that a relational modeling can add important information for this task. In social network analysis, link-based clustering is also referred to as community detection (Fortunato, 2010).

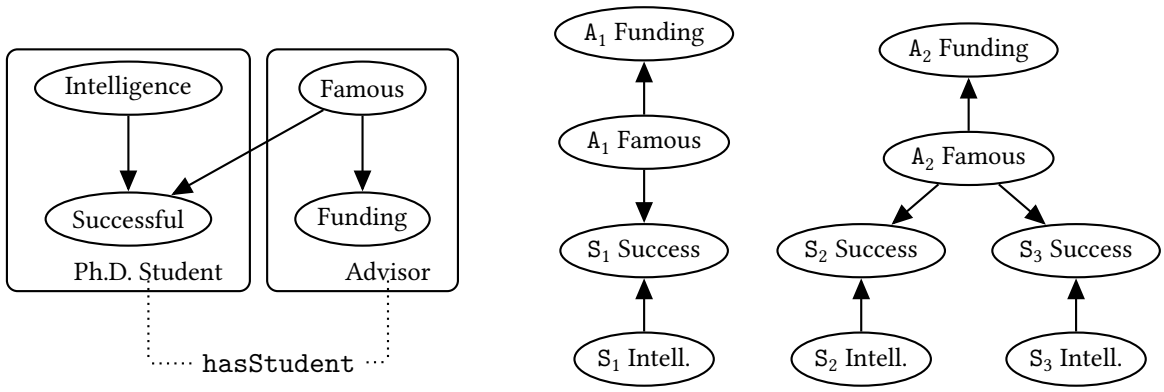
1.2.3. Relational Models

Relational learning regained the attention of the machine learning community with the work of Koller and Pfeffer (1998) and Friedman et al. (1999) on SRL, although its origins can be traced back even to Winston (1975). Since then, a variety of models has been proposed for relational learning, which also cover different learning paradigms. In this section we will briefly review the most important and influential SRL models, as well as the concepts on which these models are based.

Relational Graphical Models

The vast majority of *statistical* relational learning methods are based on probabilistic graphical models (PGMs); a formalism which uses graph theory to encode the joint distribution of multiple random variables. Roughly speaking, random variables are represented as nodes in a graph and the edge configuration of this graph encodes the statistical dependencies among the variables of a PGM. The most common classes of graphical models are Bayesian networks, which are defined over directed graphs and Markov networks, which are defined over undirected graphs. The attractiveness of graphical models for relational learning stems from their ability to efficiently model high-dimensional probability distributions: Usually only a small number of the dependencies between all possible relationships are reasonable to be considered. Being able to confine the considered dependencies to this set of reasonable dependencies would therefore reduce the complexity of learning significantly and render it tractable in a relational setting. PGMs permit relational learning methods to achieve this task by enabling a form of template mechanism which allows to specify these dependencies intuitively. In the following we will review two important relational learning methods based on Bayesian and Markov networks that employ such a template mechanism. For the sake of simplicity, we will make no distinction between a node and its associated random variable in a PGM.

Probabilistic Relational Models Probabilistic Relational Models (PRMs) are based on Bayesian networks and were one of the earliest approaches to statistical relational learning (Koller and Pfeffer, 1998; Friedman et al., 1999; Getoor et al., 2007). The underlying idea of



(a) PRM dependency structure in a simple university domain. The "cross-class" dependency between *Famous* and *Successful* is permitted because the classes *Advisor* and *Ph.D. Student* are connected via the `hasStudent` relation, as indicated by the dotted line in the diagram.

(b) Grounded PRM for the dependency structure shown in figure 1.1a, where Advisor = $\{\mathtt{A}_1, \mathtt{A}_2\}$, Ph.D. Student = $\{\mathtt{S}_1, \mathtt{S}_2, \mathtt{S}_3\}$, and `hasStudent` = $\{(\mathtt{A}_1, \mathtt{S}_1), (\mathtt{A}_2, \mathtt{S}_2), (\mathtt{A}_2, \mathtt{S}_3)\}$.

Figure 1.1.: Example for dependency structure of Probabilistic Relational Models with associated grounded Bayesian network. This example has been adapted from Pasula and Russell (2001).

PRMs is to use an object-oriented representation of a database as a template for the dependency structure of a Bayesian network. PRMs start from a relational schema that describes that classes in a domain, their attributes, and the relations between classes. Furthermore the set of entities that exist in the domain is given. In its most basic form, PRMs also assume that the relationships between entities are known and that only the attribute values are uncertain. Then, for each class C and each attribute $\mathcal{A}_p$ of C, a PRM consists of

1. a specification of the parents $\text{par}(\mathcal{A}_p)$ of $\mathcal{A}_p$. The elements of $\text{par}(\mathcal{A}_p)$ are allowed to be either attributes of C, or to be attributes of a different class D, with the constraint that D is directly or indirectly related to C via a chain of relations.

2. a conditional probability distribution $\text{P}\left(\mathcal{A}_p \,\middle|\, \text{par}(\mathcal{A}_p)\right)$

Given a set of entities, a PRM is then "compiled" into a ground Bayesian network $(\mathcal{G}, \mathcal{P})$ in the following way: Let A_{ip} be the random variable associated with the p-th attribute of the i-th entity. Then, the Bayesian network graph $\mathcal{G}$ contains a node A_{ip} for each attribute of each entity. Furthermore, it contains an edge from A_{ip} to A_{jq} if $\mathcal{A}_p \in \text{par}(\mathcal{A}_q)$ and if one of the following conditions is satisfied:

1. the nodes A_{ip}, A_{jq} correspond to attributes of the same entity, i.e. $i = j$

2. entity i and entity j are directly or indirectly related via a chain of relations

Figure 1.1 shows an example of the dependency structure of a PRM and its associated grounded Bayes net. The conditional probability distribution (CPD) associated with each node A_{ij} is the CPD $\mathrm{P}\left(\mathcal{A}_j \,\middle|\, \mathrm{par}(\mathcal{A}_j)\right)$, such that all entities of one class share the same CPD. It follows from item 2 that this basic form of PRMs requires knowledge about the relationships between entities. Subsequently, PRMs have also been extended to handle uncertain relationships, which is referred to as *structural uncertainty* within the PRM nomenclature. When the dependency structure is known, learning a PRM consists of estimating the parameters of the CPDs what is usually referred to as *parameter learning*. However, in most cases the dependency structure will not be known and therefore has to be inferred from data, what is referred to as *structure learning*.

Markov Logic Networks Markov Logic Networks (MLNs) are a combination of first-order logic and Markov networks to create a form of probabilistic logic (Richardson and Domingos, 2006). Similarly to PRMs, Markov Logic Networks can be regarded a template based approach to create the dependency structure of a graphical model. The template mechanism in MLNs however is based on first-order logic, such that a set of logical formulas is used to create a grounded Markov network over a particular domain. Formally, a MLN L is defined as a set of weighted first-order logic formulas $L = (\mathcal{K}, \boldsymbol{w})$, where $w_i \in \mathbb{R}$ denotes the weight of the i-th formula $F_i \in \mathcal{K}$. The set of logical formulas $\mathcal{K}$ is also referred to as a *knowledge base* which incorporates prior knowledge about a domain. Given a finite set of entities $\mathcal{E}$, a Markov network $M_{L,\mathcal{E}}$ is created from L in the following way: For each relation $\mathcal{R}_k$ and each possible relationship $t_i \in \mathrm{dom}(\mathcal{R}_k)$ a node X_i is created in $M_{L,\mathcal{E}}$. The value of a node is set to the value of the characteristic function of the corresponding relationship, i.e. $X_i = \phi_{\mathcal{R}}(t_i)$. This construction corresponds to the setting considered in section 1.2.1. Furthermore, for each possible grounding of a formula F_i – what corresponds to a set of nodes in $M_{L,\mathcal{E}}$ – a feature function f_k is defined, which is set to 1 if the grounded formula is true and 0 otherwise. In a graphical representation of $M_{L,\mathcal{E}}$ this is equivalent to creating an edge between two nodes if the corresponding relationships occur in at least one formula F_i. An example of this construction is shown in figure 1.2. All groundings f_k of a formula F_i share the same weight

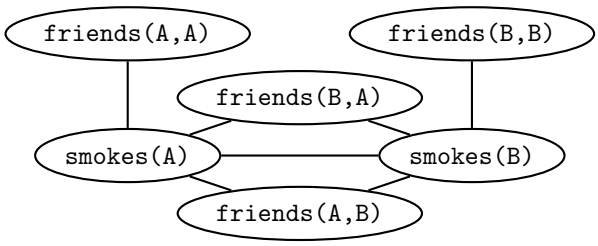


Figure 1.2.: Example of a grounded Markov Logic Network, by applying the formula $\texttt{friends(x,y)} \Rightarrow \texttt{smokes(x)} \Leftrightarrow \texttt{smokes(y)}$ to the entities $\{\texttt{A},\texttt{B}\}$. This example has been adapted from Richardson and Domingos (2006).

w_i, such that the probability distribution of $M_{L,\urcorner}$ can be written as

$$\mathrm{P}\left(X_1 = x_1, \ldots X_n = x_n\right) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(\{x_j\}_{j=1}^n)\right) \tag{1.2}$$

where $\{x_j\}_{j=1}^n$ denotes the state of all random variables and $n_i(\{x_j\}_{j=1}^n)$ denotes the number of true groundings of F_i in $\{x_j\}_{j=1}^n$. Just as for PRMs, learning MLNs involves parameter and structure learning. When the knowledge base $\mathcal{K}$ is known, parameter learning corresponds to learning the optimal weights w_i for each formula F_i. When the knowledge base $\mathcal{K}$ is unknown, structure learning has to be employed. In the context of MLNs this is equivalent to learning the logical formulas in $\mathcal{K}$. Structure learning can be accomplished by inductive logic programming (ILP), or by more integrated approaches that evaluate the gain of adding a formula to $\mathcal{K}$ via its effect on equation (1.2) (Richardson and Domingos, 2006; Kok and Domingos, 2005).

Further Relational Graphical Models In addition to PRM and MLN, a large number of further relational learning methods based on graphical models have been considered. Important models based on this paradigm include Relational Dependency Networks (Neville and Jensen, 2007), Relational Markov Networks (Taskar et al., 2002), DAPER (Heckerman et al., 2004; Heckerman et al., 2007) and Bayesian Logic Programs (Kersting and De Raedt, 2001; Kersting and De Raedt, 2007).

Discussion Exact inference in PGMs, is known to be at least $\mathcal{NP}$-hard (Koller et al., 2007). For this reason, a variety of approximate inference methods has been considered, ranging from variational inference and loopy belief propagation to Markov Chain Monte Carlo (MCMC) methods such as Gibbs Sampling (Jordan, 1998; Murphy et al., 1999; Wainwright and Jordan, 2007). While these approximate methods made inference in graphical models tractable and

enabled their success in many fields of machine learning, they still remain highly expensive in terms of computational complexity; especially in a relational setting with a large number of variables and a large number of statistical dependencies. Moreover, algorithms to compute these inference methods can be complex and difficult to implement. A second point of consideration is that nearly all methods require extensive prior knowledge about the learning task at hand. If the required prior knowledge is unknown, it has to be inferred automatically from data, which is often a time-consuming and error prone task. For instance, PRMs require the dependency structure from which the grounded Bayesian network is created. Unfortunately, finding the *optimal* structure of a PRM is intractable, since it is at least as hard as finding the optimal structure of a Bayesian network, what is known to be $\mathcal{NP}$-complete (Chickering, 1996; Getoor et al., 2007). The standard approach to overcome this problem is to employ some form of heuristic or greedy search strategy to find a good solution. However, even these approximate methods involve expensive computations such as database joins and aggregation, in addition to the heuristic scoring function. Similarly, Markov Logic Networks require a set of first-order logic formulas to create the grounded Markov network. Unfortunately, learning logical formulas from data is a very difficult problem, such that state-of-the-art structure learning methods for MLNs still require multiple hours of runtime on fairly small datasets (Kok and Domingos, 2009; Davis and Domingos, 2010; Van Haaren and Davis, 2012). Structure learning is in addition to these scalability problems also a considerable source of error in the learning process – again due to the difficulty of the problem – which can deteriorate results significantly when erroneous structure is inferred.

Latent Variable Models

All relational learning methods considered so far model the statistical dependencies in the data solely using variables that have been observed in the data. *Latent variable models* for relational learning take a fundamentally different approach: Entities are modeled via *latent* variables, i.e. variables that have *not* been observed in the data but which are assumed to be hidden causes for the observable variables. The probability of a particular relationship between entities is then derived from a simple operation on these latent variables. In SRL, expressing data in terms of newly invented latent variables is also referred to as *predicate invention* and considered a powerful asset for relational learning (Kok and Domingos, 2007). *Latent class models* for relational learning are latent variable models in which each entity is assigned to exactly one out of multiple latent classes, i.e. where the latent variables are binary and mutually exclusive. The Infinite Hidden Relational Model (IHRM) and the Infinite

Relational Model (IRM) are *infinite* latent class models for relational learning and have been introduced independently by Xu et al. (2006) and Kemp et al. (2006). The underlying idea of both models is to assign entities to classes and to derive the probability of a relationship from the probability that there exists a relationship between members of the respective classes. This corresponds to the concept of stochastic equivalence in relational data as discussed in section 1.2.1. In its most simple manifestation, i.e. data consisting only of dyadic relations and without attributes, the model for IHRM and IRM is the following: Let $c_k \in \mathbb{N}$ denote the class of entity e_k. Then, the relationships between entities are created according to the generative process

$$\boldsymbol{c} \mid \gamma \quad \sim CRP(\gamma) \tag{1.3}$$

$$\eta(\mathcal{S}, a, b) \mid \beta \quad \sim Beta(\beta, \beta) \tag{1.4}$$

$$\mathcal{R}(e_i, e_j) \mid \boldsymbol{c}, \eta \sim Bernoulli(\eta(\mathcal{R}, c_i, c_j)), \tag{1.5}$$

where equation (1.3) assigns each entity to a class, equation (1.4) generates the probability of a relationship in relation $\mathcal{S}$ between members of classes a and b, and equation (1.5) generates the relationships between entities from these probabilities. Since the class assignments $\boldsymbol{c}$ are drawn from a Chinese restaurant process (CRP), the number of classes is potentially infinite and automatically inferred from data. When learning an IHRM or IRM, the task is then to infer the class assignments c_k as well as the class-relationship probabilities $\eta(\mathcal{S}, a, b)$. A particular advantage of IHRM over the IRM model is that attributes of entities can be modeled naturally using any appropriate probability distribution, while IRMs have to use a binary representation. The mixed-membership stochastic block model is as a generalization of IHRM where entities are allowed to be members of multiple classes, i.e. where the latent variables are *not* mutually exclusive (Airoldi et al., 2008).

Discussion An important advantage of latent variable models compared to previously discussed methods is that no structure learning is necessary for their functioning. The dependency structure is already defined in the model itself, i.e. given the latent variables all observable variables are conditionally independent. However, similar to relational learning methods based on PGMs, scalability is a significant concern for IHRM and IRM. Due to their Bayesian non-parametric nature they also require expensive inference methods such as MCMC. Even with sophisticated inference methods as developed by Xu et al. (2007), inference in IHRM remains impractical for large-scale data.

1.3. An Introduction to Tensors and Tensor Factorizations

This thesis is concerned with learning from relational information via tensor factorization. While tensor factorizations have long been used in psycho- and chemometrics to analyze multi-way data (Smilde et al., 2005; Kroonenberg, 2008), they have only recently been applied to machine learning and data mining. In the context of relational learning, tensor models are appealing because of their balance between the expressiveness and the complexity of their model. Tensors methods allow not only to model data with multiple modalities – such as entities and relations in relational data – but tensor factorizations can also be related to multilinear models, which overcome the limited expressiveness of linear models and at the same time remain more scalable and easier to handle than non-linear approaches. In this section, we will briefly review definitions and properties of tensors and tensor factorizations as far as they are relevant for the course of this thesis. The review will follow closely the discussion in Burdick (1995) and Kolda and Bader (2009), which also provide a more extensive introduction into this subject matter.

1.3.1. Tensors and the Tensor Product

Tensors, as generalizations of vectors and matrices, are defined via the tensor product of vector spaces. For this reason, we will first introduce the tensor product of vectors and vector spaces before defining the concept of a tensor.

Definition 1 (Tensor Product of Vectors, Burdick, 1995)**.** *Let $\boldsymbol{x} \in V$ and $\boldsymbol{y} \in W$, where $V \subseteq \mathbb{R}^n$, $W \subseteq \mathbb{R}^m$ are vector spaces. The tensor product of $\boldsymbol{x}$ and $\boldsymbol{y}$, in the following denoted by $\boldsymbol{x} \otimes \boldsymbol{y}$, is an array with mn entries, where*

$$(\boldsymbol{x} \otimes \boldsymbol{y})_{ij} = x_i y_j$$

The defining property of the tensor product of vectors is that $(\boldsymbol{x} \otimes \boldsymbol{y})_{ij} = x_i y_j$. However, since the "shape" of the array $\boldsymbol{x} \otimes \boldsymbol{y}$ is not defined, there exists a deliberate ambiguity in how to compute the tensor product of vectors. In particular, for two vectors $\boldsymbol{x}$, $\boldsymbol{y}$ we can obtain one- or two-dimensional arrays with

$$\boldsymbol{x} \otimes \boldsymbol{y} = \begin{bmatrix} x_1 \boldsymbol{y}^T & x_2 \boldsymbol{y}^T & \dots & x_n \boldsymbol{y}^T \end{bmatrix}^T \in \mathbb{R}^{mn} \tag{1.6}$$

$$\boldsymbol{x} \otimes \boldsymbol{y} = \boldsymbol{x}\boldsymbol{y}^T \in \mathbb{R}^{m \times n} \tag{1.7}$$

We will refer to equation (1.6) as a vectorized representation of the tensor product, as its result is again a vector, while equation (1.7) will be called a structured representation. Usually,

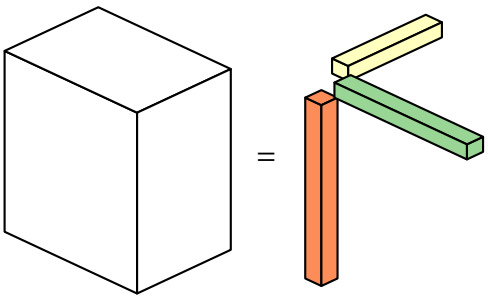


Figure 1.3.: Illustration of the tensor product of three vectors $\boldsymbol{a}$ (orange), $\boldsymbol{b}$ (turquoise), $\boldsymbol{c}$ (yellow). The resulting structured representation is a three-dimensional array.

it will be clear from context which representation is used. The tensor product of vectors is easily extended to more than two vectors, e.g. $(\boldsymbol{x} \otimes \boldsymbol{y} \otimes \boldsymbol{z})_{ijk} = x_i y_j z_k$. In the following, we will denote the tensor product of n vectors also by $\bigotimes_n \boldsymbol{v}_n$. In the structured representation, the tensor product of n vectors corresponds to an n-dimensional array. Furthermore, the tensor product of vectors preserves their linear independence: if the vectors $\mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and $\mathcal{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m\}$ are, respectively, linearly independent, then the vectors $\{\boldsymbol{x}_i \otimes \boldsymbol{y}_j \mid \boldsymbol{x}_i \in \mathcal{X} \wedge \boldsymbol{y}_i \in \mathcal{Y}\}$ are also linearly independent.

Definition 2 (Tensor Product of Vector Spaces, Burdick, 1995)**.** *The tensor product of the vector spaces V and W, in the following denoted $V \otimes W$, is the vector space consisting of all linear combinations $\sum_i a_i \boldsymbol{v}_i \otimes \boldsymbol{w}_i$, where $\boldsymbol{v}_i \in V$ and $\boldsymbol{w}_i \in W$.*

Similarly to the tensor product of vectors, the tensor product of vector spaces is easily extended to more than two vector spaces and $\bigotimes_n V_n$ will denote the tensor product of n different vector spaces. We will refer to a vector space that is the result of tensor products of vector spaces also as *tensor product space*.

Now, the central concept of a tensor can be defined formally as:

Definition 3 (Tensor, Kolda and Bader, 2009)**.** *Let $V = \bigotimes_n W_n$ be a tensor product space with $n \geq 1$. Then, the elements $\mathbf{X} \in V$ are called n-th order tensors.*

Following definition 1 and definition 3, tensors can be interpreted in different ways. One way is as a *vector in a structured vector space*, what corresponds to the vectorized representation in equation (1.6). However, since there exists the equivalent structured representation in equation (1.7), tensors can also be regarded as *multidimensional arrays*, which is the more commonly used interpretation. Here, we will use both interpretations interchangeably. It also

follows immediately, that vectors are first-order tensors and matrices are second-order tensors. In the following, ord($\mathbf{X}$) will denote the order of tensor $\mathbf{X}$. For notational convenience, we will write $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_k}$ also as $\mathbf{X} \in \mathbb{R}^{\prod_i n_i}$.

The tensor product of matrices, i.e. second-order tensors, is also known as the *Kronecker product*,[1] which is defined accordingly as:

Definition 4 (Kronecker Product)**.** *For two matrices $X \in \mathbb{R}^{n \times m}$ and $Y \in \mathbb{R}^{p \times q}$, their Kronecker product $X \otimes Y \in \mathbb{R}^{np \times mq}$ is computed as*

$$X \otimes Y = \begin{bmatrix} x_{11}Y & \dots & x_{1m}Y \\ \vdots & \ddots & \vdots \\ x_{n1}Y & \dots & x_{nm}Y \end{bmatrix} \tag{1.8}$$

Important properties of the Kronecker product which will be used throughout this thesis are (Laub, 2004, Theorem 13.3, 13.4, 13.6)

$$(A \otimes B)^T = A^T \otimes B^T \tag{1.9}$$

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1} \tag{1.10}$$

$$(A \otimes B)(C \otimes D) = AC \otimes BD \tag{1.11}$$

Related to the Kronecker product is the vectorization of matrices, which is defined as follows

Definition 5 (Vectorization, Golub and Loan, 2013, Section 1.3.7)**.** *The vectorization of a matrix $M \in \mathbb{R}^{m \times n}$, in the following denoted by* $\operatorname{vec}(M)$*, is a linear transformation that rearranges the columns of M into a column vector of length mn:*

$$\operatorname{vec}(M) = \begin{bmatrix} \boldsymbol{m}_{:,1} \\ \vdots \\ \boldsymbol{m}_{:,n} \end{bmatrix}$$

We also define the inverse vectorization operator, in the following denoted by $\operatorname{vec}_r^{-1}(\boldsymbol{v})$*, which rearranges a vector $\boldsymbol{v} \in \mathbb{R}^n$ into a matrix $M \in \mathbb{R}^{r \times \frac{n}{r}}$, such that $M = \operatorname{vec}_r^{-1}(\operatorname{vec}(M))$.*

An important property of the vectorization operation in conjunction with the Kronecker product is that the following equivalence holds for matrix product of three matrices A, X, B (Golub and Loan, 2013, Section 1.3.7):

$$Y = AXB \Leftrightarrow \operatorname{vec}(Y) = (B^T \otimes A) \operatorname{vec}(X) \tag{1.12}$$

[1] Outside of tensor methods, the Kronecker product has also received significant recognition in the machine learning and social network community lately, e.g. Leskovec et al., 2010

It will prove convenient in the remainder of this thesis to define various forms of subarrays on tensors: *Fibers* are one-dimensional subarrays of tensors, i.e. vectors, which are created by keeping all but one index fixed and varying the remaining index over all entries of the corresponding mode. The fibers of a third-order tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ will be denoted by $\boldsymbol{x}_{:,j,k}$, $\boldsymbol{x}_{i,:,k}$, and $\boldsymbol{x}_{i,j,:}$ with entries:

$$\boldsymbol{x}_{:,j,k} := \begin{bmatrix} x_{1,j,k} & x_{2,j,k} & \dots & x_{n_1,j,k} \end{bmatrix}^T \in \mathbb{R}^{n_1} \quad \text{(Mode-1 fiber)}$$

$$\boldsymbol{x}_{i,:,k} := \begin{bmatrix} x_{i,1,k} & x_{i,2,k} & \dots & x_{i,n_2,k} \end{bmatrix}^T \in \mathbb{R}^{n_2} \quad \text{(Mode-2 fiber)}$$

$$\boldsymbol{x}_{i,j,:} := \begin{bmatrix} x_{i,j,1} & x_{i,j,2} & \dots & x_{i,j,n_3} \end{bmatrix}^T \in \mathbb{R}^{n_3} \quad \text{(Mode-3 fiber)}$$

It follows directly that for a matrix X, which is a second-order tensor, its mode-1 fibers correspond to the columns of X, while its mode-2 fibers correspond to the rows of X. *Slices* are two-dimensional subarrays of a tensor, i.e. matrices, which are created by keeping all but two indices fixed and varying the remaining indices over all entries of the corresponding modes. The slices of a third-order $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ are denoted by $X_{i,:,:}$, $X_{:,j,:}$, and $X_{:,:,k}$. The mode-3 slices of a tensor $X_{:,:,k}$ will play a special role in this thesis and will be denoted conveniently as X_k. We will refer to mode-3 slices also as *frontal slices* of a tensor. Figure 1.4 visualizes the fibers and slices of a third-order tensor.

1.3.2. Operations on Tensors

In this section, we will briefly review operations that are important to define factorization models on tensors. Two operations which are central for this matter and which will be used frequently throughout this thesis are the unfolding operation and the n-mode product. The unfolding operation transforms a tensor into a matrix representation and is defined as follows:

Definition 6 (n-Mode Unfolding, Kolda and Bader, 2009)**.** *The unfolding of a tensor* $\mathbf{X}$ *in the* n*-th mode, in the following denoted by* $X_{(n)}$*, rearranges the elements of* $\mathbf{X}$ *into a matrix, by using the mode-*n *fibers of* $\mathbf{X}$ *as the columns of the matrix* $X_{(n)}$*.*

Figure 1.5 illustrates the unfolding operation by visualizing all possible unfoldings of a $2 \times 3 \times 2$ tensor. An exact mapping of the indices of $\mathbf{X}$ to the indices of $X_{(n)}$ can be found in Kolda and Bader (2009). Often, unfolding is also referred to as *matricization* or *flattening*.

Just as matrices, tensors can be multiplied, although the notation of tensor multiplication can be quite complex. Here, we will consider only a special case which is sufficient for the

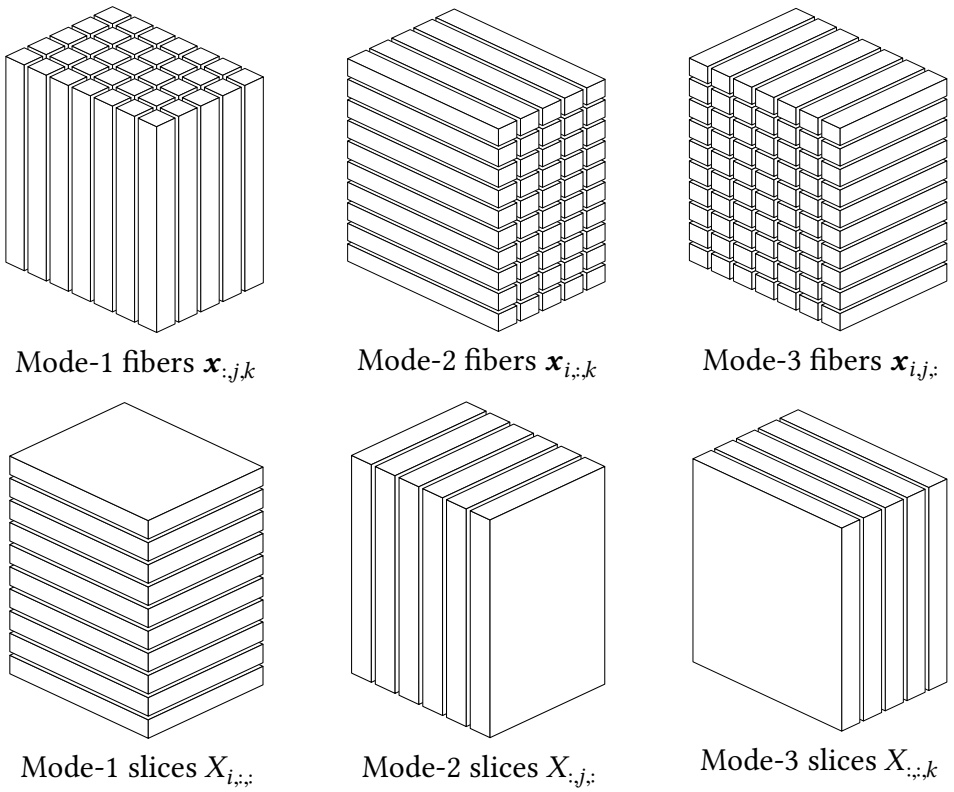


Figure 1.4.: Fibers and slices of a third-order tensor $\mathbf{X}$.

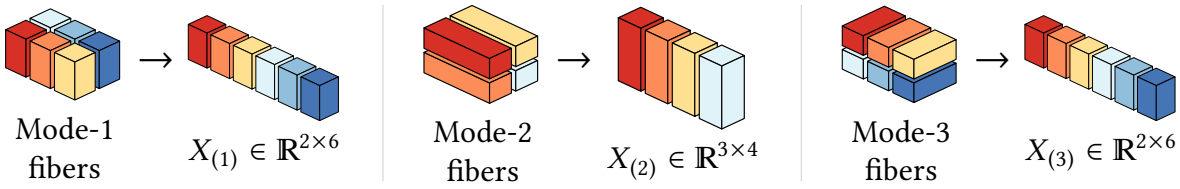


Figure 1.5.: Illustration of all possible unfoldings of a $2 \times 3 \times 2$ tensor.

course of this thesis, i.e. the n-mode product, what refers to the multiplication of a tensor with a matrix in a specific mode. The n-mode product is defined as follows:

Definition 7 (n-Mode Product, Kolda and Bader, 2009)**.** *The multiplication of a k-th order tensor $\mathbf{X} \in \mathbb{R}^{m_1 \times \ldots \times m_k}$ with a matrix $U \in \mathbb{R}^{m_n \times p}$ in mode n, in the following denoted by $\mathbf{X} \times_n U$, is defined as*

$$(\mathbf{X} \times_n U)_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_k} = \sum_{i_n=1}^{m_n} x_{i_1,\ldots,i_k} u_{ji_n}$$

where $\mathbf{X} \times_n U$ is of size $m_1 \times \cdots \times m_{n-1} \times p \times m_{n+1} \times \cdots \times m_k$. By using the unfolding operation,

an equivalent definition of the n-mode product is

$$\mathbf{Y} = \mathbf{X} \times_n U \Leftrightarrow Y_{(n)} = UX_{(n)}$$

At last, it will be useful to define the norm of a tensor, for instance to specify loss functions and regularization functions on tensors.

Definition 8 (Norm of a Tensor, Kolda and Bader, 2009)**.** *The norm of a tensor* $\mathbf{X} \in \mathbb{R}^{n_1 \times \cdots \times n_k}$*, in the following denoted by* $\|\mathbf{X}\|$*, is defined as the square root of the sum of its squared elements, i.e. as*

$$\|\mathbf{X}\| := \sqrt{\sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_k=1}^{n_k} x_{i_1,i_2,\ldots,i_k}^2}$$

1.3.3. Tensor Factorizations

The rank of a *matrix* X can be defined in multiple, equivalent ways: Commonly, it is defined as the minimal number of rank-one matrices whose sum generates X. However, the rank of a matrix can also be defined as the maximal number of linearly independent columns of X, i.e. its *column rank*, or as the maximal number of linearly independent rows of X, i.e. its *row rank*. For matrices, these definitions are equivalent and a truncated singular value decomposition (SVD) of X can be used to compute the best rank-r approximation to X for all of these definitions. Interestingly, in the case of tensors these concepts of rank are not equivalent anymore; such that they are referred to by two different terms and two different factorizations are used to compute the best approximations with regard to these concepts. One concept of the rank of tensors is called *tensor-rank* and refers to the minimal number of rank-one tensors whose sum generates $\mathbf{X}$. Although tensor-rank seems to be a natural generalization of the rank of a matrix, it is often thoroughly different from its second-order counterpart (Kolda and Bader, 2009). Most notably, it has been shown that computing the tensor-rank is $\mathcal{NP}$-complete (Håstad, 1990) and that the best rank-r approximation in terms of tensor-rank might not exist for a particular tensor (De Silva and Lim, 2008). In the following the tensor-rank of a tensor $\mathbf{X}$ will be denoted by $\text{rank}(\mathbf{X}) \in \mathbb{N}_+$. The second concept of the rank of tensors is called *n-rank*, which specifies the number of linearly independent rows for *each* mode-n unfolding of $\mathbf{X}$, such that it can be regarded as the generalization of the column and row rank of matrices. In the following, the n-rank of a tensor $\mathbf{X}$ will be denoted by $\text{n-rank}(\mathbf{X}) \in \mathbb{N}_+ \times \cdots \times \mathbb{N}_+$. To compute the best approximations according to these concepts of rank, two different tensor decompositions are used, i.e. the Candecomp / Parafac decompositions for tensor-rank and

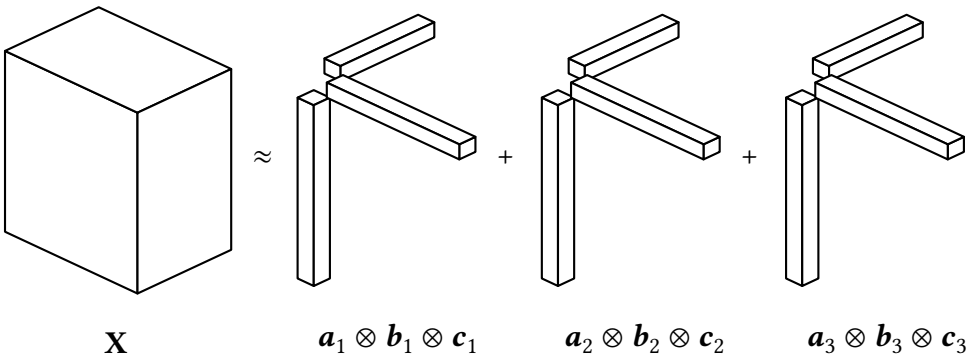


Figure 1.6.: Illustration of a CP decomposition with rank 3 of a third-order tensor $\mathbf{X}$.

the Tucker decomposition for n-rank. Both models will be relevant for the remainder of this thesis and are briefly outlined in the following subsections:

The Candecomp / Parafac Decomposition

The Candecomp / Parafac (CP) decomposition corresponds to the concept of tensor-rank and decomposes a tensor into a sum of rank-one tensors. The CP decomposition of a third-order tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is defined as

$$\mathbf{X} \approx \sum_{i=1}^{r} \boldsymbol{a}_i \otimes \boldsymbol{b}_i \otimes \boldsymbol{c}_i \tag{1.13}$$

where $\boldsymbol{a}_i \in \mathbb{R}^{n_1}$, $\boldsymbol{b}_i \in \mathbb{R}^{n_2}$, $\boldsymbol{c}_i \in \mathbb{R}^{n_3}$, and where $r \in \mathbb{N}_+$ is the rank of the decomposition. The symbol "$\approx$" is used here and in the following to indicate the best approximation under some arbitrary loss function $\Delta(\cdot;\cdot)$. In most cases, the loss function will be the least-squares loss, i.e.

$$\Delta(\mathbf{X}; \widehat{\mathbf{X}}) := \left\| \mathbf{X} - \widehat{\mathbf{X}} \right\|^2$$

where $\mathbf{X}$ denotes the original tensor and $\widehat{\mathbf{X}}$ its approximation. Figure 1.6 shows a visualization of the CP decomposition. CP has been introduced independently by Carroll and Chang (1970) and Harshman (1970), while its origins date even back to Hitchcock (1927). Although the exact determination of tensor-rank is $\mathcal{NP}$-complete, in practice the CP decomposition can be used to compute the tensor-rank *numerically* by fitting multiple factorizations (Kolda and Bader, 2009). The CP decomposition is essentially unique, meaning that except for basic scaling and permutation indeterminacies, there exists only one possible decomposition of a tensor into a sum of rank-one tensors under mild conditions. This is a very appealing property when the latent components have meaning and are sought to be analyzed, as it is commonly

the case in psycho- and chemometrics. To compute the CP decomposition of a third-order tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ under the least-squares loss, the optimization problem

$$\arg\min_{A,B,C} \left\| \mathbf{X} - \sum_{i=1}^{r} \boldsymbol{a}_i \otimes \boldsymbol{b}_i \otimes \boldsymbol{c}_i \right\|^2 \tag{1.14}$$

is minimized, where the vectors $\boldsymbol{a}_i$, $\boldsymbol{b}_i$, $\boldsymbol{c}_i$ correspond to the i-th column of $A \in \mathbb{R}^{n_1 \times r}$, $B \in \mathbb{R}^{n_2 \times r}$, and $C \in \mathbb{R}^{n_3 \times r}$. Carroll and Chang (1970) and Harshman (1970) presented and alternating-least squares algorithm to optimize equation (1.14) (CP-ALS), while Acar et al. (2010) proposed an all-in-one optimization approach (CP-OPT).

The Tucker Decomposition

The Tucker decomposition (Tucker, 1966) corresponds to the concept of n-rank and decomposes a tensor into a *core tensor* and separate *factor matrices* for each mode of the tensor. The Tucker decomposition of a third-order tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is defined as

$$\mathbf{X} \approx \mathbf{G} \times_1 A \times_2 B \times_3 C \tag{1.15}$$

where $\mathbf{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is the core tensor of the decomposition and the matrices $A \in \mathbb{R}^{n_1 \times r_1}$, $B \in \mathbb{R}^{n_2 \times r_2}$, $C \in \mathbb{R}^{n_3 \times r_3}$ are the factor matrices for the three modes and where (r_1, r_2, r_3) is the n-rank of the factorization. When operating on unfolded tensors, equation (1.15) can be rewritten as

$$X_{(1)} \approx A G_{(1)} (C \otimes B)^T, \tag{1.16}$$

$$X_{(2)} \approx B G_{(2)} (C \otimes A)^T, \tag{1.17}$$

$$X_{(3)} \approx C G_{(3)} (B \otimes A)^T. \tag{1.18}$$

which is useful to derive algorithms for the computation of the decomposition. Amongst others, the Tucker decomposition is also known as Higher-Order SVD (Hosvd) as introduced in De Lathauwer et al. (2000a) and Three-Mode PCA (3mpca) as introduced in Kroonenberg and De Leeuw (1980).

In contrast to CP, the Tucker decomposition is generally not unique. However, different methods such as super-diagonalization of the core tensor or simultaneous rotations of the core tensor and factor matrices have been developed to improve the uniqueness of the factorization. Two variations of the Tucker decomposition which are important for the remainder of this thesis are the Tucker-2 and Tucker-1 decompositions of third-order tensors. In the case of

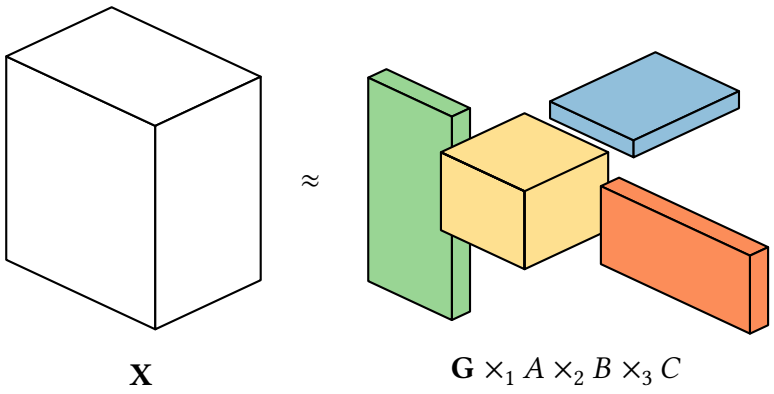


Figure 1.7.: Illustration of a Tucker Decomposition of a third-order tensor $\mathbf{X}$ into a core tensor $\mathbf{G}$ (yellow) and latent factors A (green), B (orange), C (blue).

Tucker-2, one factor matrix in the decomposition of a third-order tensor is constrained to be the identity matrix. In tensor notation, the Tucker-2 model can therefore be stated as

$$\mathbf{X} \approx \mathbf{G} \times_1 A \times_2 B \times_3 I = \mathbf{G} \times_1 A \times_2 B \tag{1.19}$$

From a modeling perspective, the Tucker-2 model can be interpreted as assuming no redundancies in the mode where the factor matrix is the identity matrix, i.e. in the mode that is not factorized. In the case of Tucker-1, two factor matrices of the decomposition are constrained to be the identity matrix, such that it can be written in tensor notation as

$$\mathbf{X} \approx \mathbf{G} \times_1 A \times_2 I \times_3 I = \mathbf{G} \times_1 A \tag{1.20}$$

Please note that Tucker-1 can be regarded as a standard matrix factorization of the unfolded tensor, since it can be written in matrix notation as

$$X_{(1)} \approx AG_{(1)}$$

1.4. Contributions of this Thesis

Relational models have been shown on numerous occasions to greatly improve learning results when relational information is available, as discussed in the previous sections. While this underlines the importance of relational learning, existing models also exhibit deficiencies that limited their wider application. In section 1.2.3 we argued that their limited scalability as well as their need for extensive prior knowledge are of particular concern. Considering that relational data is generated in an unprecedented amount – a fact often referred to by

the newly coined term "Big Data" – this has serious ramifications for their applicability. A single knowledge base in the Linked Open Data (LOD) cloud can consist of millions of entities, hundreds of relations and even billions of known facts and it would be very difficult to apply most of these methods to relational data of this size. To overcome these problems and to enable new applications for relational learning, this thesis is concerned with relational learning via tensor factorization. Its main contributions are the following:

Tensor Model for Relational Learning In section 1.2.1 we outlined the importance of collective learning to exploit the patterns in relational data efficiently. To enable collective learning via tensor factorization we propose a novel factorization model called RESCAL in section 2.2. We will show that due to its special structure – where entities have a unique representation in the factorization – RESCAL can be considered one of the first tensor models that can combine collective learning with a strong learning performance on general multi-relational data. Moreover, we will show that the factorization offers great flexibility in how relational learning tasks can be approached as it allows for the application of feature-based machine learning methods to relational problems as discussed in section 2.3. In section 2.5 we derive an efficient algorithm to compute the factorization.

Large-Scale Relational Learning on Semantic Web Data Learning from large-scale relational data has become especially important with the advent of the Semantic Web and large multi-relational knowledge bases. By exploiting the inherent sparsity of relational data, we show in section 3.4 that the RESCAL model can be computed very efficiently, such that it scales linearly with the number of entities, the number of predicates and the number of known facts. As one of the first relational models, RESCAL can therefore be applied to large knowledge bases consisting of millions of entities, hundreds of relations, and possibly billions of known facts. Furthermore, based on a thorough analysis of its runtime complexity, we provide an improved algorithm to compute the RESCAL factorization in section 3.5. We will show that this improved algorithm decreases the runtime complexity from $O(r^5)$ to $O(r^3)$, where r represents the complexity of the factorization in term of latent components. In section 3.6 we propose an efficient method to handle the attributes of entities in the RESCAL model via a novel coupled tensor-matrix factorization.

Learning Theory for Tensor Models Although tensor factorizations have become increasingly popular for learning on various forms of structured data, only very few theoretical results exist on the generalization abilities of these methods. In chapter 4 we provide therefore a theoretical analysis of tensor methods for learning from structured and relational data. In section 4.2.1, we discuss the tensor product as a principled way to obtain vector space representations of structured data for machine learning tasks. In section 4.3, we provide the first known generalization error bounds for tensor factorizations in a classification setting, what is essentially equivalent to the link-prediction task on multi-relational data. Furthermore, in section 4.4 we analyze analytically and experimentally how tensor factorization behaves when applied to over- and understructured representations, for instance, when two-way tensor factorization, i.e. matrix factorization, is applied to three-way tensor data. This analysis suggests that structuring vector space representations via the tensor product, up to the true order of the data, adds important information such that factorizations of these representations often scale better with sparsity and missing data than their less structured counterparts.

Publications accompanying the contributions of this thesis are listed in the following:

- **Tensor Model for Relational Learning**
  - M. Nickel, V. Tresp, and H.-P. Kriegel. "A Three-Way Model for Collective Learning on Multi-Relational Data". In: *Proceedings of the 28th International Conference on Machine Learning*. ICML '11. Bellevue, WA, USA: ACM, 2011, pp. 809–816. ISBN: 978-1-4503-0619-5.
  - M. Nickel and V. Tresp. "Tensor Factorization for Multi-Relational Learning". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013*. Ed. by H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezny. Nectar track for "high-quality research related to machine learning". Springer, 2013, to appear.
  - M. Nickel and V. Tresp. "Three-Way DEDICOM for Relational Learning". In: *NIPS 2010 Workshop - Tensors, Kernels and Machine Learning*. Whistler, Canada, 2010.
  - M. Nickel and V. Tresp. "Logistic Tensor-Factorization for Multi-Relational Data". In: *ICML Workshop - Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*. Atlanta, GA, USA, 2013.

- **Large-Scale Relational Learning on Semantic Web Data**
  - M. Nickel, V. Tresp, and H.-P. Kriegel. "Factorizing YAGO: scalable machine learning for linked data". In: *Proceedings of the 21st international conference on World Wide Web*. WWW '12. New York, NY, USA: ACM, 2012, pp. 271–280. ISBN: 978-1-4503-1229-5. .
  - Y. Huang, V. Tresp, M. Nickel, A. Rettinger, and H.-P. Kriegel. "A scalable approach for statistical learning in semantic graphs". In: *Semantic Web* (2013), to appear. .
  - M. Nickel and V. Tresp. "Learning Taxonomies from Multi-Relational Data via Hierarchical Link-Based Clustering". In: *NIPS Workshop - Learning Semantics*. Granada, Spain, 2011.
- **Learning Theory for Tensor Models**
  - M. Nickel and V. Tresp. "An Analysis of Tensor Models for Learning on Structured Data". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013*. Ed. by H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezny. ECML/PKDD'13. Springer, 2013, to appear.
- **Tutorials**
  - M. Nickel and V. Tresp. *Machine Learning on Linked Data: Tensors and their Applications in Graph-Structured Domains*. Tutorial at the 11th International Semantic Web Conference. Boston, MA, USA, 2012. URL: `http://www.cip.ifi.lmu.de/~nickel/iswc2012-learning-on-linked-data/`.
- **Overview Chapters in Books**
  - V. Tresp and M. Nickel. "Relational Models". In: *Encyclopedia of Social Network Analysis and Mining*. Ed. by J. Rokne and R. Alhajj. Heidelberg: Springer, 2013, to appear.
  - V. Tresp, Y. Huang, and M. Nickel. "Querying the Web with Statistical Machine Learning". In: *To be published as a chapter in a book reviewing the results of the THESEUS project*. 2013.

Chapter 2

A Three-Way Model for Relational Learning

In this chapter, we propose a novel approach to relational learning via the factorization of a third-order tensor. The focus of this chapter lies on its properties, and its benefits for statistical relational learning. We will show that the proposed method can overcome problems of existing SRL methods, that it offers a great flexibility in how relational learning tasks can be approached and, maybe most importantly, that it learns from relational data very efficiently. Due to the structure of the factorization, we will show that the model is able to perform collective learning and that it can improve the learning results on relational data significantly over standard tensor factorizations such as CP and Tucker. In this chapter, we will only consider entity-entity relations and focus on the learning aspect of the factorization. The handling of attributes as well as the discussion of its scalability is deferred until chapter 3.

2.1. Introduction

During the review of relational models in section 1.2.3, we discussed some problems of existing SRL methods that hindered their wider adoption, such as their extensive need for prior knowledge, their complexity and their limited scalability. In this chapter, we propose a novel approach to relational learning via the factorization of a third-order tensor to overcome aforementioned problems and to enable new applications for statistical relational learning. The motivation to employ a factorization approach, and in particular tensor factorizations, for this purpose is manifold. Matrix factorization can be interpreted as maximum likelihood

Table 2.1.: Data statistics of recent recommendation challenges.

Dataset	Dimensions	Entries	Density
Netflix	480,189 users × 17,770 items	100,480,507	0.0118
KDD Cup 2011	1,000,990 users × 624,961 items	262,810,175	0.0004
KDD Cup 2012	2,320,895 users × 6,095 items	73,209,277	0.0052

(MLE) or maximum a posteriori (MAP) estimation in a graphical model, where the estimated parameters correspond to latent representations of the objects and attributes in the data (Singh and Gordon, 2008a; Salakhutdinov and Mnih, 2008a; Salakhutdinov and Mnih, 2008b). Tensor factorizations can be interpreted in a similar way (Xiong et al., 2010). As discussed in section 1.2.3, an important advantage of latent variable models for relational learning is that they do not require structure learning, as the dependency structure between relationships is already defined through their model. For factorization methods, this dependency structure is determined through the structure of the factorization, i.e. how an observed matrix or tensor is approximated by the product of latent factors. A focal point of this chapter lies on the development of a factorization whose dependency structure allows for collective learning on relational data.

Another strong motivation to use a factorization approach for relational learning is that relational data is usually high-dimensional and sparse. For instance, there may be many persons in a relational data set, but only a small number of them will be friends. Similarly, there may be many items, but a single person will usually have bought only a small subset of all possible items. This sparsity of relational data is well-known and has, for instance, been exploited by Singla and Domingos (2006b) to improve the memory efficiency of MLNs. In this setting of high-dimensional and sparse data, factorization methods have shown very good results. The winning submission to the *Netflix Grand Prize* was heavily based on matrix factorization (Koren, 2009), as were most of the leading solutions to the *KDD Cup* of 2011 and 2012 (Dror et al., 2011; Chen et al., 2012). Basic statistics about the associated data sets are available in Table 2.1 which shows their high-dimensional and sparse nature. The size of these data sets also indicates that factorization methods can not only be expected to provide good learning results, but also might be able to overcome the scalability problem of SRL methods.

From a modeling perspective, tensors are appealing because they provide an elegant way to represent multiple dyadic relations. Similar to the graph-based interpretation of RDF (Anto-

niou and Harmelen, 2004, Section 3.2.4), dyadic relational data can be interpreted as a labeled directed multigraph, where entities are represented as nodes, relations are represented as edge labels and relationships are represented as labeled directed edges which point from the subject to the object of a relationship. Under this interpretation, dyadic relational data consisting of n entities and m different relations can then be represented in form of an *adjacency tensor* $\mathbf{X} \in \mathbb{R}^{n \times n \times m}$ with entries

$$x_{ijk} = \begin{cases} 1, & \text{if the relationship } \texttt{relation}_k(\texttt{entity}_i, \texttt{entity}_j) \text{ exists} \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

The first mode of an adjacency tensor models therefore the occurrences of all entities as a subject, the second mode models the occurrences of all entities as an object, and the third mode models the different relations in a data set. This interpretation of relational data as a multigraph is not only interesting from a data representation point of view, but serves also as an additional motivation to use a factorization approach for relational learning. On graphs, factorizations of (weighted) adjacency *matrices* have long been used for tasks like information retrieval (Brin and Page, 1998; J. M. Kleinberg, 1999), link prediction (Hoff et al., 2002; Liben-Nowell and J. Kleinberg, 2007), community detection (Sarkar and Dong, 2011) and clustering (Ng et al., 2002). Moreover, based on the CP tenor factorization, Kolda et al. (2005) proposed an extension of the Hypertext-Induced Topic selection (HITS) algorithm of J. M. Kleinberg (1999) to multigraphs. Franz et al. (2009) applied this extension for information retrieval on Semantic Web data. Bader et al. (2007) used the decomposition into directional components (DEDICOM) tensor factorization to analyze time-varying graph which are modeled as multigraphs, where each slice of the adjacency tensor corresponds to a certain time period.

In section 1.2.1, we emphasized the importance to exploit relational patterns such as homophily, stochastic equivalence for relational learning as well as the modeling of global dependencies between relationships. In this chapter, we present a novel tensor factorization model, for which we will show that it fulfil these requirements. Central to the proposed model is the unique latent representation of entities in the factorization, what enables a strong collective learning effect, such that the proposed method provides significantly better learning results on relational data than standard factorization methods like CP and TUCKER on relational data. We will also show that our approach provides better or on-par results on commonly used benchmark data sets for relational learning compared to current state-of-the art SRL methods.

The remainder of this chapter is organized as follows: In section 2.2 we will describe the

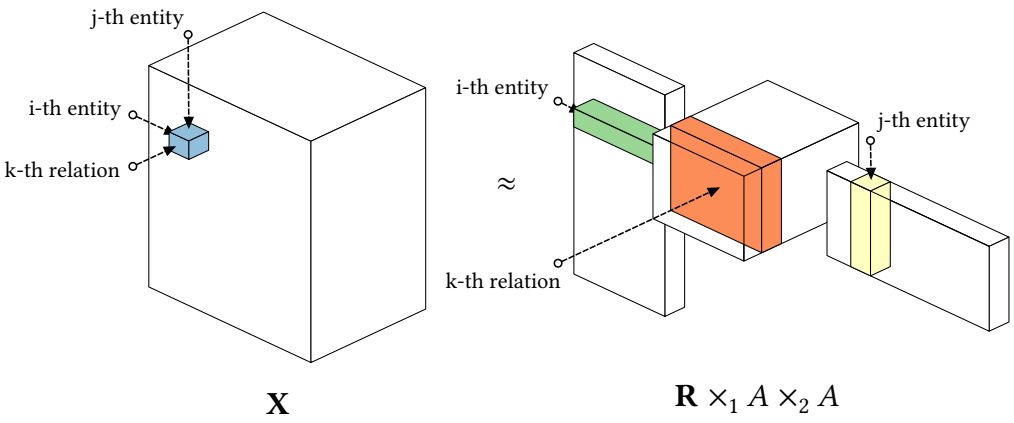


Figure 2.1.: The RESCAL tensor factorization for relational learning. The state of an individual relationship x_{ijk} (blue cell in $\mathbf{X}$) is predicted from the product of latent factors $\boldsymbol{a}_i$ (green), $\boldsymbol{a}_j$ (yellow) and R_k (orange).

model of the factorization and discuss its properties from a relational learning point of view. In section 2.3 we will discuss how canonical relational learning tasks can be approached by using the factorization. section 2.4 we will put the proposed approach in the context of state-of-the-art methods for relational learning and tensor factorizations. In section 2.5 we will present an efficient algorithm to compute the factorization. At last, in section 2.6, we evaluate the our approach through various experiments on benchmark data sets.

2.2. The RESCAL Factorization

Here, we propose RESCAL, a novel approach for learning from dyadic multi-relational data via the factorization of a third order tensor. Given relational data consisting of n entities and m dyadic relations, RESCAL computes a factorization of the associated adjacency tensor $\mathbf{X} \in \mathbb{R}^{n \times n \times m}$ into a single factor matrix $A \in \mathbb{R}^{n \times r}$ and a core tensor $\mathbf{R} \in \mathbb{R}^{r \times r \times m}$ such that

$$\mathbf{X} \approx \mathbf{R} \times_1 A \times_2 A, \tag{2.2}$$

where r is a user-given positive integer with $0 < r < n$ which specifies the complexity the model. The symbol "$\approx$" denotes the approximation under a given loss function. Figure 2.1 shows a visualization of an adjacency tensor and its factorization according to equation (2.2). In the following, it will often prove convenient to consider an alternative specification of the factorization. It can be seen from equation (1.19) that equation (2.2) can be interpreted as a TUCKER-2 factorization with the additional constraint that the factor matrices in equation (1.19) are identical. It follows from the unfolded representation of TUCKER models and the product

of block-partitioned matrices that equation (2.2) is equivalent to

$$X_k \approx AR_kA^T \text{ , for all } k = 1 \dots m, \tag{2.3}$$

where X_k and R_k denote the k-th frontal slice of $\mathbf{X}$ and $\mathbf{R}$ respectively. In appendix A.1 we provide a detailed derivation of equation (2.3) from equation (2.2). Furthermore, it also follows from equations (2.2) and (2.3) that an individual element x_{ijk} of $\mathbf{X}$ is approximated by

$$x_{ijk} \approx \boldsymbol{a}_i^T R_k \boldsymbol{a}_j \tag{2.4}$$

where the column vectors $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$ correspond to the i-th and j-th row of A respectively.

Equation (2.3) provides a convenient form for the interpretation of the RESCAL model for relational learning. Each frontal slice X_k of an adjacency tensor $\mathbf{X}$ as constructed in equation (2.1) corresponds to the adjacency *matrix* of the k-th relation in the data. It can be seen from equation (2.3) that these adjacency matrices are jointly factorized, such that A factorizes all frontal slices of $\mathbf{X}$ simultaneously. The matrix A can be viewed as an embedding of the entities in a data set into an r-dimensional latent space, where the i-th row $\boldsymbol{a}_i$ of A corresponds to the latent representation of the i-th entity. In analogy to the principal components of PCA, we will refer to the columns of A as *latent components*.[1] Each frontal slice R_k of $\mathbf{R}$ then specifies how these latent components interact for the k-th relation. Since R_k is a full asymmetric matrix, adjacency tensors with asymmetric frontal slices, i.e. tensors representations of directed multigraphs, can be factorized. In terms of relational data, the asymmetry of R_k models the different interactions whether a latent component occurs in the subject- or object-position in a relationship. The RESCAL factorization can therefore be interpreted as a latent factor model which learns an embedding of entities into a latent space and simultaneously learns how the components of this embedding interact for different relations. The embedding and interactions are optimized such that the observed data $\mathbf{X}$ is best explained according to a particular loss function. Here, we will employ the least-squares loss, which is commonly used for tensor factorizations and which has also been applied for the factorizations of graphs and multigraphs (Kolda et al., 2005; Liben-Nowell and J. Kleinberg, 2007; Bader et al., 2007; Franz et al., 2009; Huang et al., 2011). More precisely, we approximate

[1] Please note that unlike in prinicpal component analysis (PCA), the columns of A are *not* required to be mutually orthogonal.

$\mathbf{X}$ by computing the solution to the optimization problem

$$\arg\min_{A,\mathbf{R}} \|\mathbf{X} - \mathbf{R} \times_1 A \times_2 A\|^2 + \lambda_A \|A\|^2 + \lambda_{\mathbf{R}} \|\mathbf{R}\|^2. \tag{2.5}$$

In equation (2.5), the term $\|\mathbf{X} - \mathbf{R} \times_1 A \times_2 A\|^2$ measures the quality of the approximation under the least-squares loss, while the regularization terms $\lambda_A \|A\|^2$ and $\lambda_{\mathbf{R}} \|\mathbf{R}\|^2$ are added to prevent overfitting of the model. An efficient algorithm to compute equation (2.5) will be provided in section 2.5. Prior to that, we will discuss important properties of the factorization in equation (2.2) and the optimization problem in equation (2.5) with regard to relational learning.

Probabilistic Interpretation

It has been shown by Singh and Gordon (2008a) that for appropriate loss functions[1], many matrix factorizations can be interpreted probabilistically in the following way: Each entry x_{ij} of a matrix X can be viewed as a random variable that follows an exponential family distribution with natural parameter θ_{ij}. A factorization $X \approx \widehat{X}$ then maximizes the log-likelihood of the observed data where the natural parameter θ_{ij} corresponds to the reconstructed entry $\widehat{x}_{ij}$ after factorization. The chosen loss function determines the nature of the exponential family distribution. Since, the Tucker-2 structure of Rescal allows to specify its model in form of a matrix factorization, it is straightforward to extend this interpretation to Rescal: For an appropriate loss function, equation (2.2) can be viewed as maximizing the log-likelihood of the joint distribution

$$\mathrm{P}(\mathbf{X} \mid A, \mathbf{R}) = \prod_{i=1}^{n} \prod_{j=1}^{n} \prod_{k=1}^{m} \mathrm{P}\left(x_{ijk} \,\middle|\, \boldsymbol{a}_i^T R_k \boldsymbol{a}_j\right) \tag{2.6}$$

where each entry x_{ijk} is drawn from an exponential family distribution with natural parameter $\boldsymbol{a}_i^T R_k \boldsymbol{a}_j$. Figure 2.2 shows the associated graphical model for this distribution in plate notation. Analogously to the discussion on probabilistic matrix factorization in Salakhutdinov and Mnih (2008a), equation (2.5) can therefore be interpreted as computing the MAP estimate for A and $\mathbf{R}$, where the observable variables x_{ijk} are drawn from a normal distribution with

$$x_{ijk} \sim \mathcal{N}(\boldsymbol{a}_i^T R_k \boldsymbol{a}_j, \sigma^2),$$

[1] To be precise, for loss functions from the family of decomposable regular Bregman divergences, which subsumes loss functions such as the least-squares loss and the logistic loss.

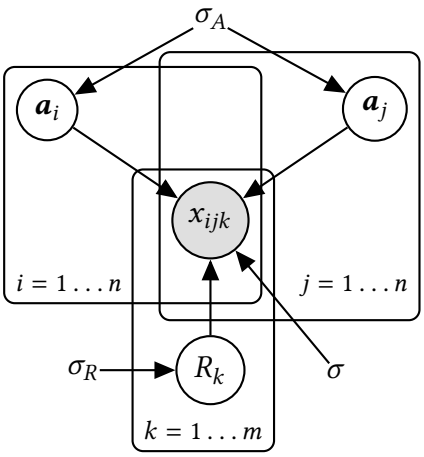


Figure 2.2.: Graphical model of the Rescal factorization in plate notation. Nodes in the graphical model represent variables, where shaded nodes represent observable variables and non-shaded nodes represent latent variables. Plates indicate that the enclosed subgraph is repeated multiple times. Constants are represented as simple letters.

and where the latent variables $\boldsymbol{a}_i$, R_k are also drawn from a normal distribution with

$$\boldsymbol{a}_i \sim \mathcal{N}(0, \sigma_A^2 I)$$
$$R_k \sim \mathcal{N}(0, \sigma_R^2 I).$$

This interpretation follows from the choice of the least-squares loss and the ℓ_2 regularization terms in equation (2.5). Furthermore, the variance parameters of the normal distributions are determined by the user-given regularization parameters, such that $\lambda_A = \frac{\sigma^2}{\sigma_A^2}$ and $\lambda_{\mathbf{R}} = \frac{\sigma^2}{\sigma_R^2}$. It can be seen that by choosing the least-squares loss to approximate a tensor $\mathbf{X}$, we assume that the random variation in the data is normally distributed, although a Bernoulli distribution would be a more appropriate choice for binary random variables like x_{ijk}. Please note the we made this choice deliberately, as it enables an efficient and scalable algorithm to compute the factorization and we will employ this approximation throughout this and the following chapter. Given the factorization of an adjacency tensor, we take a similar approach as considered very successfully by Liben-Nowell and J. Kleinberg (2007) and Huang et al. (2011) and interpret the entries of $\widehat{\mathbf{X}} = \mathbf{R} \times_1 A \times_2 A$ as *confidence values* that a particular relationship exists, i.e. that

$$\mathrm{P}\left(x_{ijk} = 1 \,\middle|\, \boldsymbol{a}_i, \boldsymbol{a}_j, R_k\right) \propto \boldsymbol{a}_i^T R_k \boldsymbol{a}_j. \tag{2.7}$$

In section 5.2 we will briefly present a variant of the factorization that explains the random variation in the data via a Bernoulli distribution and discuss its benefits but also its considerable shortcomings for large-scale relational learning. Furthermore, please note that the particular choice of a loss function does not change the dependency structure of the graphical model in figure 2.2, as this structure is defined through the computation of the natural parameter, i.e. through the structure of the factorization.

Global dependencies

For multi-relational data, each possible relationship corresponds to exactly one entry x_{ijk} in an adjacency tensor $\mathbf{X}$, such that the joint distribution in equation (2.6) represents a complete model of a relational domain. An important feature of RESCAL is that its latent variable structure decouples inference in this relational model, meaning that global dependencies are captured during learning the latent representations A and $\mathbf{R}$, whereas the prediction of relationships x_{ijk} relies only on a few latent variables. It can be seen from equation (2.6) that a variable x_{ijk} is conditionally independent from all other variables given $\boldsymbol{a}_i^T R_k \boldsymbol{a}_j$, what enables fast query answering independently of the size of a data set and what can essentially be computed in real-time. However, it is important to note that this locality of computation does not imply that the confidence values for relationships are only influenced by local information. On the contrary, the conditional independence assumptions depicted in figure 2.2 show that information is propagated globally when computing the factorization. Due to the repeated colliders in figure 2.2, latent variables can not be d-separated from any other latent or observable variable and thus are possibly dependent on all of these variables (Pearl, 2009, Definition 1.2.3, Theorem 1.2.4). Since the random variable x_{ijk} depends on the latent variables $\{\boldsymbol{a}_i, \boldsymbol{a}_j, R_k\}$, it depends therefore *indirectly* on the state of any other variable, such that global dependencies between relationships can be captured.

Latent Similarity of Entities and Relations

An important feature of RESCAL is that the latent space A reflects the similarity of entities in the relational domain. In relational data, the similarity of entities is determined by the similarity of their relationships, following the intuition that "if two objects are in the same relation to the same object, this is evidence that they may be the same object" (Singla and Domingos, 2006a). In terms of an adjacency tensor $\mathbf{X}$, this notion of similarity is captured by the similarity of the mode-1 and mode-2 slices of the respective entities as shown in figures 2.3a and 2.3b. Due to its TUCKER-2 structure, it follows from equation (1.16) and

equation (1.17) that the mode-1 and mode-2 unfoldings of the Rescal model are given by

$$X_{(1)} \approx AR_{(1)}(I \otimes A)^T$$
$$X_{(2)} \approx AR_{(2)}(I \otimes A)^T$$

From this it follows immediately that the mode-1 and mode-2 slices for the i-th and j-the entity are modeled in Rescal by the following matrix products:

$$\text{vec}\left(X_{i,:,:}\right) \approx \boldsymbol{a}_i R_{(1)}(I \otimes A)^T \qquad \text{vec}\left(X_{:,i,:}\right) \approx \boldsymbol{a}_i R_{(2)}(I \otimes A)^T$$
$$\text{vec}\left(X_{j,:,:}\right) \approx \boldsymbol{a}_j R_{(1)}(I \otimes A)^T \qquad \text{vec}\left(X_{:,j,:}\right) \approx \boldsymbol{a}_j R_{(2)}(I \otimes A)^T.$$

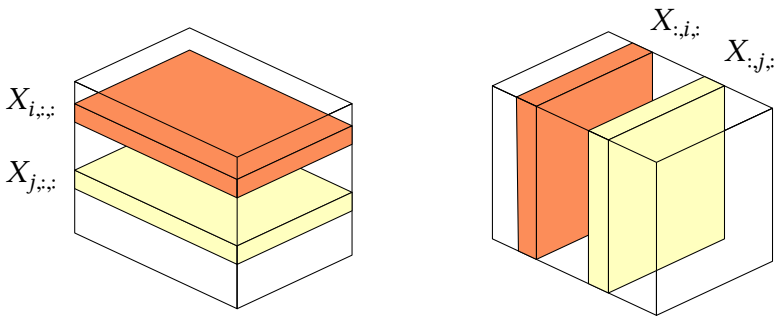
Since the terms $R_{(1)}(I \otimes A)^T$ and $R_{(2)}(I \otimes A)^T$ are constant for different values of i and j, it follows that similar relationship patterns in $X_{i,:,:}$ and $X_{j,:,:}$ as well as $X_{:,i,:}$ and $X_{:,j,:}$ lead to similar latent representations $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$. Moreover, a similar argument can be made for the similarity of relations. In an adjacency tensor, the k-th relation in a data set is represented via the frontal slice X_k. It follows from the application of equation (1.12) to equation (2.3) that the frontal slices of the i-th and j-th relation are modeled in Rescal by

$$\text{vec}\left(X_i\right) \approx (A \otimes A)\,\text{vec}\left(R_i\right)$$
$$\text{vec}\left(X_j\right) \approx (A \otimes A)\,\text{vec}\left(R_j\right).$$

Since the term $A \otimes A$ is constant for different values of i and j it follows that the similarity of the interaction matrices R_i and R_j reflects the similarity of the i-th and j-th relation. To determine the relational similarity of the i-th and j-th entity, it is therefore sufficient to consider only the similarity of the latent representations $\boldsymbol{a}_i$, $\boldsymbol{a}_j$. Importantly, as this measure of similarity is based on latent representations, it also takes the similarity of entities and relations into consideration, what can be seen from the element-wise formulation of the Rescal model for two different relationships x_{ijk} and x_{qrs}:

$$x_{ijk} \approx \boldsymbol{a}_i^T R_k \boldsymbol{a}_j$$
$$x_{qrs} \approx \boldsymbol{a}_q^T R_s \boldsymbol{a}_r.$$

If these relationships are in the same state and the relations and objects are similar to each other, i.e. when $x_{ijk} = x_{qrs}$, $R_k \approx R_s$, and $\boldsymbol{a}_j \approx \boldsymbol{a}_r$, the optimal values for $\boldsymbol{a}_i$ and $\boldsymbol{a}_q$ will also be similar to each other. By measuring the similarity of entities through their latent similarity, this measure is therefore not only based on counting identical relationships of identical entities, but it also considers the similarity of the entities and relations that are involved in



(a) Outgoing relationships *(b)* Incoming relationships

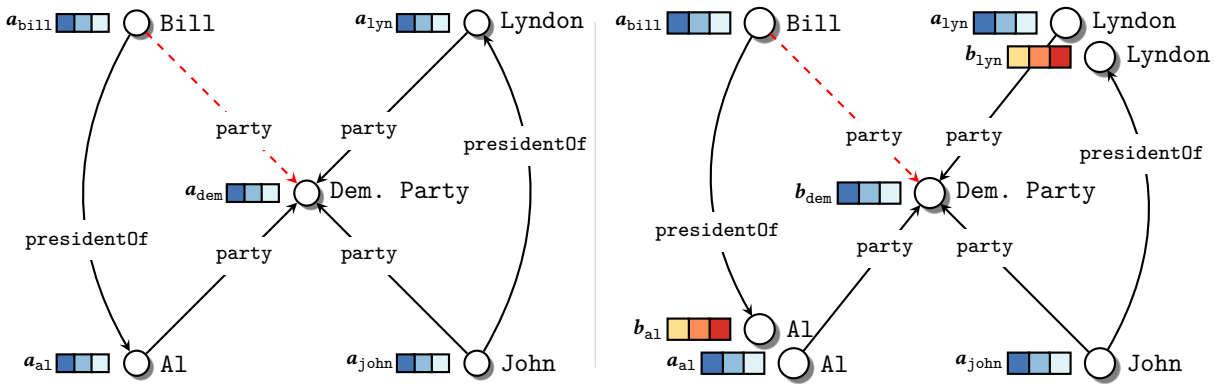
Figure 2.3.: The mode-1 and mode-2 slices of an adjacency tensor group incoming and outgoing relationships for a specific entity. All possible relationships in which the i-th entity occurs as a subject are grouped in the mode-1 slice $X_{i,:,:}$, whereas all possible relationships in which it occurs as an object are grouped in the mode-2 slice $X_{:,i,:}$.

a relationship. The previous intuition of Singla and Domingos (2006a) could therefore be restated in our approach as: *if two objects are in similar relations to similar objects, this is evidence that they may be the same object.*

Unique Representation of Entities

A central aspect that distinguishes Rescal from tensor factorizations such as CP or Tucker is that two modes are factorized by the identical matrix A. For an adjacency tensor, this means that entities have a *unique representation* via the latent space A, regardless of their occurrence as a subject or an object in a relationship.[1] This property of the Rescal factorization is of great importance for its relational learning capabilities, as it enables the efficient propagation of information via these latent representations of entities. It has been discussed previously that latent variables can not be d-separated from any other variable in figure 2.2, such that global dependencies can be captured via information propagation through this network of variables. Due to importance of this propagation effect for the learning process, it is crucial that network of latent variables has the correct structure. Standard tensor factorization models such as CP and Tucker compute a bipartite factorization of an adjacency tensor, meaning that these factorizations assume that the underlying multigraph of an adjacency tensor is bipartite, because the employ different factor matrices for each mode. For instance, a standard Tucker-2 model would factorize each frontal slice X_k of an adjacency tensor $\mathbf{X}$

[1]Please note that we use the term "unique representation" to refer to the fact that entities are represented by a single factor matrix A and *not* to refer to a unique *factorization* in the sense of section 1.3.3. In fact, the Rescal factorization itself is not unique as it will be shown later in this chapter.



(a) Unipartite Model AR_kA^T *(b)* Bipartite Model AR_kB^T

Figure 2.4.: Visualization of the U.S. presidents collective learning example. Figures 2.4a and 2.4b show unipartite and bipartite modelings of an examplary U.S. presidents domain. Each entity is represented as a node in a multigraph and relationships are represented as links between nodes. Furthermore, for each entity, we show the associated latent representation consisting of three latent components. Similar colors in the latent components indicate similar values. Since a bipartite model treats subject and object as different entities, we introduce separate nodes for subject- and object-occurances of the same entity in figure 2.4b. For clarity of presentation, only the relevant parts of the multigraph are shown.

such that $X_k \approx AR_kB^T$. However, when this bipartite modeling is applied to unipartite data – what is usually the case for multi-relational data – the same entities would have different representations whether they occur subjects (through the latent factor A) or objects (through the latent factor B). Unfortunately, this effectively breaks the flow of information from subjects to objects, as the modeling does not account for the fact that the latent variables $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ refer to the identical entity. For instance, consider the task of predicting the party membership of a president of the United States of America. Without any additional information, this could be done quite accurately when the party of the president's vice president is known, since both persons have mostly been members of the same party. Figure 2.4 shows a small excerpt of an exemplary data set. Recall from the previous discussion that the latent variable modeling decouples the learning process, such that the state of an unknown relationship x_{ijk} depends only on the product of the associated latent variables. In an unipartite modeling, the latent representation $\boldsymbol{a}_{\texttt{al}}$ would encode the information that `Al` is a member of `Dem. Party`, as it has to account for the fact $\boldsymbol{a}_{\texttt{al}}^T R_{\texttt{party}} \boldsymbol{a}_{\texttt{dem.}} \approx 1$. When the party membership of `Bill` is unknown, a unipartite model could then access the information that his vice president is a member of `Dem. Party` via the `presidentOf` relation, since $\boldsymbol{a}_{\texttt{bill}}^T R_{\texttt{presidentOf}} \boldsymbol{a}_{\texttt{al}} \approx 1$

where $\boldsymbol{a}_{\texttt{al}}$ encodes the party membership of `Al`. This way, the party membership information could propagate over the latent variables, such that the model can learn the correct latent representation of `Bill`, which in turn leads to the correct party membership prediction. In a bipartite model this method of information propagation would not work as entities have different representations as subjects or objects. The latent representation $\boldsymbol{a}_{\texttt{al}}$ would still encode that `Al` is a member of `Dem. Party`. However, all the information that $\boldsymbol{a}_{\texttt{bill}}$ could access for the entity `Al` is $\boldsymbol{b}_{\texttt{al}}$, what does not encode the necessary information, as `Al` occurs only as a subject in relationships to `Dem.Party` and not as an object. Consequently, the structure of Rescal allows the factorization to access information that is more distant in the relational graph, while the information flow for bipartite models would effectively break over subject-object-boundaries.

Modeling Relational Patterns

It has been emphasized in section 1.2.1 that relational patterns such as homophily and stochastic equivalence are an important characteristic of relational data, whose exploitation can be essential for relational learning. In this section, we will demonstrate how these patterns *can* be modeled within Rescal, to show that the model is expressive enough to capture these patterns. Please note that the following discussion shall not imply that these patterns *have* to be modeled this way, i.e. we only discuss the expressiveness of Rescal and not its interpretability. For this reason, we are free to choose any form of representation and only have to show that the considered patterns can be modeled within this representation. To improve the clearness of presentation, we chose to present these patterns in form of vectors $\boldsymbol{a}_i$ whose entries sum to one and which are confined to the interval $[0, 1]$.

Stochastic Equivalence Stochastic equivalence can be modeled within Rescal, by interpreting the latent components as (soft) clusters of entities, while the cluster memberships of an entity are represented by the magnitude of its entries in a particular latent component. The probability of interactions between clusters can then be modeled via the entries in R_k, such that the probability of a particular relationship is proportional to $\boldsymbol{a}_i^T R_k \boldsymbol{a}_j$. For instance, consider the following interaction matrix R_k and entity representations $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{a}_3$:

$$R_k = \begin{bmatrix} 0.1 & 0.8 \\ 0.2 & 0.1 \end{bmatrix}, \boldsymbol{a}_1 = \begin{bmatrix} 0.9 \\ 0.1 \end{bmatrix}, \boldsymbol{a}_2 = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, \boldsymbol{a}_3 = \begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}.$$

In this simple example, a link from the first latent component to the second latent

component is very likely, such that the relationship $\texttt{relation}_k(\texttt{entity}_1, \texttt{entity}_2)$ is likely to exist, whereas a relationship from $\texttt{entity}_1$ to $\texttt{entity}_3$ is very unlikely. This modeling is very similar to the modeling of stochastic equivalence employed in IRM and stochastic block models.

Homophily It has been discussed previously that the similarity of entities in the latent space A reflects their similarity in a relational domain. In RESCAL, homophily can therefore be modeled by a nearly diagonal interaction matrix R_k. Consider latent representations of entities, all of similar length $\|\boldsymbol{a}_i\| \approx \alpha$. It follows from $R_k \approx I$ that $\boldsymbol{a}_i^T R_k \boldsymbol{a}_j \approx \alpha^2 \frac{\boldsymbol{a}_i^T \boldsymbol{a}_j}{\|\boldsymbol{a}_i\| \|\boldsymbol{a}_j\|}$, such that the probability of a relationship x_{ijk} depends on the cosine similarity of $\boldsymbol{a}_i$ and $\boldsymbol{a}_j$. Furthermore, the magnitude of the diagonal entries in R_k can model the importance of a latent component for the homophily pattern in a particular relation. For instance, consider the following interaction matrix R_k and entity representations $\boldsymbol{a}_1$, $\boldsymbol{a}_2$, $\boldsymbol{a}_3$:

$$R_k = \begin{bmatrix} 0.8 & 0.1 \\ 0.1 & 0.9 \end{bmatrix}, \boldsymbol{a}_1 = \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix}, \boldsymbol{a}_2 = \begin{bmatrix} 0.2 \\ 0.8 \end{bmatrix}, \boldsymbol{a}_3 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

As the latent representations of $\texttt{entity}_1$ and $\texttt{entity}_3$ are more similar then the representations of $\texttt{entity}_1$ and $\texttt{entity}_2$, it is more likely that a relationship exists between the former than between the latter pair of entities.

In general, the entries in the latent components of A and $\mathbf{R}$ are not confined to this interval $[0, 1]$ but are allowed to take negative values as well as values larger than one. Although this complicates the *interpretation* of the latent representations this does not alter the general ability of RESCAL to model such patterns.

2.3. Solving Relational Learning Tasks

Given the factorization of an adjacency tensor, RESCAL can be used to approach all relational learning tasks outlined in section 1.2.2 as follows:

LINK PREDICTION

For link prediction, the task is to predict $\mathrm{P}\left(x_{ijk} = 1\right)$. As discussed in section 2.2, under a least-squares loss function, the entries in the reconstruction $\widehat{\mathbf{X}} = \mathbf{R} \times_1 A \times_2 A$ are not confined to the interval $[0, 1]$ and do not have an easy interpretation as probabilities. However, usually we are not interested in exact probabilities, but only in a *ranking* of relationships relative to their likelihood. In such cases, we interpret the entries of $\widehat{\mathbf{X}}$ as *confidence values* that

a particular relationship exists, meaning that the probability of a relationship is set to be proportional to the corresponding entry in $\widehat{\mathbf{X}}$, i.e. $\mathrm{P}\left(x_{ijk} = 1 \,\middle|\, \boldsymbol{a}_i^T R_k \boldsymbol{a}_j\right) \propto \boldsymbol{a}_i^T R_k \boldsymbol{a}_j$. Given a reconstruction $\widehat{\mathbf{X}}$, we then simply rank relationships by these confidence values. A similar approach has successfully been used by Liben-Nowell and J. Kleinberg (2007) for link prediction on social networks via SVD, as well as by Huang et al. (2011) for relational learning via matrix factorization. If it is necessary to obtain valid probabilities, we apply an additional post-processing step, which has been introduced by Platt (1999). Let $\mathrm{sig}_\varepsilon : \mathbb{R} \mapsto [0,1]$ be the sigmoidal transfer function, which is defined as

$$\mathrm{sig}_\varepsilon(x) := \begin{cases} \frac{\varepsilon}{e} \exp\left(\frac{x}{\varepsilon}\right), & \text{if } x \leq \varepsilon \\ x, & \text{if } \varepsilon < x < 1-\varepsilon \\ 1 - \frac{\varepsilon}{e} \exp\left(\frac{1-x}{\varepsilon}\right), & \text{if } x \geq 1-\varepsilon \end{cases}$$

where $\varepsilon \in [0, 0.5]$ is a user-given parameter and e denotes Euler's number. The probability of a relationship x_{ijk} is then set to

$$\mathrm{P}\left(x_{ijk} = 1 \,\middle|\, \boldsymbol{a}_i^T R_k \boldsymbol{a}_j\right) = \mathrm{sig}_\varepsilon\left(\boldsymbol{a}_i^T R_k \boldsymbol{a}_j\right)$$

Please note the $\mathrm{sig}_\varepsilon(\cdot)$ is a monotone function, such that the relative ordering of entries $\boldsymbol{a}_i^T R_k \boldsymbol{a}_j$ is preserved. The parameter ε is determined via cross-validation.

Entity Resolution and Link-Based Clustering

Entity resolution and link-based clustering are both learning tasks which are defined over the similarity of entities in a relational domain, meaning that similar entities are assumed to be identical (entity resolution) or that similar entities are grouped in identical clusters (link-based clustering). To approach these learning tasks, or any other task that is defined over the relational similarity of entities, we make use of the fact that the latent space A reflects the similarity of entities in the relational domain, as discussed in section 2.2. Since A is a *vector space* representation of entities, any feature-based machine learning method such as k-means or even non-linear kernel methods can be applied to these tasks and still exploit the similarity of the entities in the relational domain.

Collective Classification

Collective classification can be approached in two alternative ways. One way, is to cast collective classification as a link prediction problem, by introducing an additional `classOf`

relation and including the classes as entities in the tensor representation, as described in section 1.2.2. Then, the classification problem can also solved by computing the appropriate entries $\widehat{x}_{ijk} = \boldsymbol{a}_i^T R_k \boldsymbol{a}_j$ for the `classOf` slice. The collectivity of the classification, as for all other learning tasks, is given automatically by the structure of the model. The second way, exploits the latent similarity of entities. As the latent space A is a vector space representation of the entities that reflects their similarity in the relational domain, any feature-based classification algorithm such as Support Vector Machines (SVMs) can be applied to this task to collectively classify the entities in a data set.

2.4. Discussion and Related Work

In this section, we will put the RESCAL model into the context of related work in relational learning and tensor factorizations.

2.4.1. Comparison to Statistical Relational Learning Models

State-of-the art SRL methods have been reviewed in section 1.2.3. Compared to models such as PRM and MLN, which employ probabilistic graphical models on *observable* variables, RESCAL features two key advantages. First, the prediction of unknown relationships in RESCAL is very fast. As mentioned in section 2.3, the probability of a random variable x_{ijk} in RESCAL is conditionally independent of all other variables x_{pqr}, given the latent factors, i.e. it holds that $\mathrm{P}\left(x_{ijk} = 1 \,\middle|\, \boldsymbol{a}_i, \boldsymbol{a}_j, R_k\right) \propto \boldsymbol{a}_i^T R_k \boldsymbol{a}_j$. Consequently, it is sufficient to compute a simple vector-matrix-vector product to determine the confidence value for a particular relationship, what can be computed very efficiently. Second, being a latent variable model like IHRM, no structure learning is required for the functioning of RESCAL, as the dependency structure is already defined within the factorization. However, while IHRM is an *infinite* latent *class* model, where each entity is assigned to exactly one out of a possibly infinite number of binary latent variables, RESCAL is a *finite* latent *feature* model, where entities are represented via a predetermined number of continuous variables. Moreover, IHRM is a fully Bayesian approach that estimates the distribution of latent variables, while RESCAL computes the MAP point estimate of these variables. It will be shown in section 2.5 and chapter 3 that these differences allow for a very efficient method to compute the RESCAL factorization. Furthermore, a conceptual advantage of latent feature models over a latent class models is that the latter do not allow for an analysis of entities via their latent representations as discussed in section 2.2. In latent class models, exactly one latent variable is set to one for each entity,

while all its other variables are set to zero, such that the application of feature-based machine learning algorithms would not be meaningful. Typically, a latent feature model is also more economical in its latent representations, meaning that fewer latent components are required to describe a data set.

2.4.2. Comparison to Tensor Models

It has been outlined in section 2.2 that RESCAL can be regarded a TUCKER-2 model, with the additional constraint that the latent factor matrices have to be identical. We reasoned in section 2.2 why these constraints are an important factor for relational learning. Compared to unconstrained TUCKER models, it can be argued that a key advantage of RESCAL is that its the model does not loose information about the structure of the data as it takes the identity of entities into account. The Bayesian clustered tensor factorization (BCTF) considered by Sutskever et al. (2009) is an approach to relational learning that is based on such an unconstrained TUCKER model. Huang et al. (2011) an approach to multi-relational learning based on matrix factorization called Statistical Unit Node Sets (SUNS). This approach is related to a TUCKER-1 model, where an adjacency tensor $\mathbf{X}$ is unfolded in the first mode and then factorized. However, unlike in a simple TUCKER-1 model, SUNS confines the first mode to a particular statistical unit in the data and also employs regularization by default. However, TUCKER-1 models, on which SUNS is based, even loose information about the distinction between predicates and objects, such that they can also not be expected to show good collective learning results.

CP is a popular tensor factorization – partly due to its uniqueness properties – which has also been employed for learning from relational data (Kolda et al., 2005; Franz et al., 2009). However, CP can not express RESCAL-type constraints and simultaneously model asymmetric relations. By enforcing a unique representation of entities via a single latent factor A, the CP model would become

$$\mathbf{X} \approx \sum_{i=1}^{r} \boldsymbol{a}_i \otimes \boldsymbol{a}_i \otimes \boldsymbol{b}_i.$$

Since the outer product of identical vectors $\boldsymbol{a}_i \otimes \boldsymbol{a}_i$ is symmetric, each frontal slice of $\mathbf{X}$ would be modeled as a symmetric relation. Unlike CP, the RESCAL factorization is not unique, as it holds for any unitary matrix $Q \in \mathbb{R}^{r \times r}$ that

$$AR_kA^T = AQQ^TR_kQQ^TA^T.$$

In RESCAL, the latent factor A could therefore be freely transformed by a unitary matrix

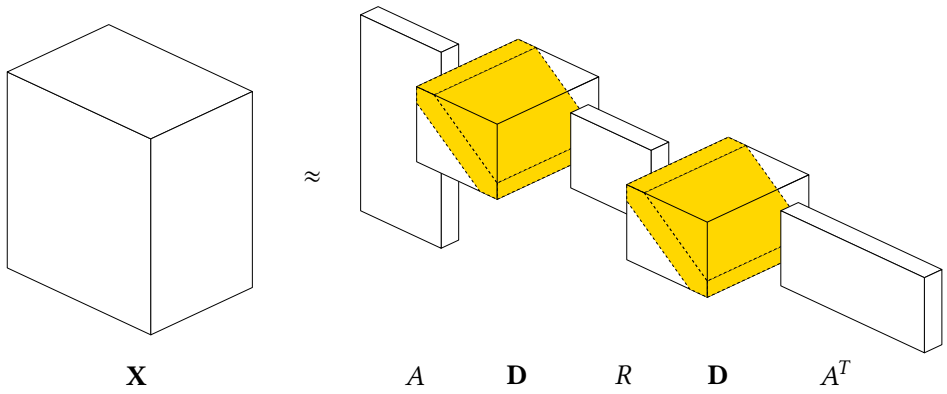


Figure 2.5.: The Dedicom factorization. Each frontal slice D_k of $\mathbf{D}$ is a diagonal matrix. Please note that the DEDICOM model can not be expressed via standard tensor operations, but only as a joint factorization of the frontal slices of $\mathbf{X}$.

Q, as long as the inverse transformation Q^T is applied simultaneously to the core tensor. Uniqueness is a desirable property when the latent components should be interpretable, e.g. to analyze physical components in chemometrics. However, for the tasks considered in this thesis, we do not seek to interpret the latent representations directly, such that uniqueness of the factorization is not needed.

At last, a tensor factorization that is closely related to RESCAL is the decomposition into directional components (DEDICOM) by Harshman (1978). For uni-relational data, i.e. data in form of an adjacency *matrix* X, the RESCAL and DEDICOM models are identical, since both models compute the factorization $X \approx ARA^T$. However, this changes for data with multiple relations, i.e. when an adjacency *tensor* is factorized. In this case, DEDICOM factorizes a third-order tensor $\mathbf{X} \in \mathbb{R}^{n \times n \times m}$ such that each frontal slice X_k is approximated by

$$X_k \approx AD_kRD_kA^T$$

where $A \in \mathbb{R}^{n \times r}$, $R \in \mathbb{R}^{r \times r}$, and where $D_k \in \mathbb{R}^{r \times r}$ is a diagonal matrix. Figure 2.5 shows a visualization of the DEDICOM factorization. Similar to RESCAL, DEDICOM factorizes $\mathbf{X}$, such that two modes of $\mathbf{X}$ are explained by identical latent factors A. However, in contrast to RESCAL, DEDICOM requires also that *all* frontal slices X_k are explained by a single matrix R, which encodes the global interactions of the latent component for all slices X_k. Furthermore, the variation in these global interaction patterns over different slices X_k is only expressed by *diagonal* matrices D_k. This are obviously much stronger constraints than in the RESCAL model, for which the interaction patterns of each slice X_k are expressed by a separate, full, asymmetric matrix R_k. The constraints of DEDICOM offer great value for its original purpose, i.e. the analysis

of a single relation that varies over time or over different populations (Harshman, 1978; Bader et al., 2007). However, these constraints are hard to justify for general multi-relational data, as it is not reasonable to assume that different relations always follow a global interaction pattern that changes only by diagonal factors D_k. Furthermore, the DEDICOM model is only applicable to third-order tensors, such that it is confined to binary relations. Although it is not pursued in this thesis, conceptually, the RESCAL model can be easily extended to higher-order tensors such that n-ary relations can be modeled.

2.5. Computing the Factorization

Recall from section 2.2 that to compute the RESCAL factorization we seek the solution to the optimization problem

$$\arg\min_{A,\mathbf{R}} \|\mathbf{X} - \mathbf{R} \times_1 A \times_2 A\|^2 + \lambda_A \|A\|^2 + \lambda_{\mathbf{R}} \|\mathbf{R}\|^2$$

The dimensionality of the latent space $r \in \{r \in \mathbb{N} \mid 0 < r < n\}$, as well as the regularization parameters $\lambda_A \geq 0$ and $\lambda_{\mathbf{R}} \geq 0$ are assumed to be known. In practice, we will use cross-validation to find the optimal settings for these parameters. Local optima to the optimization problem equation (2.5) can be computed by taking the partial derivatives of its objective function with respect to A and $\mathbf{R}$ and using any gradient-based non-linear optimization algorithm. However, to improve the computational efficiency of the algorithm, we exploit the connections between RESCAL and DEDICOM and adapt techniques from the very efficient Alternating Simultaneous Approximation, Least Squares and Newton (ASALSAN) algorithm by Bader et al. (2007) to compute the RESCAL model.

ASALSAN is based on the alternating least-squares (ALS) method, which is also the standard method to compute tensor factorizations such as CP or TUCKER for a least-squares loss function (Acar and Yener, 2009; Kolda and Bader, 2009). Being a block coordinate descent (BCD) method, in ALS, the variables of an optimization problem are partitioned into disjoint blocks, such that the objective function is optimized via alternating updates of these variable blocks until a certain convergence criterion is met (Sra et al., 2012). For RESCAL, the optimization variables are already partitioned naturally via the core tensor and the factor matrices, such that we alternatingly keep one of the factors A and $\mathbf{R}$ fixed and compute the update for the remaining factor. In each of these update steps, we will use the method of normal equations to compute *closed-form* solutions for the optimization subproblems. In this context, regularization serves also another purpose, as it increases the numerical stability

of normal-equation-based algorithms (Neumaier, 1998). In the following, we will refer to this algorithm as RESCAL-ALS The updates for the latent factors A and $\mathbf{R}$ in RESCAL-ALS are computed as follows:

Updates for A To compute updates for the factor matrix A, we keep $\mathbf{R}$ fixed and seek the solution to the optimization problem

$$\arg\min_{A} \|\mathbf{X} - \mathbf{R} \times_1 A \times_2 A\|^2 + \lambda_A \|A\|^2. \tag{2.8}$$

Since the variable A appears twice in equation (2.8), this optimization problem can not be reduced to linear regression anymore and it would be difficult to derive scalable closed-form updates for it. For this reason, we use a similar approach as taken in ASALSAN and use an approximate procedure, which is based on the idea to consider the unfolding of the first and second mode of $\mathbf{X}$ simultaneously. To motivate this approach, we rewrite equation (2.8) as the constrained optimization problem

$$\begin{aligned} &\arg\min_{A} \|\mathbf{X} - \mathbf{R} \times_1 A_\ell \times_2 A_r\|^2 + \lambda_A \|A_\ell\|^2 \\ &\text{subject to } A_\ell = A_r \end{aligned} \tag{2.9}$$

where A_ℓ corresponds to the left-hand A in equation (2.8) and A_r to the right-hand A. It can be seen from the equivalence

$$\begin{aligned} \|\mathbf{X} - \mathbf{R} \times_1 A_\ell \times_2 A_r\|^2 &= \|X_{(1)} - A_\ell R_{(1)} (I \otimes A_r)^T\|^2 \\ &= \|X_{(2)} - A_r R_{(2)} (I \otimes A_\ell)^T\|^2 \end{aligned}$$

that A_ℓ appears as the left-hand factor for the unfolding of the first mode, while A_r appears the left-hand factor for the unfolding of the second mode. Furthermore, it holds for both tensors $\mathbf{X}$ and $\mathbf{R}$ that

$$X_{(1)} = \begin{bmatrix} X_1 & \cdots & X_m \end{bmatrix} \qquad R_{(1)} = \begin{bmatrix} R_1 & \cdots & R_m \end{bmatrix}$$
$$X_{(2)} = \begin{bmatrix} X_1^T & \cdots & X_m^T \end{bmatrix} \qquad R_{(2)} = \begin{bmatrix} R_1^T & \cdots & R_m^T \end{bmatrix}$$

since they are third-order tensors with square frontal slices. Now, to approximate equation (2.9), the idea is to stack $X_{(1)}$ and $X_{(2)}$ as well as $R_{(1)}$ and $R_{(2)}$ side by side and to solve only for the left-hand factor, while keeping the right-hand factor constant. This way, the information of both unfoldings is included in an update of A, but the optimization problem

can be reduced to simple linear regression. Therefore, to compute an update of A, we set

$$\underline{X} = \begin{bmatrix} X_1 & X_1^T & \cdots & X_m & X_m^T \end{bmatrix}$$
$$\underline{R} = \begin{bmatrix} R_1 & R_1^T & \cdots & R_m & R_m^T \end{bmatrix}$$

Furthermore, let $M = \underline{R}(I_{2m} \otimes \underline{A})^T$ and $\underline{A}$ be constant. Then, we approximate equation (2.8), by computing the solution to

$$\arg\min_A \|\underline{X} - AM\|^2 + \lambda_A \|A\|^2 \tag{2.10}$$

The advantage of the optimization problem equation (2.10) over equation (2.8) is that it is in the form of a Tikhonov regularization problem which has a closed-form solution, i.e. (Boyd and Vandenberghe, 2004, Section 6.3.2)

$$A = \underline{X}M^T(MM^T + \lambda_A I)^{-1}$$

However, computing MM^T directly is not practical – other than for very small data sets – since M is a dense matrix of size $r \times 2mn$. Fortunately, both terms $\underline{X}M^T$ and MM^T can be reduced significantly using properties of the Kronecker product and block partitioned matrices, such that an update for A can be computed by

$$A \leftarrow \left(\sum_{k=1}^{m} X_k A R_k^T + X_k^T A R_k\right)\left(\sum_{k=1}^{m} B_k + C_k + \lambda_A I\right)^{-1} \tag{2.11}$$

where

$$B_k = R_k A^T A R_k^T, \quad C_k = R_k^T A^T A R_k$$

The derivation of equation (2.11) from properties of the Kronecker product and block partitioned matrices is given in detail in appendix A.2.

Updates for R To update the core tensor $\mathbf{R}$, we keep the matrix A fixed and seek the solution to

$$\arg\min_{\mathbf{R}} \|\mathbf{X} - \mathbf{R} \times_1 A \times_2 A\|^2 + \lambda_{\mathbf{R}} \|\mathbf{R}\|^2. \tag{2.12}$$

Due to the Tucker-2 structure of Rescal, equation (2.12) can be written equivalently in matrix notation as

$$\arg\min_{A,\mathbf{R}} \sum_{k=1}^{m} \|X_k - A R_k A^T\|^2 + \lambda_{\mathbf{R}} \|R_k\|^2, \tag{2.13}$$

where $A \in \mathbb{R}^{n \times r}$ and $R_k \in \mathbb{R}^{r \times r}$. It can be seen from equation (2.13), that the TUCKER-2 structure of RESCAL, renders updates of the frontal slices independent of each other, such that each slice R_k can be updated separately. Furthermore, by vectorizing X_k as well as AR_kA^T we can use equation (1.12) and rewrite the minimization problem equation (2.13) as

$$\underset{\text{vec}(R_k)}{\arg\min} \left\| \text{vec}\left(X_k\right) - \left(A \otimes A\right) \text{vec}\left(R_k\right) \right\|^2 + \lambda_{\mathbf{R}} \left\| \text{vec}\left(R_k\right) \right\|^2 . \tag{2.14}$$

Equation (2.14) is now again a Tikhonov regularization problem and can be solved accordingly, i.e. by computing

$$\text{vec}\left(R_k\right) \leftarrow \left(Z^T Z + \lambda_{\mathbf{R}} I\right)^{-1} Z^T \text{vec}\left(X_k\right) \tag{2.15}$$

where $Z = A \otimes A$. However, computing Z directly is again not practical, as it is a dense $n^2 \times r^2$ matrix. Instead, we apply equations (1.9) and (1.11) such that $Z^T Z$ can be computed by $A^T A \otimes A^T A$ without ever computing Z explicitly. Similarly, $Z \text{ vec}\left(X_k\right)$ can be computed via $\text{vec}\left(A^T X_k A\right)$, which is significantly faster to compute (Golub and Loan, 2013, Section 1.3.7). Furthermore, for non-regularized problems, i.e. where $\lambda_{\mathbf{R}} = 0$, we can apply equation (1.10), such that it is only necessary to compute the matrix inversion of $r \times r$ matrices $A^T A$ instead of $r^2 \times r^2$ matrices $A^T A \otimes A^T A$.

Convergence and Initialization These update steps for A and R_k are iterated alternatingly until the algorithms reaches convergence or a maximum number of iterations are exceeded. As a convergence criterion, we employ the relative change of the objective function in equation (2.5) compared to a user-specified threshold ε. The starting points for A and R_k can either be set to random matrices or can be initialized as in DEDICOM, i.e. by setting A to the r largest eigenvectors of the eigendecomposition of $\sum_k (X_k + X_k^T)$. In most cases we will employ this eigenvector based initialization, as we found it experimentally to be far more reliable than the random initialization.

2.6. Experiments

In the following, we evaluate the performance of the RESCAL factorization for several relational learning tasks. All experiments have been evaluated on a personal computer with an Intel Core 2 Duo 2.5 GHz CPU and 4 GB RAM. Table 2.2 lists the algorithms that have been used to compute other tensor decompositions in these experiments. For MLN, we used the current "`Aug 23, 2010`" release of the ALCHEMY toolbox (Kok et al., 2005). Due to the large skew in

Table 2.2.: Tensor decomposition algorithms used in experiments

Decomposition	Algorithm
CP	CP-ALS (Kolda and Bader, 2009, Fig. 3.3)
Tucker	Higher-Order Orthogonal Iterations (HOOI) (De Lathauwer et al., 2000b)
Dedicom	Asalsan (Bader et al., 2007)

the distribution of existing and non-existing relationships, we will evaluate the results via the area under the precision-recall curve (AUC-PR), which has been shown to be a suitable measure when the number of negative examples exceeds the number of positive examples significantly (Davis and Goadrich, 2006).

2.6.1. Collective Learning

The first experiment in the evaluation of Rescal has been designed to asses its collective learning capabilities, i.e. its ability to learn from information that might be more distant in the relational graph. To do this in a controlled setting, where any influence from non-relational information is excluded, we created a data set specifically for this experiment: First, we retrieved all presidents and vice-presidents of the United States of America from DBpedia, along with all political parties, in which these persons were members. These objects, persons and parties, formed the set of entities in our data set. Furthermore, we also retrieved all relationships between these entities for the relations `presidentOf` , `vicePresidentOf` , and `partyOf` from DBpedia. After the collection of the data, we removed duplicate entities and corrected erroneous relationships manually. In total, the cleaned data set consisted of 78 persons, 10 parties, 49 `presidentOf` and `vicePresidentOf` relationships, and 83 `partyOf` relationships. The objective was then to predict the party memberships for all presidents and vice presidents in the data, i.e. to predict relationships of the form $\texttt{partyOf}(\texttt{person}_i, \texttt{party}_j)$. Such a link prediction setting can often benefit from the exploitation of relational patterns such as homophily and stochastic equivalence and is therefore a good measure for the collective learning capabilities of a relational model.

It can be seen from figure 2.6 that a president and *his* vice-president have mostly been members of the same party. This relational pattern should enable relational models with collective learning capabilities to predict the correct party when it is unknown. The patterns in the data can be regarded a combination of stochastic equivalence (persons link to parties) and homophily (related persons are members of the same party). It can also be seen that

party memberships are very unevenly distributed overall parties, such that some parties are far more likely to occur as an object in a `partyOf` relationship than others. To evaluate the performance of RESCAL compared to standard tensor factorizations and to determine the effect of a unique representation of entities, we included the following models in the experiment:

Baseline As a baseline method we ranked all possible party memberships for a person *randomly*. Since the distribution of party memberships is very uneven across parties, we also ranked the parties by their *frequency*, what should give significantly better results than a random ranking.

Factorization Methods To compare RESCAL to related factorization methods, we included all variants of the TUCKER family, i.e. TUCKER-1, TUCKER-2, and TUCKER-3. Since the eigenvector-based initialization of RESCAL is different to standard TUCKER initialization (see section 2.5), we also included a TUCKER-2 model which has been initialized like RESCAL, to ensure that this difference in initialization is *not* the cause for diverging results of RESCAL and TUCKER-2. We also included SUNS and CP in the evaluation, which have been used for large-scale relational learning (Huang et al., 2011; Huang et al., 2013) and information retrieval on multigraphs (Kolda et al., 2005; Franz et al., 2009) respectively. At last, we included DEDICOM, which is the only factorization model apart from RESCAL that enforces a unique representation of entities.

Markov Logic Networks To compare RESCAL against state-of-the-art SRL methods, we included MLNs in three variants: MLN with structure learning and MLN with two different sets of manually defined rules. For MLN with structure learning, in the following denoted by *MLN(Struct)*, we used the `learnstruct` command of the Alchemy toolkit. Detailed parameter settings and the formulas that haven been learned are listed in appendix B. For MLN with manually defined rules, we modeled the relational patterns in the data set in form of logical formulas. The first set of rules, in the following denoted by *MLN(H)*, modeled the global homophily pattern that a person and his related person are usually members of the same party, i.e.

$$\texttt{partyOf}(x,y) \wedge \texttt{presidentOf}(x,z) \Rightarrow \texttt{partyOf}(z,y)$$
$$\texttt{partyOf}(x,y) \wedge \texttt{vicePresidentOf}(x,z) \Rightarrow \texttt{partyOf}(z,y)$$

The second set of rules, in the following denoted by *MLN(HF)*, extended the rule set

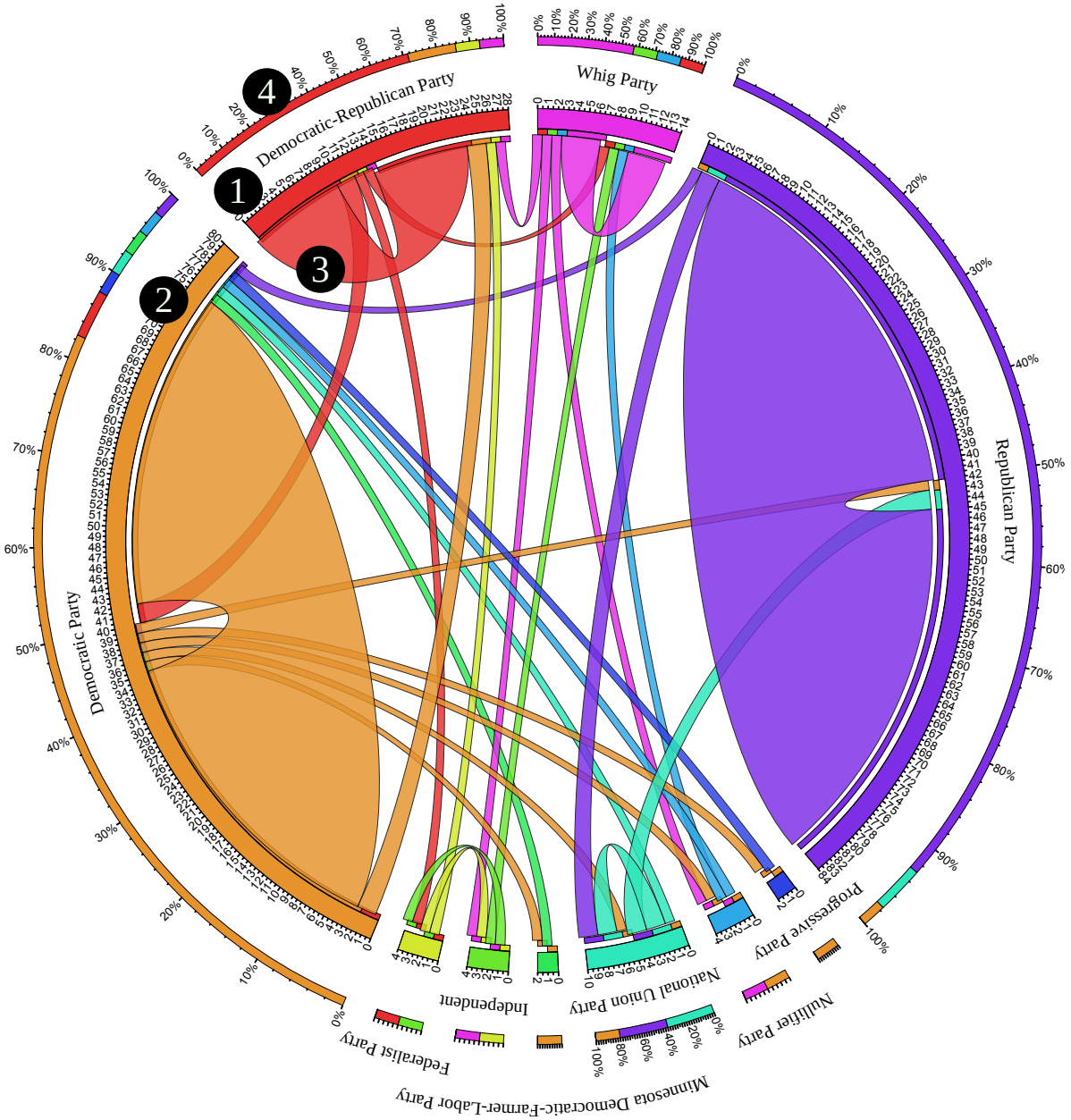


Figure 2.6.: Visualization of the U.S. Presidents data. Each tick in the inner circle (1) indicates a tuple (*president's party*, *his vice president's party*). For persons with more than one party affiliation we included a separate tuple for each combination of party memberships. Each inner ring segment (2) represents a party, while the size of an inner segment indicates for how many tuples the president has been a member of the respective party. The size of an arc (3) indicates how often a `presidentOf` relationship existed between members of the connected party. The color of an arc indicates the president's party. The colored subsegments in an outer ring segment (4) visualize the percentage of how many presidents of the respective party have vice presidents of the party that is associated with the subsegments color. It can been seen that a president and his vice president have been members of the same party for more than 90% of the relationships for the Republican party, more than 80% of the relationships for the Democratic party, more than 70% of the relationships for the Democratic-Republican party and more than 50% of the relationships for the Whig party.

MLN(H) with multiple formulas such as

$$\texttt{partyOf}(x,\texttt{DemocraticParty})$$
$$\texttt{partyOf}(x,\texttt{RepublicanParty})$$
$$\vdots$$
$$\texttt{partyOf}(x,\texttt{WhigParty})$$

for each individual party in the data set. These formulas are the Markov Logic Network equivalent to the *frequency* baseline, as they learn how likely it is that a particular party occurs as the object of a `partyOf` relation.

To evaluate these learning methods, we performed ten-fold cross-validation over all persons, where we removed *all* `partyOf` relationships for a person in the test fold, such that the party membership of a person has to be inferred from his/her related persons. In this setting, collective learning methods should have a significant advantage, as the data set does not contain any non-relational information that could help for this task. To facilitate this prediction of party memberships from relational information, we ensured that for each person, at least one related person was assigned to a different fold during the creation of the random test/train splits. For RESCAL, we computed a factorization with $r = 5$ latent components, ranked all parties by their predicted values in the `partyOf` relation for all persons in the test fold and recorded the area under the precision-recall curve. For all other models in the evaluation we employed an equivalent procedure. All parameters were determined via cross-validation. Figure 2.7 shows the results of this evaluation. It can be clearly seen that RESCAL is able to exploit the relational information in the data very efficiently, as it nearly matches the best MLN with manually optimized rules. Furthermore, it can be seen that this ability is due to the unique representation of entities. All bipartite factorization models show significantly worse results and are not even matching the *frequency* baseline method. The DEDICOM factorization shows significantly better performance, what indicates, in combination with the results of RESCAL, the importance of a unipartite modeling for collective learning. However, DEDICOM is not able to match the results of RESCAL, what also demonstrates that the constraints on the core tensor of DEDICOM are not appropriate for relational data. The comparison of RESCAL to MLNs is also very favorable: RESCAL surpasses the results of MLN(HF) and nearly matches the results of the best manual rule set MLN(H), what shows its state-of-the-art performance. Counterintuitively, MLN(HF) performs worse than MLN(H), what points to the difficulty of choosing the best rule set in MLNs manually. Moreover, since RESCAL does not incorporate

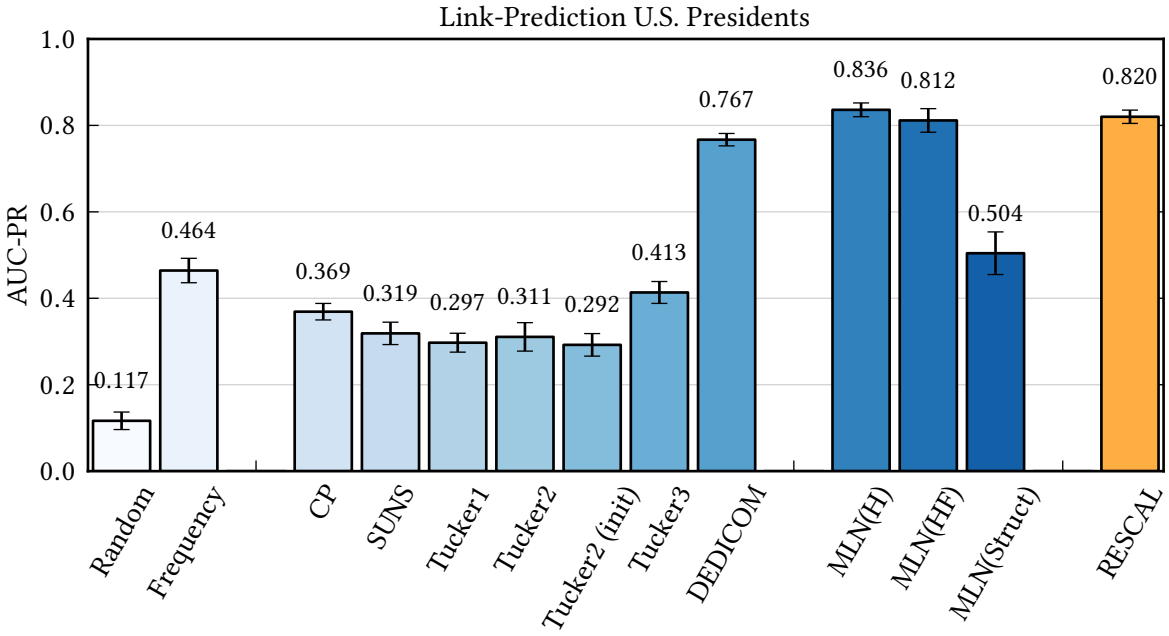


Figure 2.7.: Link prediction results on U.S presidents data set

any prior knowledge about a domain, it can be argued that MLN with structure learning is the more appropriate comparison. In this setting, RESCAL clearly outperforms MLN, what is especially important for real-world applications where the optimal set of rules is unknown.

2.6.2. Entity Resolution

The U.S. Presidents data set demonstrated the collective learning capabilities of RESCAL on a relatively small and specifically created data set. The Cora data set is a larger real-world data set, consisting of a citation network in the computer science field and is commonly used to assess the performance of relational learning methods for tasks like collective classification and entity resolution. Moreover, it is has been shown in Sen et al. (2008) that collective learning can improve the quality of a model significantly on this data set. Different versions of the data have been used in the literature. Here, we use the data set, the experimental setup, and the train-test-splits as described in Singla and Domingos (2006a).[1] In this version, the data set consists of 1295 publication records, where each publication is the subject of a relationship to its first author, a relationship to its title, and a relationship to its publication venue. A visualization of the relations between different types of entities in this data set is shown in figure 2.8 The data set is generally very noisy, containing duplicate entries for

[1]The Cora data set together with fixed train-test-splits is available from `http://alchemy.cs.washington.edu/data/cora/cora.er.tgz`.

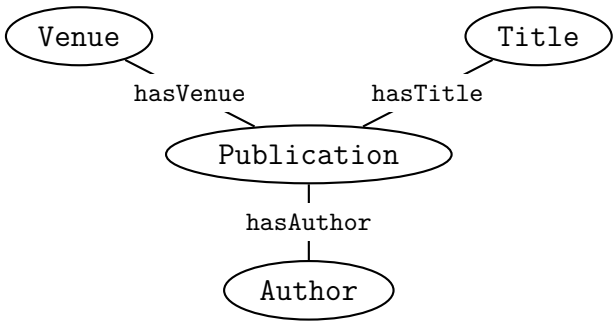


Figure 2.8.: Relations between types of entities in the Cora data set.

publications, authors, titles and venues. The task was then to perform entity resolution on this data set, i.e. to identify which authors, entities and venues refer to identical entities. The ground truth, consisting of 132 unique publications, 50 authors and 103 venues, has been manually compiled by Singla and Domingos (2006a).

To perform entity resolution with RESCAL, we computed a factorization of the $2725 \times 2725 \times 8$ adjacency tensor with $r = 100$ latent components. To create a ranking of the most similar entities, we exploited the property that the similarity in the latent space A reflects the relational similarity of entities. To be more precise, we computed the similarity of entities using the heat kernel

$$k(x, y) = \exp\left(-\frac{\|x - y\|^2}{\delta}\right), \tag{2.16}$$

where δ is a user-given constant. Then, we used this similarity score as a measure for the likelihood that the entities x and y refer to the same underlying entity. Before applying the heat kernel to A, we also normalized each of its rows by their norms, i.e. we set $\boldsymbol{a}_i \leftarrow \frac{\boldsymbol{a}_i}{\|\boldsymbol{a}_i\|}$. This procedure is due to the fact that the magnitude of entries in A reflects the general link probability of an entity, such that the magnitude for entries of uncommon duplicates would be very low compared to duplicates that occur more commonly. Since equation (2.16) considers also the magnitude of entries, this effect would disturb the resolution process, such that we remove it by normalizing each row. Table 2.3 shows the results of RESCAL for five-fold cross-validation compared to the non-relational Naive Bayes algorithm, a Markov Logic Network with very basic rules (B), a Markov Logic Network with sophisticated rules (BCTS), and the CP tensor factorization. Detailed descriptions for Naive Bayes, MLN (B), and MLN (BCTS) can be found in Singla and Domingos (2006a). It can be seen that RESCAL gives generally very good results, surpassing the best MLN model on all entity types. Furthermore, the evaluation results are also very interesting in combination with the structural representation

Table 2.3.: Area under the precision-recall curve for entity resolution on the Cora data set with five-fold cross validation.

	AUC-PR				
Entity Type	Naive Bayes	MLN (B)	MLN (BCTS)	CP	RESCAL
Publications	0.913	0.915	0.988	0.991	0.991
Authors	0.986	0.987	0.992	0.984	0.997
Venue	0.738	0.736	0.807	0.746	0.810

of the data shown in figure 2.8. Since `Publication` is the central class in figure 2.8 and since there exist no relations between `Author` , `Venue` , and `Title` , it is sufficient for the deduplication of citations, to consider only directly connected entities. However, the case is different for the deduplication of authors and venues. Since there are many duplicate publications, it is very helpful for a learning method if it can look past a publication and include the information about related venues, titles, and authors in the resolution process. While CP shows state-of-the-art results for the deduplication of publications, i.e. the case where collective learning is not needed, it surpasses the non-relational Naive Bayes approach only slightly for the deduplication of venues and gives even worse results for authors. RESCAL on the other hand shows strong results in all three experiments. This underlines again the collective learning capabilities of RESCAL and the lack thereof for CP.

2.6.3. Link Prediction on Relational Learning Benchmark Datasets

To evaluate how well RESCAL performs for link prediction compared to current state-of-the-art relational learning methods, we applied it to multiple benchmark data sets introduced by Kemp et al. (2006) and compared it to the results of BCTF, IRM, and MRC published in Sutskever et al. (2009) and Kok and Domingos (2007). These data sets are the following:[1]

Kinships The *Kinships* data set is based on data compiled by Denham (1973) and consists of multiple kinship relations between members of the Alyawarra tribe in central Australia. Among anthropologists, Australian tribes are known for their complex kinship systems such that predicting individual relationships within these kinship systems an interesting benchmark for relational learning methods. In total, the Kinships data consists of 10,790 kinship relationships between 104 persons over 26 relations.

UMLS The *UMLS* data set consists of a small semantic network which is part of the Unified

[1] All data sets are available from `http://alchemy.cs.washington.edu/data/`

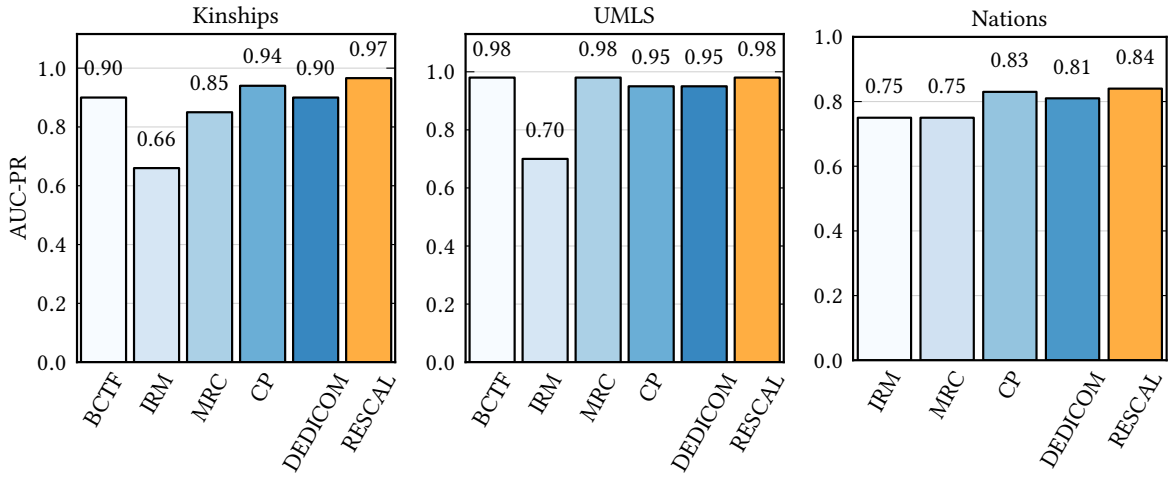


Figure 2.9.: Link prediction on benchmark data sets

Medical Language System (UMLS) ontology. UMLS itself is a large biomedical ontology which provides a controlled vocabulary over multiple biomedical databases and dictionaries. The semantic network used in the experiments describes relationships between concepts such as `Bacterium` , `Virus` and `Clinical Drug` over multiple relations such as `causes` , `diagnoses` or `consistsOf` . In total, there exist 6,752 relationships between 135 concepts over 49 relations.

Nations The *Nations* data set describes political interactions of countries between 1950 and 1965 (Rummel, 1999). It contains information such as military alliances, trade relationships or whether a country maintains an embassy in a particular country. The data set contains 2,024 relationships between 14 countries over 56 dyadic relations. The Nations data set also contains information about the attributes of countries in form of unary relations. In particular, continuous variables have been binarized by thresholding values at their mean and using one-out-of-n coding for categorical attributes (Kemp et al., 2006). Using this procedure, 111 different binary attribute values and 541 unary relationships between countries and these attribute values have been created. In our experiments, we introduced these attribute values as new entities and modeled attribute information as entity-entity relationships.

To get comparable results to BCTF, IRM and MRC, we followed the experimental protocol of Kok and Domingos (2007) and performed ten-fold cross-validation over randomly selected relationships. For a comparison to standard tensor factorizations we also included CP and DEDICOM in the evaluation. Figure 2.9 shows the results of our evaluation. It can be seen

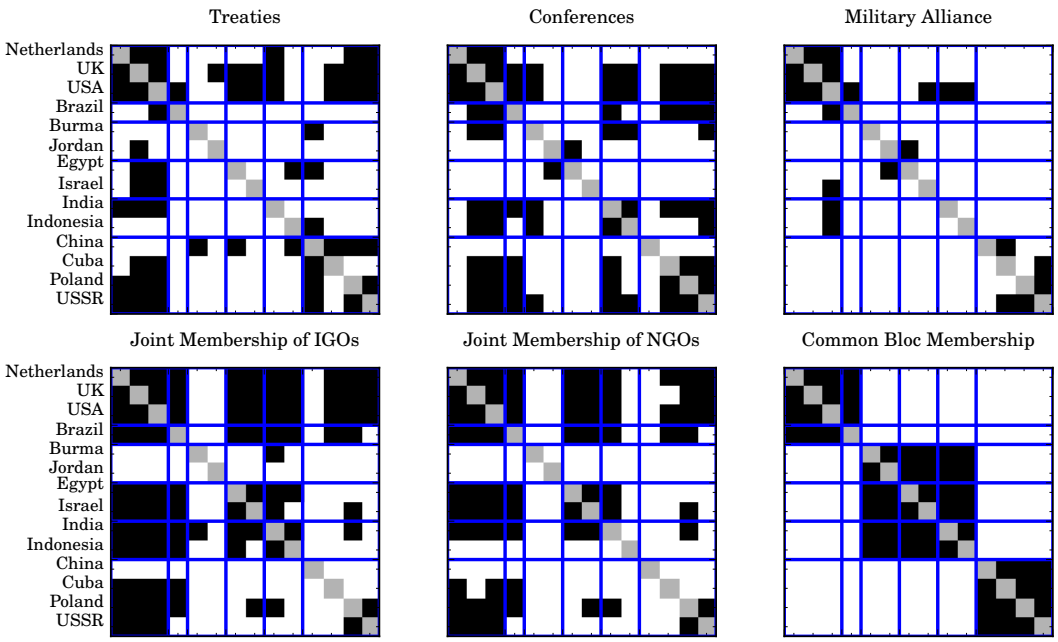


Figure 2.10.: A clustering of countries in the Nations data set. Black squares indicate an existing relation between the countries. Gray squares indicate missing values.

that RESCAL provides comparable or better results than BCTF, IRM, and MRC on all of these benchmark data sets.

2.6.4. Link-Based Clustering

To briefly demonstrate the link-based clustering capabilities of RESCAL, we computed a rank-20 decomposition of the Nations data set and applied k-means with $k = 6$ to the latent space A. It can be seen from figure 2.10 that in doing so, similar clusters as in Kemp et al. (2006) are obtained. The countries are partitioned into one group containing countries from the communist block, two groups from the western block, where Brazil is separated from the rest, and three groups for the neutral block. The six relations shown in Figure 2.10 indicate that this is a reasonable partitioning of the data. In section 3.7 we will provide further, more sophisticated experiments for link-based clustering on relational data.

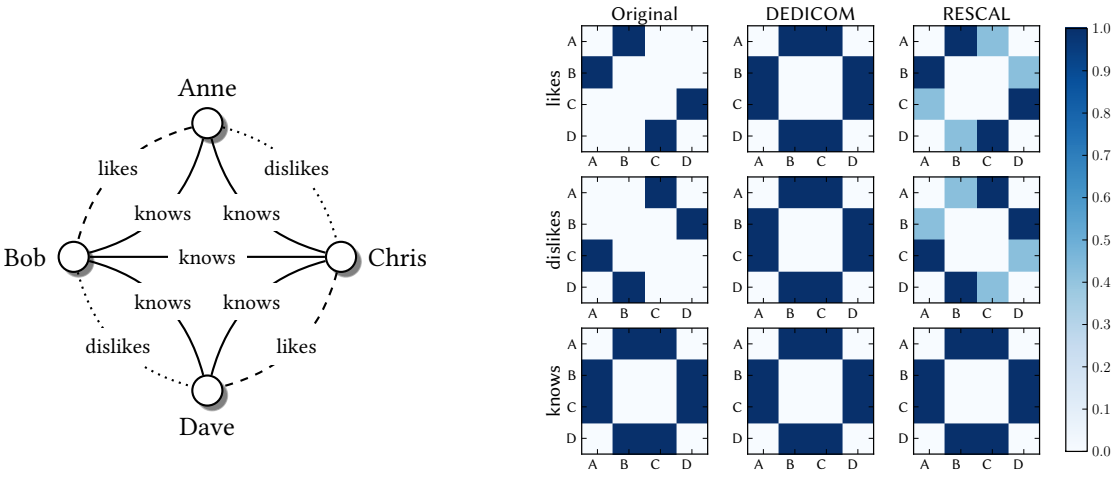
2.6.5. Comparison to DEDICOM

At last, we also conducted experiments on synthetic data to illustrate that the constraints of DEDICOM are not appropriate for *multi*-relational data. Consider the synthetic data set shown in figure 2.11a. It shows a very simple network, where four persons are connected via the

Table 2.4.: Area under the precision-recall-curve for full reconstruction setting and leave-one-out cross-validation on the synthetic likes-dislikes-knows data.

	AUC-PR	
Algorithm	Reconstruction	Leave-one-out
RESCAL	1.0	1.0
DEDICOM	0.833	0.832
TUCKER-2	1.0	0.797
Random	0.358	0.358

relations `likes` , `dislikes` , and `knows` . In the first experiment, the objective was to show that these relations are already to diverse to be modeled by DEDICOM, even without any form of missing data. Therefore, we created the adjacency tensor for this network and directly applied the DEDICOM, the RESCAL, as well as the TUCKER-2 factorization to the full tensor. For each factorization, we choose the rank that produced the largest area under the precision-recall curve without any form of cross-validation. Figure 2.11b shows the reconstructed slices that were produced by the best model for DEDICOM and RESCAL. The column *Reconstruction* in table 2.4 shows the results for the area under the precision-recall curve. It can be seen that even in this simple setting without unknown data and with optimization of the model parameters on the test set, DEDICOM is to constrained to express this data set. RESCAL and TUCKER-2 on the other hand have enough degrees of freedom to model all three relations perfectly. In a second experiment, we considered a real learning setting and performed leave-one-out cross-validation on all combinations of relations and persons. More precisely, for each pair of relation k and person i, we set $\mathbf{X}_{i,:,k} = 0$, meaning that we removed all information where a particular person occurs as a subject in a relation. The task was then to rank all persons by the likelihood that they are related to person i via relation k. Due to the collective learning effect of RESCAL and DEDICOM, both factorizations can exploit the symmetry in the data. RESCAL even gives a perfect result in this setting, although complete rows of $\mathbf{X}$ are unknown. TUCKER-2 on the other hand – which does not know about the identity of entities – gives significantly worse results, not only compared to RESCAL, but also compared to its own results in the reconstruction setting. This underlines again the importance of the constraint that identical entities are modeled by identical latent representations for relational learning.



(a) Illustration of the synthetic likes-dislikes-knows network.

(b) Visualization of the reconstructed network after factorization. The columns show the adjacency matrices for all relations of the original adjacency tensor as well as their reconstructions with DEDICOM and RESCAL.

Figure 2.11.: Example of a simple multi-relational social network, where the DEDICOM model is not expressive enough to model the patterns of the three relations `likes` , `dislikes` and `knows` simultaneously.

2.7. Summary

In this chapter, we proposed RESCAL, a novel approach to multi-relational learning via the factorization of a third-order tensor. We have shown that the factorization can overcome problems of existing SRL methods such as structure learning and costly inference, that it offers a great flexibility in how relational learning tasks can be approached and, maybe most importantly, that it learns from relational data very efficiently, showing state-of-the art performance on multiple benchmark data sets. In section 2.2, we have reasoned how the RESCAL model can fulfil important requirements on a relational learning method such as collective learning and the modeling of important relational patterns. We have also shown that the structure of the factorization decouples the random variables x_{ijk} to enable fast and efficient predictions of relationships, while simultaneously capturing global dependencies between relationships when inferring the state of the latent variables. A central aspect of RESCAL is its unique representation of entities, what allows to propagate information efficiently over the latent factors of the factorization. We have discussed the importance of this modeling for collective learning and shown experimentally that it can improve the learning results on relational data significantly over standard tensor factorizations such as CP and TUCKER. Furthermore, the latent representations of entities in RESCAL provide great flexibility in how relational learning tasks can be approached, as they make the relational similarity of entities available to any feature-based machine learning algorithm. In entity experiments, we have exploited this property of the factorization and achieved state-of-the-art prediction performance. In all experiments, RESCAL showed very good results for all canonical relational learning tasks and outperformed state-of-the-art SRL methods in many cases. Yet, not structure learning is needed for the functioning of the factorization, what improves it applicability significantly. Furthermore, in the next chapter we will show that the factorization is very scalable, such that even large knowledge bases can be factorized.

Chapter 3

Large-Scale Relational Learning and Applications on Linked Data

This chapter is mainly concerned with two topics, namely enabling an efficient algorithm for RESCAL such that it can be used to learn from large knowledge bases and the applications of RESCAL in the Semantic Web and on Linked Data. By exploiting the inherent sparsity of relational data, we will show that RESCAL can be computed very efficiently, scaling linearly with the number of entities, the number of predicates and the number of known facts, such that it can be applied to knowledge bases consisting of millions of entities, hundreds of relations, and possibly billions of known facts. Based on a thorough analysis of its runtime complexity, we will also provide an improved algorithm to compute the RESCAL factorization, which decreases the computational and the memory complexity significantly. Furthermore, we will propose a novel way to handle attributes efficiently within RESCAL via coupled tensor-matrix factorization. Since this chapter is mainly concerned with applications in the Semantic Web and on Linked Data, we will adopt the terminology used in this field. Hence, we will refer to relations as *predicates* and to relationships as *triples*.

3.1. Introduction

The Semantic Web's Linked Data cloud aims to create the "web of data" – a global data space of interlinked information that is created via technologies like RDF, HTTP and URIs (Bizer et al., 2009). At the time of this writing, it consists of hundreds of interlinked databases, where some of these databases store billions of facts in form of RDF triples and it is still

growing rapidly (Cyganiak and Jentzsch, 2011; Auer et al., 2013). For the first time in history, relational data from heterogeneous and interlinked domains is publicly available in large amounts, which, in combination with relational learning, provides exciting opportunities for many fields of application, as state-of-the-art relational learning methods can be expected to utilize much of the information and patterns that are introduced through the relational modeling within and across domain boundaries. Moreover, Linked Data itself can benefit greatly from machine learning: Unlike traditional approaches to the Semantic Web, Linked Data focuses mostly on publishing and linking data on a large scale, whereas less emphasis is put on the formal semantics and ontological descriptions of the published data (Hitzler and Harmelen, 2010). Traditional techniques of the Semantic Web such as reasoning or ontology engineering face therefore some serious challenges in processing information in the Linked Data cloud, due to its size, its inherent noisiness and its inconsistencies. For instance, Halpin et al. (2010) showed that `owl:sameAs` is often misused in the Linked Data cloud, what leads to inconsistencies between different data sources such that reasoning can be problematic in these cases. Further examples of frequently occurring problems for traditional approaches on Linked Data include malformed datatype literals, undefined classes and properties, misuses of ontological terms (Hogan et al., 2010) or the modeling of a simple fact such as "Nancy Pelosi voted in favor of the Health Care Bill" using no less than eight RDF triples (Hitzler and Harmelen, 2010). Moreover, these are not isolated examples, as partial inconsistencies or noise such as duplicate entities and predicates are direct consequences of the open nature of Linked Data. For this reason, it has been recently proposed to look at alternative approaches for new Semantic Web reasoning paradigms (Hitzler and Harmelen, 2010). The underlying idea is that reasoning can often be reduced to the task of classifying the truth value of potential statements. By abandoning requirements such as logical soundness and completeness, this classification can be carried out by approximate methods, such as machine learning. Moreover, it is reasonable to assume that there exist many dependencies in the Linked Data cloud which are rather statistical in nature than deterministic, such that statistical methods have the potential to add significant surplus value in addition to what logical reasoning already provides. Reliable predictions of unknown triples in the scale of entire knowledge bases could mark a first step towards this new reasoning paradigm for the Semantic Web. Here, in our approach to this challenge, we focus on the YAGO 2 ontology (Suchanek et al., 2007), a large knowledge base that lies, along with other databases such as DBpedia (Auer et al., 2008), at the core of the Linked Data cloud.

Applying machine learning to Linked Data at this scale however, is not trivial. For instance,

due to the linked nature of the data, using a relational learning approach is mandatory. But, as discussed in section 1.2.3, many relational learning algorithms often require a considerable amount of prior knowledge about the domain of discourse, e.g. a set of logical formulas in the case of MLNs or the structure of a Bayesian Network for PRMs. This can become a serious obstacle when applying machine learning to Linked Data, since it is difficult and expensive to gather this kind of knowledge manually or automatically for complete knowledge bases. Furthermore, many relational learning algorithms have problems to process data of the size that is required to approach serious, real-life Semantic Web problems as these algorithms usually do not scale well with the number of known facts or entities in the data.

Here, we address these challenges as follows: To enable large-scale relational learning from Linked Data, we employ the RESCAL model from chapter 2, which has been shown to produce very good results for canonical relational learning tasks and which, as it will be shown in section 3.2, fits nicely to the triple structure of RDF(S). In section 3.3 we will discuss multiple tasks that are important for Linked Data and how they can be approached via relational learning in general and with RESCAL in particular. Furthermore, we will also discuss the importance of scalable learning methods for these tasks. In section 3.4 we will show via a thorough complexity analysis of RESCAL-ALS that an implementation which honors the sparsity of relational data scales linearly with the size of the data such that large knowledge bases can be factorized even on commodity hardware. Based on the complexity analysis of section 3.4, we will identify scalability issues of the original RESCAL-ALS algorithm in section 3.5 and propose an improved algorithm that reduces the computational complexity with regard to the number of latent components from $O(r^5)$ to $O(r^3)$ and the memory complexity from $O(r^4)$ to $O(r^2)$. Furthermore, in section 3.6 we will present an extension to the RESCAL model to handle attributes of entities efficiently via coupled tensor-matrix factorization. In section 3.7, we will evaluate the scalability of RESCAL-ALS on synthetic data and by factorizing the YAGO2 core ontology. Furthermore, we will evaluate the performance of RESCAL on learning tasks important to Linked Data. In section 3.8 we will discuss the scalability of RESCAL-ALS with regard to related tensor factorizations and put it into context of related machine learning methods that have been applied to Semantic Web data.

3.2. Modeling Linked Data

Creating a tensor representation of Linked Data is straightforward, as it is built upon RDF(S), which represents information via dyadic relations just as considered in chapter 2. In general,

we will interpret all instances of the classes `rdfs:Resource` and `rdfs:Class` in RDF(S) data as entities, whereas instances of `rdfs:Literal` are interpreted as attribute values. For n entities over m predicates, we employ the same modeling as in chapter 2 and represent Linked Data via the third-order adjacency tensor $\mathbf{X} \in \mathbb{R}^{n \times n \times m}$ where

$$x_{ijk} = \begin{cases} 1, & \text{if the triple } (i\texttt{-th entity}, k\texttt{-th predicate}, j\texttt{-th entity}) \text{ exists} \\ 0, & \text{otherwise.} \end{cases}$$

As in chapter 2, this modeling is, for now, restricted to entity-entity relationships. In section 3.6, we will present an efficient extension to RESCAL, such that attributes of entities, i.e. literal values, can be included in the factorization.

It is important to note that in this modeling of RDF(S) data, we do not draw a distinction between ontological knowledge (the $\mathcal{T}$-Box) and instance data (the $\mathcal{A}$-Box). Instead, for a given domain, classes and all instances of these classes are modeled equally as entities in the adjacency tensor $\mathbf{X}$. Furthermore, all predicates from the $\mathcal{T}$-Box and the $\mathcal{A}$-Box form the slices X_k of $\mathbf{X}$. This way, ontological knowledge is represented similarly to instance data by an appropriate entry $x_{ijk} = 1$, such that facts about instances as well as data from ontologies are integrated simultaneously in a single tensor representation. In doing so, ontologies are handled like soft constraints, meaning that the additional information present in an ontology guides the factorization to semantically more reasonable results, but doesn't impose hard constraints on the model. This makes the model more robust to erroneous or incomplete ontological data such as the inconsistent usage of `owl:sameAs`. Consequently, our modeling has aspects of both a pure data-centric and an ontology-driven Semantic Web approach.

3.3. Machine Learning Tasks on Linked Data

Once the factorization of an adjacency tensor has been computed with RESCAL, it can be used for various learning tasks that are important to Linked Data and the Semantic Web. In the following we will briefly describe some of these tasks and discuss the benefits of RESCAL as well as the advantages of large-scale relational learning in these application scenarios.

Prediction of Triples The prediction of the truth value of triples is an important task in many fields of application and can also be used to improve the data quality in automatically created knowledge bases. This task corresponds the link prediction in relational learning and can be approached with RESCAL as described in section 2.3. A very interesting property of RESCAL is that the state of a single relationship x_{ijk} is conditionally

independent from all other variables given the expression $\boldsymbol{a}_i^T R_k \boldsymbol{a}_j$. Once a factorization of a knowledge base has been computed – what can be done "offline" – this property enables fast query answering, as the computational complexity of the corresponding matrix-vector multiplications depends only on the dimensionality of the latent space A and is independent of the size of a knowledge base. This is an important feature of RESCAL compared to other relational learning approaches where exact inference is often intractable and even approximate inference remains very time consuming.

Instance Matching In instance matching the task is to determine which entities from heterogeneous data sources refer to the same underlying entity; a task that is considered critical for Linked Data (Ferrara et al., 2008; Ferrara et al., 2011). Instance matching is essentially equivalent to entity resolution in relational learning and can be approached with RESCAL, by creating a joint adjacency tensor for all data sources whose instances should be matched and by applying the entity resolution methods as described in chapter 2.

Retrieval of Similar Entities A particular strength of the RESCAL factorization is that it computes a global latent representation of entities, i.e. the factor matrix A. Analogous to the retrieval of documents via latent-variable models, the latent representation of entities can be used to retrieve entities that are similar to a queried entity. As discussed in section 2.2, the matrix A can be interpreted as an embedding of entities into a latent space that reflects their similarity over all relations in the domain of discourse. Therefore, in order to retrieve entities that are similar to a particular entity e with respect to all relations in the data, it is sufficient to compute a ranking of entities by their similarity to e in A. This can be done efficiently, since A is only an $n \times r$ matrix.

Decision Support for Knowledge Engineers Another important application of RESCAL is the automatic creation of taxonomies from instance data. Recently, it has been proposed that machine learning methods should assist knowledge engineers in the creation of ontologies, such that an automated system suggests new axioms for an ontology, which are added under the supervision of an engineer (Auer and Lehmann, 2010). Here, we consider the simpler task of learning a taxonomy from instance data, which can be interpreted as a hierarchical grouping of entities. Consequently, a natural approach to learning a taxonomy for a particular domain is to compute a hierarchical clustering of the entities in this domain and to interpret the resulting clusters according to their members.

However, there are only very few approaches that are able to compute a *hierarchical* clustering for *multi-relational* data (Roy et al., 2007), and even less approaches that could be applied to complete knowledge bases. To compute such a clustering with RESCAL, we exploit again the fact that A reflects the similarity of entities in the relational domain, and simply compute a hierarchical clustering in this latent-component space. This approach has the advantage that any *feature*-based hierarchical clustering algorithm can be readily be applied to this task. The clustering, however, will still be determined by the entities' similarities in the relational domain. While this approach differs in some important aspects from the system envisioned by Auer and Lehmann (2010), it can be used to address some of the discussed challenges, in particular scalability.

For all of these tasks the quality of the model can be improved via the application of RESCAL to complete knowledge bases. For tasks like instance matching and taxonomy learning the scalability to one or multiple knowledge bases is even required, as these tasks are defined over complete knowledge bases. Scaling RESCAL to data sets of this size can therefore be an important step towards relational learning from complete knowledge bases in the Semantic Web, what is one of the main motivations for the work in this chapter.

Another noteworthy aspect about learning on Linked Data is the importance of collective learning, i.e. the inclusion of information that might be more distant in the relational graph in learning and prediction tasks. This ability of a learning method is not only important because of the relational nature of Linked Data as discussed in section 1.2.1, but also because of the way how information is modeled in Linked Data. Since RDF is restricted to dyadic relations, higher-order relations are often modeled via intermediary nodes such as blank nodes or abstract entities, such that the actual fact is not included in a single triple but is spread over a chain of triples. For instance, in version `3.7` of the DBpedia ontology, many geographical locations such as river mouth locations are modeled via chains of triples such as

```
(Rhone, mouthPosition, Rhone-mouthPosition),
(Rhone-mouthPosition, longitude, 4.845555782318115),
(Rhone-mouthPosition, latitude, 43.330799).
```

To access, for instance, the longitude of the river Rhône, a learning method requires therefore the ability to include information that is "past" the abstract entity `Rhone-mouthPosition` , what is enabled through collective learning.

Table 3.1.: Computational complexity of standard matrix operations.

Operation	Notation	Relevant Parameters	Complexity
Matrix Multiplication[a]	AB	$A \in \mathbb{R}^{n \times m}$, $B \in \mathbb{R}^{m \times \ell}$	$O(nm\ell)$
Kronecker Product[b]	$A \otimes B$	$A \in \mathbb{R}^{p \times r}, B \in \mathbb{R}^{q \times s}$	$O(pqrs)$
Sparse Matrix × Dense Vector[a]	$A\boldsymbol{b}$	$p = \text{nnz}(A)$	$O(p)$
Sparse Matrix × Dense Matrix[c]	AB	$p = \text{nnz}(A), B \in \mathbb{R}^{n \times m}$	$O(pm)$
Matrix Inversion[d]	A^{-1}	$A \in \mathbb{R}^{n \times n}$	$O(n^3)/O(n^{2.5})$
Singular Value Decomposition[e]	$A = U\Sigma V^T$	$A \in \mathbb{R}^{m \times n}$, $m \geq n$	$O(mn^2)$

[a]See Boyd and Vandenberghe (2004, Section C.1.2)

[b]Follows from $A \otimes B$ being a $pq \times rs$ matrix and the fact that each entry of $A \otimes B$ has to be computed individually when there is no special structure in A or B.

[c]Follows from computing m times the sparse matrix × dense vector product $A\boldsymbol{b}_i$, where $\boldsymbol{b}_i$ denotes the i-th column of B.

[d]See Meyer (2000, p.119) for the standard method, Press et al. (2007, p.108) for the theoretically fastest method. For all implementations, we consider the complexity of the standard approach $O(n^3)$.

[e]See Trefethen and Bau III (1997, p.236)

3.4. Complexity Analysis of RESCAL-ALS

Scaling learning methods to large data sets, requires low computational complexity and low memory usage. Bottou and Bousquet (2008) argue that algorithms for large-scale learning should "scale roughly linearly" with the size of the data. It has already been discussed in chapter 2 that an important characteristic of relational data is its sparsity, i.e. the fact that usually only a small number of all possible relationships are true. The key insight to enable a scalable implementation of RESCAL-ALS is that this sparsity of relational data translates directly to its representation as an adjacency tensor. An entry x_{ijk} in an adjacency tensor $\mathbf{X}$ is non-zero if and only if the corresponding relationship is true. Consequently, $\mathbf{X}$ is sparse if and only if the relational data is sparse. Here, we will show that by using sparse linear algebra it is possible to exploit this property of $\mathbf{X}$ such that a sparse implementation of RESCAL-ALS has indeed only a linear computational complexity with regard to the number of entities, the number of predicates and the number of known facts in a data set. In this analysis we will assume that each frontal slice X_k is a sparse matrix such that the number of its non-zero entries $\text{nnz}(X_k)$ is much smaller than the number of possible entries, i.e. $\text{nnz}(X_k) \ll n^2$. The latent factors of the factorization, i.e. the matrices A and R_k, are assumed to be dense. Furthermore, in table 3.1 we list the computational complexity of standard matrix operations that will be used throughout this chapter.

The relevant parameters of the data and the factorization for the computational complexity

are the number of objects n, the number of relations m, the number of known facts $p = \text{nnz}(\mathbf{X})$, and the model complexity, i.e. the number of latent components r. It is easy to see that, due to the Tucker-2 structure of the Rescal model, the update steps for A and R are linear in the number of relations m – regardless of the sparsity of $\mathbf{X}$ – since the parameter m occurs only in the summation indices of equation (2.11) and in the iteration over all frontal slices R_k in equation (2.15). However, the computational complexity with regard to the number of entities n, the number of known facts p and the number of latent components r is more elaborate. Table 3.2 summarizes the complexity of each relevant operation in the update steps of A and $\mathbf{R}$, from which it can be seen that the parameters n, m, and p occur only as linear factors. Since these update steps are iterated only a small number of times until the algorithm reaches convergence or until a maximum number of iterations is exceeded, it follows that a sparse implementation of Rescal-ALS scales linearly with the size of data. The big advantage of a sparse over a dense implementation becomes apparent when considering the calculation of the terms $X_k^T A$ or $X_k A$ that occur multiple times in the updates of A and R_k. Computing these terms via *dense* matrix-matrix products would lead to a computational complexity of $O(n^2 r)$, since $X_k \in \mathbb{R}^{n \times n}$ and $A \in \mathbb{R}^{n \times r}$. In contrast, by using *sparse* matrix-matrix products, computing $X_k A$ leads to a complexity of $O(pr)$, such that the operation becomes independent of the size of X_k and only depends on the non-zero entries of X_k. For relational data, this is equivalent to becoming independent of the number of entities and being only dependent on the number of known facts. Consequently, for relational data a large reduction in the computational complexity can be gained by a sparse implementation as it is usually the case that $\text{nnz}(X_k) \ll n^2$. However, it can also be seen from table 3.2 that a standard approach to the computation of normal equations results in a computational complexity that is quintic in the number of latent components for the updates of $\mathbf{R}$, even when considering the theoretically fastest known matrix inversion method. Since this property would be prohibitive for problems with a larger number of latent components, we will present an improved algorithm that is only cubic in the parameter r in section 3.5.

The memory complexity of Rescal-ALS is the second important factor for its scalability. For now, consider the memory requirements of Rescal-ALS without the operation $F \leftarrow A^T A \otimes A^T A$ which has a memory complexity of $O(r^4)$. In each iteration, only one frontal slice X_k has to be kept in memory. Since sparse matrix implementations like the compressed spare row (CSR) and compressed spare blocks (CSB) format have only a memory complexity of $O(n + \text{nnz}(X_k))$ (Buluç et al., 2009), our approach scales up to billions of known facts with regard to it memory requirements. Moreover, except for the computation of $A^T A \otimes A^T A$, all

Table 3.2.: Computational complexity for operations in the update steps of A and R according to table 3.1. Variables and terms listed in column "Known Terms and Variables" are assumed to be already computed and do not affect the complexity of the respective row. Since operations which differ only in the terms AR_k and AR_k^T have identical computational complexity, we only list operations involving AR_k. Variable assignments are indicated by the symbol "←".

Update A				
Operation	Complexity	Known Terms and Variables		
A^TA	$\mathcal{O}(nr^2)$	$A \in \mathbb{R}^{n \times r}$		
AR_k	$\mathcal{O}(mnr^2)$	$A \in \mathbb{R}^{n \times r}$	$R_k \in \mathbb{R}^{r \times r}$	$k = 1 \ldots m$
$E \leftarrow \sum_k X_k AR_k^T$	$\mathcal{O}(pr)^a$	$AR_k \in \mathbb{R}^{n \times r}$	$p = \text{nnz}(\mathbf{X})$	$k = 1 \ldots m$
$B_k \leftarrow R_k A^T AR_k^T$	$\mathcal{O}(mr^3)$	$A^TA \in \mathbb{R}^{r \times r}$	$R_k \in \mathbb{R}^{r \times r}$	$k = 1 \ldots m$
$F \leftarrow (\sum_k B_k + C_k)^{-1}$	$\mathcal{O}(mr^2 + r^3)$	$B_k \in \mathbb{R}^{r \times r}$	$C_k \in \mathbb{R}^{r \times r}$	$k = 1 \ldots m$
$A \leftarrow EF$	$\mathcal{O}(nr^2)$	$E \in \mathbb{R}^{n \times r}$	$F \in \mathbb{R}^{r \times r}$	
Full Update	$\mathcal{O}(m(r^3 + nr^2) + pr)$			
Update R				
Operation	Complexity	Known Terms and Variables		
A^TA	$\mathcal{O}(nr^2)$	$A \in \mathbb{R}^{n \times r}$		
$E \leftarrow A^TA \otimes A^TA$	$\mathcal{O}(r^4)$	$A^TA \in \mathbb{R}^{r \times r}$		
$F \leftarrow (E + \lambda I)^{-1}$	$\mathcal{O}(r^5)$	$E \in \mathbb{R}^{r^2 \times r^2}$		
$G_k \leftarrow A^T X_k A$	$\mathcal{O}(mnr^2 + pr)$	$A \in \mathbb{R}^{n \times r}$	$p = \text{nnz}(\mathbf{X})$	$k = 1 \ldots m$
$R_k \leftarrow F \,\text{vec}\,(G_k)$	$\mathcal{O}(mr^4)$	$\text{vec}\,(G_k) \in \mathbb{R}^{r^2}$		$k = 1 \ldots m$
		$F \in \mathbb{R}^{r^2 \times r^2}$		
Full Update	$\mathcal{O}(m(r^4 + nr^2) + r^5 + pr)$			

[a] Please note that p denotes the non-zero entries of the complete tensor $\mathbf{X}$. Hence, no additional factor m is required.

operations in table 3.2 that involve dense matrices are only carried out on matrices of size at most $n \times r$. In cases where even this very moderate memory requirement is too demanding, for instance when a domain contains a very large number of entities, additional dimensionality reduction techniques such as the "hashing trick" (Weinberger et al., 2009; Karatzoglou et al., 2010) can be applied. Similar as for the computational complexity, the problematic operation $A^T A \otimes A^T A$ is introduced through the update step of $\mathbf{R}$. The improved algorithm presented in section 3.5 will also reduce the memory complexity of these updates to a complexity that is quadratic with regard to the number of latent components.

Concerning the scalability of Rescal-ALS, it is also interesting to note that the algorithm can be computed easily in a distributed way. It can be seen from Table 3.2 that the dominant costs in each update step A are the matrix multiplications $X_k A R_k^T + X_k^T A R k$, since $mnr^2 > mr^3$ for $r < n$. Due to the sums in the update of A, this step can be computed distributedly by using a map-reduce approach. First, the current state of A and R_k is distributed to a number of available computing nodes. Then, these nodes compute $X_k A R_k^T + X_k^T A R_k$ as well as the term $B_k + C_k$ locally for those k that have been assigned to them. Given the results of these computations, the master node can then reduce the results and compute the matrix inversion, which involves only $r \times r$ matrices and the final matrix product. Since the updates of R_k are independent of each other, these steps can be computed in a similar way.

3.5. Scalable Core Tensor Updates in Rescal-ALS

It has been shown in section 3.4 that the overall computational complexity for the update steps of $\mathbf{R}$ becomes $O(m(r^4 + nr^2) + r^5 + pr)$ when using a standard approach to normal equations. Moreover, the computations within this update step involve the computation of the matrix $A^T A \otimes A^T A$, which is of size $r^2 \times r^2$, such that the overall memory complexity becomes $O(r^4 + nr + p)$. Both of these properties are unacceptable for models that require more than a few latent components, what is very likely to occur on large-scale and complex data. In the following, we will derive a scalable algorithm to compute updates of $\mathbf{R}$ within Rescal-ALS, by exploiting special properties of the Kronecker product. These improved updates will feature only a computational complexity that is has cubic computational and quadratic memory complexity with regard to the number of latent components and thus can be applied to problems that require a moderate to large number of latent components.

For models without regularization, i.e. when $\lambda = 0$, it is relatively straightforward to reduce the computational complexity for the updates of R_k. By using properties of the Kronecker

product outlined in section 1.3.1, we can show that

$$R_k \leftarrow \left[(A \otimes A)^T (A \otimes A)\right]^{-1} \operatorname{vec}\left(A^T X_k A\right) \tag{3.1}$$

$$= \left(A^T A\right)^{-1} \otimes \left(A^T A\right)^{-1} \operatorname{vec}\left(A^T X_k A\right) \tag{3.2}$$

$$= \left(A^T A\right)^{-1} A^T X_k A \left(A^T A\right)^{-1} \tag{3.3}$$

where equation (3.2) follows from equations (1.9) to (1.11) and equation (3.3) follows from equation (1.12). Now, the computation of equation (3.3) requires only the inversion of an $r \times r$ matrix $A^T A$, such that it has an overall computational complexity of $O(r^3 + nr^2 + \operatorname{nnz}(X_k)r)$. Unfortunately, for models with regularization, i.e. when $\lambda \neq 0$, the inverse in equation (2.15) ranges over a *sum* of matrices, such that it can not be reduced to a simpler computation as in the non-regularized case. Since regularization is highly desirable from a machine learning and also from a numerical point of view, it is important to enable scalable updates for this class of models, which we will derive in the following. Evidently, each frontal slice R_k in equation (2.15) is computed by solving a separate ridge regression problem of the form

$$\boldsymbol{b} = \left(M^T M + \lambda I\right)^{-1} M^T \boldsymbol{v}, \tag{3.4}$$

where $\operatorname{vec}(R_k)$ corresponds to $\boldsymbol{b}$, M corresponds to $A \otimes A$, and $\operatorname{vec}(X_k)$ corresponds to $\boldsymbol{v}$. It is well-known, that equation (3.4) can be solved via the singular value decomposition of M, as described in the following theorem:

Theorem 1 (Lange, 2010, Section 9.3.2)**.** *Let $M = U\Sigma V^T$ be the singular value decomposition of M. The solution to the ridge regression problem equation* (3.4) *is then given by*

$$\boldsymbol{b} = V\widehat{\Sigma}U^T \boldsymbol{v}$$

where $\widehat{\Sigma}$ is a diagonal matrix with entries

$$\widehat{\Sigma}_{ii} = \frac{\Sigma_{ii}}{\Sigma_{ii}^2 + \lambda}.$$

Since $M = A \otimes A$ is an $n^2 \times r^2$ matrix, the computational complexity of its SVD would be $O(n^2 r^4)$ such that a direct computation of theorem 1 would still be far too costly for reasonably large values for n and r. However, since M is the result of a Kronecker product, we can apply the following theorem, which enables an efficient computation of its SVD:

Theorem 2 (Laub, 2004, Theorem 13.10). *Let $A = U_A \Sigma_A V_A^T$ and $B = U_B \Sigma_B V_B^T$ be the singular value decompositions of matrices A and B. Then*

$$(U_A \otimes U_B)(\Sigma_A \otimes \Sigma_B)(V_A \otimes V_B)^T$$

yields a singular value decomposition of the Kronecker product $A \otimes B$.

Consequently, to compute the SVD of $A \otimes A$, it is sufficient to compute the SVD of A – which has only a time complexity of $O(nr^2)$. Yet, the application of theorem 2 alone is not sufficient to solve theorem 1 efficiently, as it would involve the *dense* matrix $(V \otimes V)(\Sigma \otimes \Sigma)(U \otimes U)^T$, which is of size $r^2 \times n^2$ and thus intractable to compute for moderate to large values of n and r. Fortunately, it is not necessary to compute these Kronecker products explicitly, what we will show via the combination of equation (1.12) and the following theorem:

Theorem 3 (Minka, 2000, eq. 66). *Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times n}$ be matrices of identical size. Then it holds that*

$$\operatorname{vec}(A * B) = \operatorname{diag}(\operatorname{vec}(A)) \operatorname{vec}(B),$$

where the symbol "$*$" denotes the entry-wise product of matrices such that $(A * B)_{ij} = a_{ij} b_{ij}$

Now, let $A = U \Sigma V^T$ be the singular value decomposition of A, let $S = \Sigma \otimes \Sigma$, and let $\widehat{S}$ be a diagonal matrix with entries

$$\widehat{S}_{ii} = \frac{S_{ii}}{S_{ii}^2 + \lambda}.$$

Furthermore, let P be the matrix such that $\operatorname{diag}(\operatorname{vec}(P)) = \widehat{S}$, which can be constructed by rearranging the diagonal entries of $\widehat{S}$ via the inverse vectorization operator $\operatorname{vec}_r^{-1}(\cdot)$ from definition 5. Then, the following equivalences apply

$$\begin{aligned}
\operatorname{vec}(R_k) &= \left((A \otimes A)^T (A \otimes A) + \lambda I\right)^{-1} (A \otimes A)^T \operatorname{vec}(X_k) && (3.5)\\
&= (V \otimes V)\, \widehat{S}\, (U \otimes U)^T \operatorname{vec}(X_k) && (3.6)\\
&= (V \otimes V)\, \widehat{S}\, \operatorname{vec}\left(U^T X_k U\right) && (3.7)\\
&= (V \otimes V) \operatorname{vec}\left(P * U^T X_k U\right) && (3.8)
\end{aligned}$$

where equation (3.6) follows from theorem 1, equation (3.7) follows from equation (1.12), and equation (3.8) follows from theorem 3. It follows again from equation (1.12) that equation (3.8) can be further reduced to

$$R_k = V \left(P * U^T X_k U\right) V^T \tag{3.9}$$

Now, all Kronecker products have been removed in equation (3.9) such that updates for $\mathbf{R}$ can be computed efficiently. Algorithm 1 lists the entire computation for an update of $\mathbf{R}$ and table 3.3 lists the computational complexity of the important parts of this algorithm as well as its overall complexity. It can be seen that the computational complexity is reduced to $O(m(r^3 + nr^2) + pr)$ what matches the complexity of the non-regularized updates and what is a large improvement over the complexity $O(m(r^4 + nr^2) + r^5 + pr)$ of the standard approach to normal equations. Moreover, the memory complexity has also been reduced to $O(r^2 + nr + p)$ from originally $O(r^4 + nr + p)$.

Algorithm 1 Scalable computation of updates for R

Input:

Adjacency tensor $\mathbf{X} \in \mathbb{R}^{n \times n \times m}$; Latent entity representations $A \in \mathbb{R}^{n \times r}$;
Regularization parameter $\lambda_R \geq 0$; Number of latent components $r \in \mathbb{N}_+$.

Output:

Core tensor $\mathbf{R}$.

function UPDATE_R($\mathbf{X}$, A, λ_R, r)
  $U, \boldsymbol{s}, V \leftarrow \texttt{svd}(A)$ ▷ $\boldsymbol{s}$: *vector* holding the singular values of A
  $\widehat{\boldsymbol{s}} \leftarrow \boldsymbol{s} \otimes \boldsymbol{s}$
  for $i = 1, \ldots, r^2$ **do**
    $\widehat{\boldsymbol{s}_i} \leftarrow \frac{\widehat{\boldsymbol{s}_i}}{\left(\widehat{\boldsymbol{s}_i}^2 + \lambda\right)}$
  end for
  $P \leftarrow \texttt{reshape}(\widehat{\boldsymbol{s}}, r, r)$
  for $k = 1, \ldots, m$ **do**
    $R_k \leftarrow V(P * (U^T X_k U))V^T$
  end for
  return R
end function

3.6. Learning from Attributes via Coupled Tensor Factorization

In chapter 2, we were mostly concerned with collective learning over entity-entity relationships and didn't handle the attributes of entities explicitly. However, much information in the LOD cloud and in relational data in general is in the form of attributes[1] such that it is important for a relational learning method to handle attributes efficiently. A possible approach to

[1] In the Semantic Web terminology, attributes are usually called *datatype* properties and attribute values are called *literals*.

Table 3.3.: Complexity analysis for relevant computations in Algorithm 1

Scalable Updates R				
Operation	Complexity	Known Terms and Variables		
$A = U\Sigma V^T$	$\mathcal{O}(nr^2)$	$A \in \mathbb{R}^{n \times r}$		
$E_k \leftarrow U^T X_k U$	$\mathcal{O}(mnr^2 + pr)$	$U \in \mathbb{R}^{n \times r}$	$p = \text{nnz}(\mathbf{X})$	$k = 1 \dots m$
$F_k \leftarrow P * E_k$	$\mathcal{O}(mr^2)$	$E_k \in \mathbb{R}^{r \times r}$	$P \in \mathbb{R}^{r \times r}$	$k = 1 \dots m$
$R_k \leftarrow V F_k V^T$	$\mathcal{O}(mr^3)$	$F_k \in \mathbb{R}^{r \times r}$	$V \in \mathbb{R}^{r \times r}$	$k = 1 \dots m$
Full Update	$\mathcal{O}(m(r^3 + nr^2) + pr)$			

include attributes in the factorization would be to discretize the attribute values in the data, to replace all triples that include attribute values with the results of this discretization step, and to create an adjacency tensor from these preprocessed triples. Possible ways of discretization include the thresholding of continuous variables at their mean, the transformation of categorical variables via one-out-of-n coding, and the transformation of textual data via tokenizing and stemming. Then, all attribute triples would be replaced by triples of the form

$$(\texttt{entity}\ ,\ \texttt{hasAttribute}\ ,\ (\texttt{attribute name}\ ,\ v))$$

where v is the result of the discretization step. In the following, we will refer to the tuples $(\texttt{attribute name}\ , v)$ also as *attribute pairs*. If the discretization of an attribute value produces multiple values $v_1, \dots, v_n$, a single attribute triple is replaced by n different discretized triples where $v = v_1, \dots, v_n$. For instance, assuming that textual data is preprocessed via tokenization, an attribute value like

```
(Albert_Einstein, foaf:name, 'Albert Einstein')
```

would be replaced by the triples

```
(Albert_Einstein , hasAttribute , (foaf:name , 'Albert'))
(Albert_Einstein , hasAttribute , (foaf:name , 'Einstein'))
```

This is essentially the approach considered by Kemp et al. (2006) to include attributes in the Infinite Relational Model. While this handling of attributes can work for small to medium sized data sets, it is not well-suited for large-scale learning on the Semantic Web. The main problem associated with this procedure is that the attribute pairs are included as true entities in the adjacency tensor, what increases the size of the entity modes significantly. Although the number of entities n appears only as a linear factor in table 3.2, it is important to note

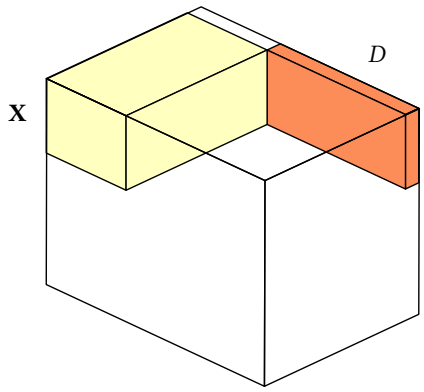


Figure 3.1.: Tensor and matrix structure in attribute modeling on relational data. The adjacency tensor $\mathbf{X}$ holds the multi-relational information, while the matrix D holds the attribute information. A combined modeling largely increases the size of a tensor representation. Furthermore, attributes only occur as objects in the attribute slice.

that it is also multiplied by the number of predicates m and the squared number of latent components r^2 in the updates of A and $\mathbf{R}$, such that an increase in n can have a noticeable effect on the runtime of the algorithm. However, it lies in the nature of attribute values and attribute pairs that they occur never as subjects in a relation or in any predicate other then the `hasAttribute` predicate, such that all the additional computations that are introduced through attribute pairs would be wasted on approximating a part of the adjacency tensor that does not hold any information and is all zero. Figure 3.1 shows an illustration of this effect. To overcome this problem, we propose to handle attributes by a separate matrix factorization which we perform jointly with the tensor factorization. The basic idea is to discretize attribute values just as described above, but to include the resulting attribute pairs not as new entities in the adjacency tensor, but as columns in a separate entity-attributes matrix $D \in \mathbb{R}^{n \times \ell}$. The entries d_{ij} of D are then set to

$$d_{ij} = \begin{cases} 1, & \text{if the } i\text{-th entity has an attribute value corresponding to the } j\text{-th attribute pair} \\ 0, & \text{otherwise.} \end{cases}$$

The matrix D is therefore constructed similar as in the relational learning algorithm Suns (Huang et al., 2010; Huang et al., 2011), with the only difference, the D runs only over all attributes in the data, while in Suns all relations are modeled this way. The entity-attributes matrix D is then factorized into the matrices $A \in \mathbb{R}^{n \times r}$ and $F \in \mathbb{R}^{r \times \ell}$ such that

$$D \approx AF. \tag{3.10}$$

The important part of this factorization is that the latent factor A is shared between the factorization of the adjacency tensor $\mathbf{X}$ and the attribute matrix D. This way, relationships *and* attribute values influence the latent representations of entities, such that attributes influence also the prediction of relationships and vice versa. In the following, we will refer to this method as *coupled tensor-matrix factorization*. To include equation (3.10) as an additional constraint on A in the tensor factorization of $\mathbf{X}$, we add the term $\|D - AF\|^2 + \lambda_F \|F\|^2$ to the minimization problem (2.5), such that the full optimization problem becomes

$$\min_{A,R,F} \|\mathbf{X} - \mathbf{R} \times_1 A \times_2 A\|^2 + \|D - AF\|^2 + \lambda_A \|A\|^2 + \lambda_{\mathbf{R}} \|\mathbf{R}\|^2 + \lambda_F \|F\|^2 \tag{3.11}$$

To adapt RESCAL-ALS to this new objective, the update step of A has to be changed such that it takes the additional factorization of D into account and a new update step for F has to be included in the algorithm. In particular, the update steps for A and F become:

Updates for A To adapt the updates of A to the new objective equation (3.11), we add two additional terms DF^T and FF^T to the updates in equation (2.11), such that an update for A is computed by

$$A \leftarrow \left(DF^T + \sum_{k=1}^{m} X_k A R_k^T + X_k^T A R_k \right) \left(FF^T + \sum_{k=1}^{m} B_k + C_k + \lambda_A I \right)^{-1} \tag{3.12}$$

where, as in section 2.5, the matrices B_k, C_k are

$$B_k = R_k A^T A R_k^T, \quad C_k = R_k^T A^T A R_k.$$

The derivation of this update step is essentially identical to the derivation of updates for A in section 2.5: First, we extend equation (3.11) by considering all frontal slices X_k, X_k^T, R_k, and R_k^T simultaneously and then solve the optimization problem only for the left hand factor A. However, because of the additional term $\|D - AF\|^2$, the method of normal equations can not be applied directly as in section 2.5. Instead, we compute the partial derivatives of equation (3.11) for the left hand A, set this gradient to zero, and then solve for A. The result of this computation is the update step equation (3.12).

Updates for F To compute updates for F, it is sufficient to consider only the subproblem $\|D - AF\|^2 + \lambda_F \|F\|^2$, since all other terms in equation (3.11) are independent of F. This is again a Tikhonov regularization problem and can therefore be solved by

$$F \leftarrow \left(A^T A + \lambda_F I \right)^{-1} A^T D.$$

Table 3.4.: Computational complexity added through coupled tensor-matrix factorization for attributes.

Update A		
Additional Operation	Complexity	Known Terms and Variables
DF^T	$\mathcal{O}(qr)$	$F \in \mathbb{R}^{r \times \ell}$ $q = \text{nnz}(D)$
FF^T	$\mathcal{O}(\ell r^2)$	$F \in \mathbb{R}^{r \times \ell}$
Full Update	$\mathcal{O}(m(r^3 + nr^2) + \ell r^2 + (p+q)r)$	
Update F		
Operation	Complexity	Known Terms and Variables
$(A^TA + \lambda_F I)^{-1}$	$\mathcal{O}(r^3 + r)$	$A^TA \in \mathbb{R}^{r \times r}$
A^TD	$\mathcal{O}(qr)$	$A \in \mathbb{R}^{n \times r}$ $q = \text{nnz}(D)$
Full Update	$\mathcal{O}(r^3 + qr)$	

Handling attributes in this way is essentially equivalent to the naive approach, but significantly more efficient to compute. Table 3.4 lists the computational complexity of the additional operations that are necessary to include attributes in the factorization.[1] It can be seen, that while the additional operations will increase the runtime of the algorithm, they do not alter the linear scalability of RESCAL-ALS. Moreover, the computational complexity with regard to the number of attributes and predicates is greatly improved. Let n be the number of entities, ℓ be the number of attributes, and m be the number of predicates. While a naive approach that treats attributes as entities would have a computational complexity with regard to these parameters of $\mathcal{O}(m(n+\ell))$, the coupled tensor-matrix factorization has only a complexity of $\mathcal{O}(mn + \ell)$, what can improve the runtime of the algorithm significantly for a large number of attributes or predicates.

3.7. Experiments

In the following, we will evaluate the scalability of RESCAL on synthetic data as well as on real-world data. Furthermore, we evaluate its performance for various learning tasks on Linked Data discussed in section 3.2. In particular, we evaluate large-scale link prediction and the automatic construction of taxonomies. For instance matching we refer to the entity resolution experiments in section 2.6. As in chapter 2, all experiments have been evaluated on a single Intel Core 2 Duo machine with two 2.5GHz cores and 4GB RAM, except for

[1] The operation A^TA is not included, since it can be reused from the update step of R_k.

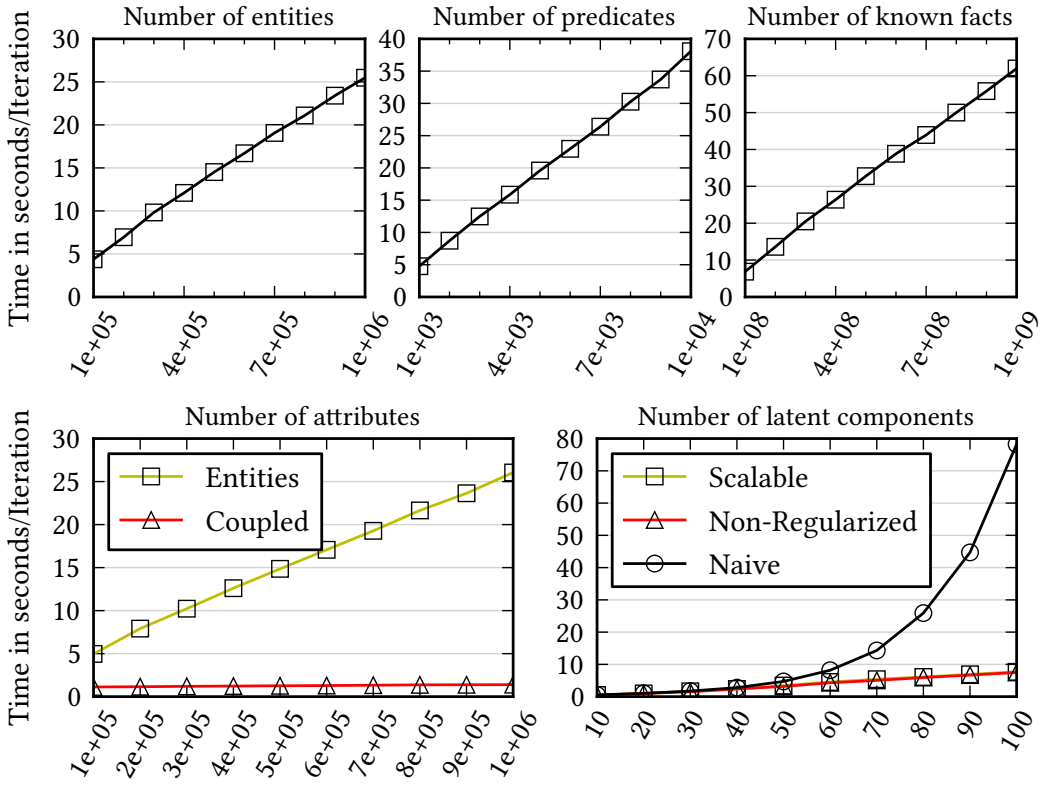


Figure 3.2.: Scalability experiments on synthetic data. In the attributes plot, *Entities* denotes the handling of attributes as entities, while *Coupled* denotes the handling of attributes via coupled tensor factorization. In the latent components plot, *Scalable* denotes the scalable update algorithm for $\mathbf{R}$ of section 3.5, *Non-regularized* denotes updates without regularization, and *Naive* denotes the naive implementation of section 3.4.

experiments where noted otherwise.

3.7.1. Runtime Experiments on Synthetic Data

To evaluate the scalability of our approach, we first conducted experiments on synthetic data, where we created various random data sets with different numbers of entities n, numbers of predicates m, and nonzero entries p. In each experiment we varied exactly one of these parameters, while keeping all other parameters fixed. Then, to evaluate how well RESCAL-ALS scales with regard to a particular parameter, we computed a factorization of the created data set with a fixed number r of latent components and recorded the average runtime of a single iteration in the ALS algorithm. In a further experiment, we also evaluated the runtime of RESCAL-ALS with respect to different values for r, while keeping n, m and p fixed. At last, by increasing the number of attribute values, we also evaluated how the coupled tensor-matrix

Table 3.5.: Parameter values for scalability experiments. The columns *Minimum* and *Maximum* list the minimum and maximum values used in the experiments when the respective parameter is varied. The column *Step* lists the step sizes in which the values are increased from minimum to maximum. The column *Default* lists the default value when a parameter is *not* varied.

	Values			
Parameter	Minimum	Maximum	Step	Default
Entities	10^5	10^6	10^5	10^4
Predicates	10^3	10^4	10^3	50
Facts	10^8	10^9	10^8	10^7
Rank	10	100	10	20
Attributes	10^7	10^8	10^7	0 / 10^8

factorization of section 3.6 scales compared to an approach where attribute values are handled as true entities. Table 3.5 lists the value ranges that have been used for each parameter in the experiments, while figure 3.2 shows the average runtime over 20 iterations for these parameter settings. It can be seen that the results of this evaluation correspond nicely to the theoretical analysis in sections 3.4 and 3.5. In figure 3.2, RESCAL-ALS shows linear scalability with respect to the number of entities, the number of predicates and the number of known facts. Even for large parameter values such as 10^9 nonzero entries, i.e. one billion known facts, the runtime for a single iteration barely exceeds one minute. Furthermore, it can be seen that the updates for $\mathbf{R}$ as developed in section 3.5 greatly improve the scalability of the algorithm compared to a naive implementation. In fact, the improved updates for $\mathbf{R}$ scale identical to the non-regularized variant of the algorithm. At last, it can also be seen that the coupled tensor-matrix algorithm of section 3.6 greatly improves the scalability with regard to the number of attribute values. As expected from the theoretical analysis, handling attributes as true entities shows still linear scalability. However, the naive approach has a much steeper slope in the runtime plot compared to the coupled factorization, because the naive approach also tries to approximate large parts of the tensor that do not hold any valuable information.

3.7.2. Comparative Runtime Experiments

To compare the runtime performance of RESCAL to other tensor decompositions on relational data, we computed factorizations of the benchmark data sets Kinships, Nations, UMLS, and Cora from section 2.6. As comparison to RESCAL we included CP, which has been used by Kolda et al. (2005) for learning from large multigraphs, and DEDICOM which has been used by

Table 3.6.: Runtime in seconds of various tensor factorization on benchmark data sets. Numbers in parentheses denote the number of iterations that were required to reach convergence. |E| denotes the number of entities, |R| the number of relations in the data. The symbol - indicates that the algorithm did not converge.

Dataset	Algorithm	Total Runtime					
		Number of Latent Components					
		10		20		40	
Kinships	CP-ALS	0.74s	(30)	1.36s	(36)	4.11s	(72)
|E|: 104, |R|: 26	ASALSAN	14.96s	(36)	42.62s	(57)	173.08s	(61)
	RESCAL-ALS	**0.15s**	(13)	**0.23s**	(17)	**0.26s**	(14)
Nations	CP-ALS	1.42s	(75)	2.64s	(101)	3.57s	(79)
(|E|: 125, |R|: 57)	ASALSAN	41.72s	(43)	109.44s	(55)	-	(-)
	RESCAL-ALS	**0.76s**	(35)	**0.15s**	(7)	**0.06s**	(2)[a]
UMLS	CP-ALS	0.58s	(25)	1.66s	(50)	4.45s	(81)
|E|: 135, |R|: 49	ASALSAN	31.71s	(32)	101.61s	(35)	423.95s	(35)
	RESCAL-ALS	**0.19s**	(11)	**0.26s**	(13)	**0.44s**	(15)
Cora	CP-ALS	9.94s	(10)	36.36s	(18)	99.92s	(25)
|E|: 2497, |R|: 7	ASALSAN	217s	(4)	444s	(5)	2085s	(11)
	RESCAL-ALS	**2.33s**	(3)	**15.94s**	(23)	**4.04s**	(5)

[a]Initialization with 40 latent components already reaches nearly perfect reconstruction.

Bader et al. (2007) for learning from time-varying networks. To compute these factorizations, we used the CP-ALS (Kolda and Bader, 2009) and ASALSAN (Bader et al., 2007) algorithms. To compute the RESCAL model, we used RESCAL-ALS. Table 3.6 lists the results of these experiments in terms of the runtime for different numbers of latent components. It can be seen that RESCAL-ALS is at least one order of magnitude faster to compute than ASALSAN and also outperforms CP-ALS on a consistent basis. These results are especially noteworthy when considering that RESCAL outperforms CP and DEDICOM also in terms of the learning results on these data sets as shown in section 2.6. Furthermore, we compared the runtime of RESCAL to the runtime of MLN, for a comparison to state-of-the-art relational learning methods. Since there are no published sets of formulas for Kinships, Nations, and UMLS, we applied structure learning as implemented in the ALCHEMY toolbox to these data sets. Unfortunately, the structure learning process did not converge within 24 hours of runtime on these data sets. For the Cora data set, we applied the `learnwts` command of the ALCHEMY toolbox to the (B+N+C+T) MLN as published in Singla and Domingos (2006a) – to learn

Table 3.7.: Statistics for YAGO2

YAGO2 core ontology	
Number of Resources	2.6 million
Number of Classes	340,000
Number of Predicates	87
Number of Known Facts	33 million

the weights for this set of logical formulas.[1] The weight learning process required over 27 minutes of runtime, whereas RESCAL-ALS can be computed within seconds as shown in table 3.6. Moreover, inferring the truth-value for all `SameBib` relationships in a test-fold with lifted first-order belief propagation (Singla and Domingos, 2008) required over 53 minutes of runtime and 5GB of memory, whereas the computation for a RESCAL model with 50 latent components requires less than 0.1 seconds of runtime and less than 170MB of memory. Again, these results are especially noteworthy considering the learning results of RESCAL compared to MLNs as shown in section 2.6.

3.7.3. Large-Scale Prediction of Unknown Triples

Given these very promising results in terms of runtime scalability, the next objective in our experiments was to evaluate the ability of RESCAL to factorize complete knowledge bases and to predict unknown triples in this large-scale setting. For this reason, we conducted several link-prediction experiments on the *entire* YAGO2 core ontology (Suchanek et al., 2007; Hoffart et al., 2011). In these experiments, we used the YAGO2 core ontology in form of the publicly available N3 data set without reified meta-facts in version `20110315`[2], for which statistics relevant to RESCAL are listed in table 3.7. Out of the 87 predicates that are included in this knowledge base, we treated 38 predicates as entity-to-entity relations, while the remaining predicates were handled as attributes. Furthermore, we included the materialization of all `rdf:type` triples and transitive rules, which can be done conveniently via the YAGO conversion tools.[3] This resulted in a total of 64 million triples. From the raw triple data we constructed a tensor $\mathbf{X}$ of size $3,000,417 \times 3,000,417 \times 38$ and an attribute matrix D of size $3,000,417 \times 1,138,407$. The attribute matrix D has been created by tokenizing and stemming

[1] The formulas for the (B+N+C+T) model are publicly available from `http://alchemy.cs.washington.edu/mlns/er/`

[2] This data set has been obtained from `http://yago-knowledge.org`.

[3] For this task we used version `20111027` of the YAGO conversion tools, downloaded from `http://yago-knowledge.org`

Table 3.8.: Statistics for selected classes in YAGO2 core

Type	Number of entities
`wordnet:person`	884,261
`wordnet:location`	429,828
`wordnet:movie`	62,296

attribute values of textual attributes such as `rdfs:label`, `yago:hasPreferredMeaning` etc. The tensor $\mathbf{X}$ has approximately 41 million non-zero entries, while D has around 35.4 million entries. It can be seen that both $\mathbf{X}$ and D are very sparse. A tensor for the YAGO2 core ontology has 4.3×10^{14} possible entries (of which 2.4×10^{13} are valid according to `rdfs:range` and `rdfs:domain` constraints), but only 4×10^7 non-zero entries.

In the first experiments on YAGO2, the objective was to correctly predict links for the `rdf:type` predicate for various higher level classes, namely `wordnet:person`, `wordnet:location` and `wordnet:movie`. The choice of these classes is motivated by the fact that they occur on different levels in the subclass hierarchy and are of different size. Some statistics for these classes are available in table 3.8. For each of these classes we performed five-fold stratified cross-validation over all entities in the data in two different settings:

a) Only those `rdf:type` triples that include the class C that should be predicted were removed from the test fold. All other type triples, including subclasses of C, are still present in the data.

b) *All* `rdf:type` triples were deleted in the test fold.

The objective in setting a) is unusual for machine learning, since a high correlation exists between classes and their subclasses. However, it is a very common Semantic Web problem, as it corresponds to the materialization of `rdf:type` triples, given an ontology. Setting b), on the other hand is a very common machine learning problem, as it corresponds to the classification of entities, given their relations and attributes. As in chapter 2, we employed the area under the precision-recall curve (AUC-PR) to evaluate the results, due to the large skew in the distribution of existing and non-existing triples (Davis and Goadrich, 2006).

Table 3.9 shows the results of these experiments. It can be seen that RESCAL learns a reasonable model in both settings. The results for setting a) indicate that given enough support in the data, RESCAL can predict triples that originate from transitive rules such as

$$\forall xyz : \texttt{classOf}(x,y) \wedge \texttt{subClassOf}(y,z) \Rightarrow classOf(x,z)$$

Table 3.9.: Link-prediction experiments on YAGO2.

	AUC-PR		
	`wordnet:person`	`wordnet:location`	`wordnet:movie`
Random	0.32	0.18	0.06
Setting a)	0.99	1.0	0.75
Setting b)	0.96	0.98	0.51
With attributes	-	-	0.85

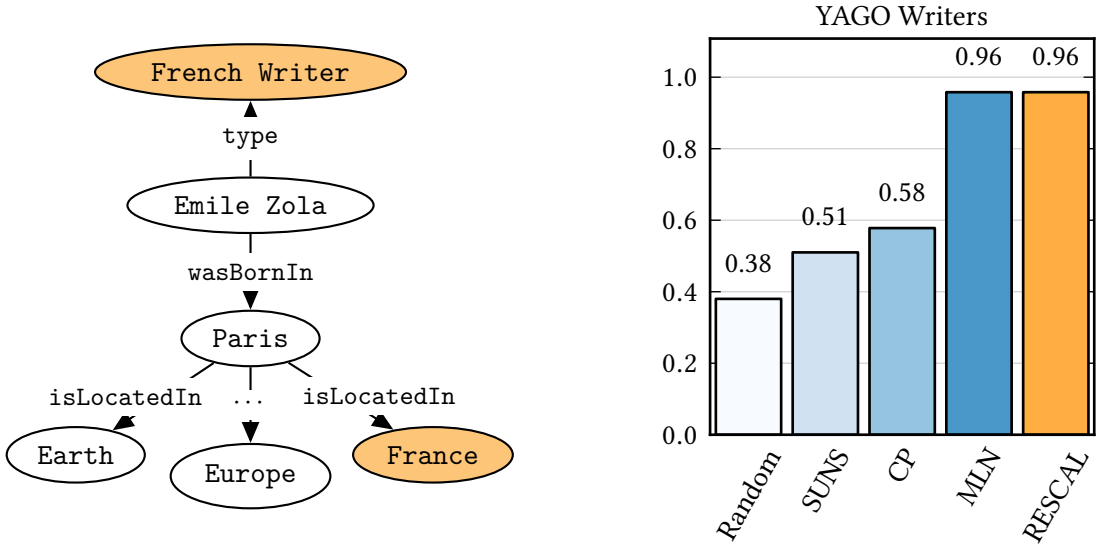
with high precision and recall. But the results for setting b) are also very encouraging, since they are close to the results of setting a). Not surprisingly, to achieve good results in setting b) it is necessary to train more complex models. For instance, the results to predict `wordnet:person` in setting a) were computed with $r = 7$, while setting b) required $r = 15$. This is consistent with the higher degree of difficulty for setting b). For persons and locations the factorization was computed without attribute information. To predict type relationships for `wordnet:movie` we performed an additional experiment, where we added values of the predicate `yago:hasWikipediaCategory` from the full YAGO2 ontology as attributes to the factorization. The rationale behind this procedure was that this predicate provides a textual description for movies that usually includes the token "films", e.g. "French comedy films", "1997 films" etc. It can be seen from the significantly improved results that RESCAL can detect these regularities by using the attribute extension, even to that extend that it surpasses the results of setting a).

3.7.4. Collective Learning on the Semantic Web

As previously stated, collective learning is an important feature for learning on Linked Data, due to relational nature of Linked Data and due to the way of how information is modeled via dyadic relations in RDF. To demonstrate the effect of collective learning for Linked Data, we carried out a specifically designed link prediction experiment on data extracted from YAGO2. In this experiment, the objective was to predict links of the kind

(`?s`, `rdf:type`, `wikicategory:?`*nationality*`_writer`),

where ?*nationality* can be any nationality such as `French` , `American` , `German` etc. and where ?*s* is any entity in the data set. The classification of a person as a writer of a particular nationality is dependent on the birthplace of the writer in question. For this purpose, the country of birth is a much better correlate than the city in which a person was born, as, for



(a) Collective learning example on YAGO. The objective is to learn the correlation between `France` and `French Writer` from examples like `Emile Zola`.

(b) Results for link prediction on YAGO2 writers data set over ten-fold cross-validation.

Figure 3.3.: YAGO2 writers collective learning example and experimental results.

instance, every French writer was born in France but not every French writer was necessarily born in Paris. However, the way knowledge is usually modeled in RDF, the country of birth is not directly connected to a particular person, but only indirectly via the person's city of birth. Figure 3.3a shows an example of this modeling. Consequently, to exploit the stronger correlation between the nationality of a writer and the writer's country of birth, collective learning is needed. To evaluate the performance of RESCAL for this task, we compared it to algorithms that were previously used for machine learning on Semantic Web data. CP has been used by Franz et al. (2009) to rank RDF data for faceted browsing, while SUNS has been applied for link prediction on RDF data (Huang et al., 2010; Huang et al., 2011). It has been shown in section 2.6 that both of these algorithms have only limited collective learning capabilities. For this reason, we also included MLNs, where we *manually*[1] specified the rule

$$\texttt{wasBornIn}(x,y) \wedge \texttt{isLocatedIn}(y,+z) \Rightarrow \texttt{type}(x,+c)$$

[1]Unfortunately, we were again not able to get reasonable results with structure learning for MLN.

and learned the weights for this rule via the Alchemy package.[1] To create a confined setting, we extracted a subgraph of the Yago ontology consisting only of American, French, German and Japanese writers, their birthplaces and their respective countries as well as the predicates `yago:wasBornIn`, `yago:isLocatedIn` and `rdf:type`. Learning only on this subgraph has the advantage that it is a very controlled setting, i.e. a relational learning algorithm should only be successful on this data when it can detect the correlation between the `rdf:type` of a person and the country of the person's birthplace. The subgraph was constructed with SPARQL queries such as

```
SELECT ?writer, ?birthPlace, ?location WHERE
{
  ?writer rdf:type wikicategory:French_writer .
  ?writer yago:wasBornIn ?birthPlace .
  ?birthPlace yago:isLocatedIn ?location
}
```

Converting the raw RDF data to a tensor representation, resulted in an adjacency tensor of size $404 \times 404 \times 3$. On this data we performed ten-fold cross-validation where the objective was to correctly predict the `rdf:type` relationships. In each iteration, all `rdf:type` triples were removed from the test data. Figure 3.3b shows the results of this experiment. It can be seen that Rescal is very successful in predicting the correct `rdf:type` triples, while CP and Suns struggle to learn something meaningful on this data, what is due to their missing collective learning ability.

3.7.5. Learning Taxonomies

In section 3.3 we briefly described how Rescal can be used to learn taxonomies on Semantic Web data. To evaluate the applicability of our approach, we conducted experiments with the objective to rebuild an existing taxonomy as closely as possible in a fully unsupervised setting, i.e. only from entity data without ontological information. For this purpose we used the large version of the IIMB 2010 benchmark provided by the Ontology Alignment Evaluation Initiative,[2] which contains around 1400 entities of a movie domain. These entities are organized in an ontology that consists of 5 distinct top-level concepts, namely `Budget`, `Creature`, `Film`, `Language`, and `Location`. The concepts `Creature`, `Film` and `Location`

[1] $+z$ and $+y$ is Alchemy-specific syntax that replaces the variable with all occurring constants.
[2] available from `http://oaei.ontologymatching.org/2010/im/index.html`

are again subdivided into multiple concepts such as `Person` and `Character`, `Anime` and `Action Movie`, or `Country` and `City`. In total, there exist 80 concepts and the maximum subclass-level is 3. A tensor representation of this data is of size $1519 \times 1519 \times 35$. As the hierarchical clustering algorithm to work in the latent space A, we selected OPTICS (Ankerst et al., 1999), which is a density-based hierarchical clustering algorithm that also provides an interpretable visual representation of its results. To evaluate the quality of our clustering, we followed the procedure suggested by Tan et al. (2006) and assigned the F-measure score to a particular concept that is the highest for this concept out of all clusterings. The idea behind this approach is that there should exist one cluster for each concept that is pure and holds most of this concept's entities. Table 3.10 shows the results of our evaluation, using a RESCAL model with $r = 10$ and an OPTICS clustering with $minpts = 1$. It can be seen that our approach

Table 3.10.: F-measure for selected concepts and weighted F-measure for all concepts per subclass-level

Level 1		**Level 2**		**Level 3**	
Locations	0.95	City	0.99	Capital	0.99
Films	1.0	Anime	0.67	Director	0.78
Creature	1.0	Character	0.73	Character Creator	0.53
Budget	1.0	Person	1.0	Actor	0.98
Language	1.0	Country	0.80		
All	0.982	**All**	0.852	**All**	0.947

achieves good results throughout all levels, especially for top-level concepts. One reason for this behaviour is that, on this level, every concept is represented by a sufficient number of entities, while e.g. some level 2 movie concepts only include two or three entities and therefore are hard to recognize. Although more sophisticated taxonomy-extraction methods could be applied, the results are very encouraging. Moreover, it is shown once again that the latent space A provides important information about a relational domain, even for such difficult tasks like hierarchical link-based clustering.

3.8. Discussion and Related Work

In this section, we will put the scalability of RESCAL-ALS into the context of related tensor factorizations and also discuss machine learning methods that have been applied to Semantic Web data.

3.8.1. Comparison to Tensor Decompositions

The complexity analysis of RESCAL-ALS in sections 3.4 and 3.5 showed its high scalability with regard to the size of relational data. Surprisingly, it is difficult to find detailed results on the computational complexity for (sparse) tensor decompositions such as CP and TUCKER. For this reason, we provide a brief analysis of state-of-the-art algorithms to compute CP and TUCKER in appendix C, in which we highlight important differences to the complexity of RESCAL-ALS. It can be seen that none of these algorithms scale linearly with the size of relational data, having at least quadratic complexity with the number of entities and with the number of predicates in a data set. The analysis in appendix C as well as the discussion in sections 3.4 and 3.5 suggest that multiple factors are important for the scalability of RESCAL-ALS: It has been shown in section 3.4 that sparse linear algebra operations can reduce the runtime complexity significantly. Similar improvements can also be applied to CP and TUCKER. However, only in combination with the following properties it can take its full effect: Factorizations such as CP and TUCKER-3, which factorize all modes of an adjacency tensor become quadratic in the number of entities when factorizing the third mode. RESCAL avoids this increased complexity via its TUCKER-2 structure, where the third mode of an adjacency tensor is not factorized. Consequently, the redundancies between relations, which form the third mode in an adjacency tensor, are not explicitly captured. Since we are not concerned with the analysis of relations in this thesis, this is a good compromise for the increased scalability. In comparison to DEDICOM, RESCAL achieves higher scalability through its simpler core structure. To the best of our knowledge, ASALSAN is currently considered to be the most efficient algorithm to compute the DEDICOM factorization on large sparse data (Bader et al., 2007). To compute updates for the core matrix R in ASALSAN, it is required to compute the term

$$\operatorname{vec}(R) \leftarrow \left(\sum_{k=1}^{m} (D_k A^T A D_k) \otimes (D_k A^T A D_k) \right)^{-1} \sum_{k=1}^{m} \operatorname{vec}\left(D_k A^T X_k A D_k \right) .$$

Since the matrices $D_k A^T A D_k \otimes D_k A^T A D_k$ are of size $r^2 \times r^2$, the matrix inversion alone would already have a runtime complexity of $O(r^5)$. Furthermore, since the sum inside the matrix inverse runs over multiple matrices, it is also not clear how this complexity could be reduced. Hence, the simpler core structure of RESCAL does not only allow for better relational models as discussed and shown experimentally in sections 2.6 and 2.7, but it also reduces the computational complexity significantly. Another important factor for the scalability of RESCAL is that it does not put orthogonality constraints on the latent factors, as

required by many matrix and tensor factorizations. This constraint would lead to a quadratic computational and a quadratic memory complexity with regard to the number of entities for a Tucker-type factorization such as Rescal, when using state-of-the-art methods to compute the factorization, as discussed in appendix C. By dropping this constraint and employing normal equations to compute updates of latent factors, we gain linear computational and linear memory complexity with regard to the number of entities at the expense of a non-minimal set of latent components. Since we are not concerned with the direct interpretation of the latent components in this thesis, this is again a good compromise for the increased scalability. In summary, the combination of sparse linear algebra with a normal-equations based algorithm and the Tucker-2 structure of Rescal enables its high scalability on relational data.

3.8.2. Comparison to Semantic Web Machine Learning Methods

To the best of our knowledge, there have yet not been any attempts to apply machine learning methods which compute a *full relational model* to Linked Data of the size considered in this chapter. In the context of the Semantic Web, Inductive Logic Programming and kernel learning have been the dominant approaches to machine learning so far (Bloehdorn and Sure, 2007; d'Amato et al., 2008; Fanizzi et al., 2008). Further approaches to learn from Semantic Web data include the work by Lin et al. (2011), which proposed to learn Relational Bayesian Classifiers for RDF data via queries to a Sparql endpoint. Sparql-ML (Kiefer et al., 2008) extends Sparql queries to support data mining constructs. Bicer et al. (2011) employ a coevolution-based genetic algorithm to learn kernels for RDF data. Recently, methods such as association rule mining and knowledge base fragment extraction have been applied to large Semantic Web databases for tasks like schema induction and learning complex class descriptions (Voelker and Niepert, 2011; Hellmann et al., 2009). However, all these methods either do not scale to large data sizes or do not compute a full relational model of a complete knowledge base. Huang et al. (2011) proposed Suns, a regularized matrix factorization to predict unknown triples in Semantic Web data. It has been shown in sections 2.6 and 3.7 that Rescal can outperform this approach significantly on data where collective learning is important. Probably most similar to our approach are TripleRank (Franz et al., 2009) and TopHits (Kolda et al., 2005), which employ the CP decomposition for learning from Semantic Web data and multigraphs. The limited scalability and collective learning ability of CP and CP-ALS compared to Rescal and Rescal-ALS translates therefore also to these models.

3.9. Summary

In this chapter, we demonstrated that tensor factorization in form of the RESCAL decomposition is a suitable approach for relational learning from Linked Data and showed that the proposed approach can scale to large knowledge bases. We showed via a thorough analysis of the computational and of the memory complexity of RESCAL-ALS that a sparse implementation scales linearly with the size of relational data. Furthermore, we derived a very efficient way to compute updates of the core tensor within RESCAL-ALS, which decreases the runtime complexity with regard to the number of latent components from $O(r^5)$ to $O(r^3)$ and the memory complexity from $O(r^4)$ to $O(r^2)$. To handle attributes of entities efficiently, we proposed coupled tensor-matrix factorization, which improves the scalability of the algorithm with regard to the number of attribute values in the data. We reassessed the theoretical analysis with experiments on synthetic data, where we showed that RESCAL-ALS scales linearly with the number of entities, predicates, and know facts in a relational data set and can factorize data consisting of millions of entities, hundreds of predicates and billions of known facts. Furthermore, we showed that our approach is able to factorize the YAGO2 core ontology and predict the types of entities for various higher-level classes in this large knowledge base. Experimentally, we also demonstrated the effectiveness of the RESCAL model for difficult tasks that are important to Linked Data such as collective link prediction and taxonomy learning. In its present form, a limitation of RESCAL-ALS is its cubic runtime complexity with regard to the number of latent components. While the algorithm is certainly scalable enough to handle large knowledge bases that require only a few hundred of latent components, a full relational model for very complex knowledge bases is currently out of reach. An interesting direction for future work is therefore to investigate if the inclusion of prior knowledge in the factorization can reduce the number of latent components that is needed to model complex knowledge bases. Despite this current limitation, we think that the proposed method opens up an interesting perspective towards learning from complete knowledge bases in the Linked Data cloud.

Chapter 4

An Analysis of Tensor Models for Learning on Structured Data

While tensor factorizations have become increasingly popular for learning on various forms of structured data, only very few theoretical results exist on the generalization abilities of these methods. In this chapter, we will discuss the tensor product as a principled way to represent structured data in vector spaces for machine learning tasks. By extending known bounds for matrix factorizations, we will derive the first known generalization error bounds for tensor factorizations in a classification setting. This setting subsumes also link prediction on multi-relational data consisting of n-ary relations. Furthermore, we will analyze analytically and experimentally how tensor factorization behaves when applied to over- and understructured representations, for instance, when two-way tensor factorization, i.e. matrix factorization, is applied to three-way tensor data.

4.1. Introduction

Learning from structured data is a very active line of research in a variety of fields, including social network analysis, natural language processing, bioinformatics, and artificial intelligence. While tensor factorizations have a long tradition in psycho- and chemometrics, only more recently have they been applied to various tasks on structured data in machine learning. Examples include link prediction and entity resolution on multi-relational data and large knowledge bases as discussed in this thesis or in (Jenatton et al., 2012; Bordes et al., 2011), item recommendation on sequential data (Rendle et al., 2010; Rettinger et al., 2012), or the

analysis of time varying social networks (Bader et al., 2007) – only to name a few examples. A reason for the success of tensor methods in these tasks is their very appealing property to efficiently impose structure on the vector space representation of data. Moreover, tensor factorizations can be related to multilinear models, which overcome some limitations of linear models, such as their limited expressiveness, but at the same time remain more scalable and easier to handle than non-linear approaches. However, despite their increasing popularity and their appealing properties, only very few theoretical results exist on the generalization abilities of tensor factorizations. Furthermore, an important open question is what kind of generalization improvements over simpler, less structured models can be expected. For instance, *propositionalization*, which transforms relational data into feature-based representations, has been considered as a means of relational learning (Kramer et al., 2001; Huang et al., 2011). In terms of tensor factorization, propositionalization would be equivalent to transforming a tensor into a matrix representation prior to computing the factorization. While it has been shown empirically that tensor methods usually scale better with the amount of missing data than their matrix counterparts (Tomioka et al., 2012; J. Liu et al., 2009; Tomioka et al., 2010; Signoretto et al., 2011) and that they can yield significantly improved results over "flat" methods which ignore a large part of the data structure as shown in chapter 2, no theoretical justification of this behavior is known in terms of generalization bounds.

In this chapter, we will approach several of these open questions. First, we will briefly discuss the tensor product as a principled way to derive vector space representations of structured data. Subsequently, we will present the first generalization error bounds of tensor factorizations for classification tasks. We will analyze experimentally the effect of imposing structure on vector space representations via the tensor product as well as the effect of constraints that are applied to popular tensor decompositions. Based on the newly derived bounds we discuss how these results can be interpreted analytically.

4.2. Theory and Methods

In this section we will discuss how structured data can be modeled as weighted sets of n-tuples. Furthermore, we will show how this modeling enables a closer analysis of the relations between tensor factorizations and structured data in the following sections of this chapter.

4.2.1. Structured Data, the Cartesian, and the Tensor Product

To analyze the relation between the order of a tensor and the "structuredness" of data representation we introduce the concept of the *order of structured data.* The general framework in which we will describe structured data is in form of sets of weighted m-tuples, which are defined as follows:

Definition 9 (Set of Weighted m-Tuples)**.** *Let $\mathcal{V} = \mathcal{V}^{(1)} \times \cdots \times \mathcal{V}^{(m)}$ be the Cartesian product over m sets $\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(m)}$ and let $\phi : \mathcal{E} \mapsto \mathbb{R}$ be a real-valued function that assigns a weight to each m-tuple in $\mathcal{E} \subseteq \mathcal{V}$. A set of weighted m-tuples $\mathcal{T}$ is then defined as a 4-tuple $(\mathcal{V}, \mathcal{E}, \phi, m)$. The* order *of $\mathcal{T}$ is defined as the length of its tuples m. For conciseness, we will refer to sets of weighted m-tuples also as* weighted tuple-sets.

Weighted tuple-sets can be interpreted in the following way: The elements of the sets $\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(m)}$ correspond to the constituents of the structured data. The set $\mathcal{E}$ corresponds to the observed m-tuples, while $\mathcal{V}$ corresponds to all possible m-tuples. For a tuple $t \in \mathcal{E}$, the pair $(t, \phi(t))$ corresponds to an observed data point. This is a very general form of data representation that allows us to consider many forms of structured data. For instance, *dyadic multi-relational* data – as considered in previous chapters – has a natural representation as a weighted tuple-set, where $\mathcal{V}^{(e)}$ is the set of all entities in the data, $\mathcal{V}^{(p)}$ is the set of all predicates, and the weight function $\phi : \mathcal{V}^{(p)} \times \mathcal{V}^{(e)} \times \mathcal{V}^{(e)} \mapsto \{\pm 1\}$ is defined as[1]

$$\phi(p_i, e_j, e_k) = \begin{cases} +1, & \text{if the relationship } p_i(e_j, e_k) \text{ exits} \\ -1, & \text{otherwise.} \end{cases}.$$

Furthermore, *sequential* or *time-varying* data can be modeled via m-tuples such as (user, item, last item) triples for item recommendation (Rendle et al., 2010) or (person, person, month) triples in time-varying social networks (Bader et al., 2007). In these cases, the function ϕ could model the rating of a product or the interaction of persons. Traditional *attribute-value data*, as it is common in many machine learning applications, can be modeled via (object, attribute) pairs, which are weighted by the respective attribute values, e.g. $\phi(Anne, age) = 36$.

Tuple-sets can be modeled very naturally using tensors in the following way: Let $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \phi, m)$ be a weighted tuple-set and let $I^{(i)}$ be the standard basis of dimension $|\mathcal{V}^{(i)}|$, such that it indexes all elements of $\mathcal{V}^{(i)}$. $\mathcal{T}$ can then be modeled as a tensor $\mathbf{Y} \in \bigotimes_{i=1}^{m} I^{(i)}$ with

[1] Please note that in this chapter we use the set $\{\pm 1\}$ to denote existing and non-existing relations instead of the set $\{0, 1\}$, which has been used in previous chapters. This is due to formal reasons, as the following generalization bounds will be based on the number of sign patterns that a factorization can express.

entries $y_{i_1,\dots,i_m} = \phi(v_{i_1},\dots,v_{i_m})$ for all observed tuples $(v_{i_1},\dots,v_{i_m}) \in \mathcal{E}$. For unobserved tuples $(v_{i_1},\dots,v_{i_m}) \in \mathcal{V} \setminus \mathcal{E}$, the corresponding entries in $\mathbf{Y}$ are modeled as missing. Using this construction, each set of objects $\mathcal{V}^{(i)}$ is indexed separately by a mode of the tensor $\mathbf{Y}$. Therefore, it holds that the order of the tensor $\mathbf{Y}$ is identical to the order of the weighted tuple-set $\mathcal{T}$. This enables us to rephrase the question how the structuring of a vector space representation affects the generalization ability of a factorization in terms of the order of weighted tuple-sets and the order of tensors. In particular we are interested in how the generalization ability changes for a tensor representation that has *not* the same order as the underlying weighted tuple-set – compared to a tensor representation that has the identical order.

In this work, we will only consider the problem of learning from sets of binary-weighted tuples, i.e. tuple-sets with weight functions of the form $\phi : \mathcal{E} \mapsto \{\pm 1\}$. This corresponds to a classification setting on binary tensors where $y_{\boldsymbol{i}} \in \{\pm 1\}$ indicates the presence or absence of an m-tuple. Please note that this setting subsumes also the link prediction task as discussed in earlier chapters.

4.2.2. Tensor Factorizations

In the remainder of this chapter, we will analyze the generalization ability of tensor factorization via the Tucker decomposition. For this purpose, we will first extend the definition of the Tucker decomposition in section 1.3.3 to tensors of arbitrary order. Furthermore, we will discuss how different factorization methods can be expressed within this framework through additional constraints. The Tucker decomposition of an m-th order tensor is defined as follows:

Definition 10 (Tucker Decomposition)**.** *Let* $\mathbf{Y} \in \mathbb{R}^{\prod_i n_i}$ *be a tensor with* $\mathrm{ord}(\mathbf{Y}) = m$*. The* Tucker *decomposition with n-rank* $(r_1,\dots,r_m)$ *factorizes* $\mathbf{Y}$ *such that each entry of* $\mathbf{Y}$ *is described by the multilinear polynomial*

$$y_{i_1,i_2,\dots,i_m} \approx \sum_{j_1=1}^{r_1} \sum_{j_2=1}^{r_2} \cdots \sum_{j_m=1}^{r_m} w_{j_1,j_2,\dots,j_m} \prod_{k=1}^{m} u^{(k)}_{i_k,j_k}. \tag{4.1}$$

We can now make the connection between the Tucker decomposition of a tensor and weighted tuple-sets as defined in definition 9: the factorization equation (4.1) can be interpreted as learning a multilinear function $\gamma : \mathcal{V}^{(1)} \times \cdots \times \mathcal{V}^{(m)} \mapsto \mathbb{R}$ which maps m-tuples to the entries of $\mathbf{Y}$. In contrast to the weight function ϕ of a tuple set, γ is defined over the whole Cartesian product $\mathcal{V}^{(1)} \times \cdots \times \mathcal{V}^{(m)}$.

In the following, it will prove convenient to state equation (4.1) in different notations. In tensor notation, equation (4.1) is equivalent to

$$\mathbf{Y} \approx \mathbf{X} = \mathbf{W} \times_1 U^{(1)} \times_2 \cdots \times_m U^{(m)}. \tag{4.2}$$

where $\times_k$ denotes the n-mode product of a tensor and a matrix in mode k as defined in section 1.3.2. The matrix $U^{(k)} \in \mathbb{R}^{n_k \times r_k}$ denotes the latent factor matrix for mode k, while $\mathbf{W} \in \mathbb{R}^{r_1 \times \ldots \times r_m}$ is the core tensor of the factorization. Furthermore, via the *unfolding* operation on tensors and the Kronecker product, equation (4.2) can be stated in matrix notation as

$$Y_{(k)} \approx U^{(k)} W_{(k)} \left(U^{(m)} \otimes \cdots \otimes U^{(k+1)} \otimes U^{(k-1)} \otimes \cdots \otimes U^{(1)} \right)^T. \tag{4.3}$$

We will also shorten n-rank$(\mathbf{Y}) = (r_1, \ldots, r_m)$ to n-rank$(\mathbf{Y}) = \boldsymbol{r}$. At last, we define some quantities associated with the Tucker decomposition that will prove convenient for the rest of this chapter.

Definition 11. *Let* $\mathbf{X} = \mathbf{W} \times_1 U^{(1)} \times_2 \cdots \times_m U^{(m)}$ *with* n-rank$(\mathbf{X}) = \boldsymbol{r}$, $m = \text{ord}(\mathbf{X})$ *and* $\mathbf{X} \in \mathbb{R}^{\prod_i n_i}$. *The number of variables of a Tucker decomposition, i.e. the number of entries in the latent factors, is then given by*

$$\text{var}(\mathbf{X}) = \prod_{i=1}^{m} r_i + \sum_{i=1}^{m} n_i r_i.$$

The number of polynomials associated with $\mathbf{X}$*, i.e. the number of entries in* $\mathbf{X}$*, is denoted by*

$$\text{pol}(\mathbf{X}) = \prod_{i=1}^{m} n_i.$$

By applying specific constraints on the core tensor or the latent factors, various important factorization methods can be expressed as special cases within the Tucker decomposition framework. One focus of this chapter is to analyze how these constraints affect the generalization ability of a factorization. In the following, we will briefly discuss some important models to illustrate these constraints: Most matrix factorization methods, can be considered a Tucker decomposition of a second-order tensor. For instance, the *singular value decomposition* can be expressed as a Tucker decomposition of a second order tensor with orthogonal factor matrices. Furthermore, CP can be described as a Tucker decomposition with the additional constraints that the core tensor $\mathbf{W}$ is superdiagonal and $r_1 = r_2 = \cdots = r_m$. Similarly, the *Block-Term decomposition* (BTD) (De Lathauwer, 2008) can be viewed as imposing the con-

straint that the core tensor $\mathbf{W}$ is block-diagonal. While CP and BTD are decompositions that put special constraints on the core tensor, RESCAL, as discussed in chapter 2, is a factorization that constrains the number of different vector spaces under consideration. Specifically, it requires that some of the latent factors are identical, which corresponds to the fact that for some sets $\mathcal{V}^{(i)}, \mathcal{V}^{(j)}$ of the underlying tuple-set, it holds that $\mathcal{V}^{(i)} = \mathcal{V}^{(j)}$.

4.3. Generalization Bounds for Low-Rank Factorizations

To get deeper theoretical insight into the generalization ability of tensor factorizations, we will now present generalization error bounds. In section 4.3.1 and section 4.3.2 we will derive generalization error bounds for the zero-one loss and real-valued loss functions respectively. These bounds will be based on the number of different sign patterns that a tensor factorization can express. In these sections, we will closely follow the theory developed by Srebro et al. (Srebro, 2004; Srebro et al., 2005) and extend it to the general multilinear setting. The actual upper and lower bounds on the number of sign patterns that a factorization can express are then given in section 4.3.3. To derive these bounds, we will employ properties of the tensor product as discussed in section 1.3.1.

Consider the following setting: Let $\mathbf{Y}$ be the tensor representation of structured data $\mathcal{T}$, where a subset of entries $y_{\boldsymbol{i}}$ has been observed and let the set $\Omega = \{\boldsymbol{i} \,|\, y_{\boldsymbol{i}} \text{ observed}\}$ hold the indices of these observed entries. Then, we seek to predict the missing entries in $\mathbf{Y}$, by computing a factorization such that

$$\mathbf{Y} \approx \mathbf{X} = \mathbf{W} \times_1 U^{(1)} \times_2 \cdots \times_m U^{(m)}.$$

Similar to the matrix case (Srebro, 2004), we now seek to bound the true discrepancy between the predicted tensor $\mathbf{X}$ and the target tensor $\mathbf{Y}$ as a function of the discrepancy of the observed entries Ω of $\mathbf{Y}$. The discrepancy of tensors is defined relative to a specific loss function $\Delta(\,\cdot\,;\,\cdot\,)$. The *true discrepancy* of a predicted tensor $\mathbf{X}$ and a target tensor $\mathbf{Y}$ with $\operatorname{ord}(\mathbf{X}) = \operatorname{ord}(\mathbf{Y}) = m$ is defined as

$$\mathcal{D}(\mathbf{X}, \mathbf{Y}) = \frac{1}{\prod_{i=1}^{m} n_i} \sum_{i_1=1}^{n_1} \sum_{i_2=1}^{n_2} \cdots \sum_{i_m=1}^{n_m} \Delta(x_{i_1,\ldots i_m}; y_{i_1,\ldots,i_m}),$$

while the *empirical discrepancy* is given as

$$\mathcal{D}_\Omega(\mathbf{X}, \mathbf{Y}) = \frac{1}{|\Omega|} \sum_{\boldsymbol{i} \in \Omega} \Delta(x_{\boldsymbol{i}}; y_{\boldsymbol{i}}).$$

We restrict the latent tensor $\mathbf{X}$ to the class of *fixed* n-rank tensors of a given order, which will

be denoted by

$$\mathcal{X}_{\boldsymbol{r}} := \{\mathbf{X} \mid \text{n-rank}(\mathbf{X}) \leq \boldsymbol{r}\}$$

Please note that by restricting the factorization to a Tucker-type decomposition and by fixing n-rank($\mathbf{X}$) = $\boldsymbol{r}$, we also fix the quantity var($\mathbf{X}$), while ord($\mathbf{X}$) and pol($\mathbf{X}$) are already determined by the target tensor $\mathbf{Y}$. We now seek to derive PAC-type error bounds of the form

$$\forall \mathbf{Y} \in \mathbb{R}^{\prod \boldsymbol{n}} : \Pr_{\Omega} \left(\forall \mathbf{X} \in \mathcal{X}_{\boldsymbol{r}} : \mathcal{D}(\mathbf{X}, \mathbf{Y}) \leq \mathcal{D}_{\Omega}(\mathbf{X}, \mathbf{Y}) + \varepsilon \right) > 1 - \delta \tag{4.4}$$

such that the true discrepancy for all tensors in $\mathcal{X}_{\boldsymbol{r}}$ is bounded by their discrepancy on the observed entries Ω plus a second term ε. An important assumption that will be made is that the set of observed entries Ω is chosen uniformly at random.

4.3.1. Bounds for Zero-One Sign Agreement Loss

A reasonable choice for $\Delta(\cdot;\cdot)$ in a classification setting is the zero-one loss, i.e.

$$\Delta(a; b) = \begin{cases} 0, & \text{if } \operatorname{sgn}(a) = \operatorname{sgn}(b) \\ 1, & \text{otherwise..} \end{cases}$$

For target entries $y_{\boldsymbol{i}} \in \{\pm 1\}$, the zero-one loss $\Delta(x_{\boldsymbol{i}}; y_{\boldsymbol{i}})$ is independent of the magnitude of the predictions $x_{\boldsymbol{i}}$ and only depends on their sign. A central concept in the following discussion will therefore be the equivalence classes of tensors with identical sign patterns, i.e. the elements of the set

$$\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}} = \left\{ \operatorname{sgn}(\mathbf{X}) \in \{-1, 0, +1\}^{\prod \boldsymbol{n}} \,\middle|\, \mathbf{X} \in \mathbb{R}^{\prod \boldsymbol{n}}, \text{n-rank}(\mathbf{X}) \leq \boldsymbol{r} \right\}.$$

The cardinality $|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}|$ specifies therefore, how many different sign patterns can be expressed by factorizations with n-rank($\mathbf{X}$) $\leq \boldsymbol{r}$ and pol($\mathbf{X}$) = $\prod \boldsymbol{n}$.

Lemma 1. *Let $\mathbf{Y} \in \{\pm 1\}^{\prod \boldsymbol{n}}$ be any binary tensor with $n_i > 2$. Furthermore, let Ω be a set of $|\Omega|$ uniformly chosen entries of $\mathbf{Y}$, let $\delta > 0$, and let $\boldsymbol{r} \in \mathbb{N}_{+}^{\text{ord}(\mathbf{Y})}$. Then, it holds with probability at least $1 - \delta$ that*

$$\forall \mathbf{X} \in \mathcal{X}_{\boldsymbol{r}} : \mathcal{D}(\mathbf{X}, \mathbf{Y}) < \mathcal{D}_{\Omega}(\mathbf{X}, \mathbf{Y}) + \sqrt{\frac{\log |\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}| - \log \delta}{2|\Omega|}}$$

where $|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}| \leq \left(\frac{4e(\text{ord}(\mathbf{X})+1)\,\text{pol}(\mathbf{X})}{\text{var}(\mathbf{X})} \right)^{\text{var}(\mathbf{X})}$

Proof. The following proof is analogue to the matrix case (Srebro et al., 2005), hence we will

only provide a brief outline. First, we fix $\mathbf{Y}$ and $\mathbf{X}$. For an index $\boldsymbol{i}$, chosen uniformly at random, it holds that $\Delta(x_{\boldsymbol{i}}; y_{\boldsymbol{i}}) \sim \mathit{Bernoulli}(\mathcal{D}(\mathbf{X}, \mathbf{Y}))$. For independently and uniformly chosen observed entries, the sum of Bernoulli distributed random variables $|\Omega|\mathcal{D}_\Omega(\mathbf{X}, \mathbf{Y})$ follows a binomial distribution with mean $|\Omega|\mathcal{D}(\mathbf{X}, \mathbf{Y})$. It follows from Chernoff's inequality that

$$\Pr_{\Omega}\left(\mathcal{D}(\mathbf{X}, \mathbf{Y}) \geq \mathcal{D}_\Omega(\mathbf{X}, \mathbf{Y}) + \varepsilon\right) \leq \exp\left(-2|\Omega|\varepsilon^2\right)$$

Furthermore, since $\Delta(x_{\boldsymbol{i}}; y_{\boldsymbol{i}})$ depends only on the sign of $x_{\boldsymbol{i}}$, the random variable $\mathcal{D}_\Omega(\mathbf{X}, \mathbf{Y})$ is identical for all tensors $\mathbf{X}$ in the same equivalence class of sign patterns. Since there exist $|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}|$ different equivalence classes, lemma 1 follows by taking a union bound of the events $\mathcal{D}(\mathbf{X}, \mathbf{Y}) \geq \mathcal{D}_\Omega(\mathbf{X}, \mathbf{Y}) + \varepsilon$ for these random variables. The actual bound on $|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}|$ is deferred until section 4.3.3. □

4.3.2. Bounds for Real-Valued Loss Functions

Before deriving upper and lower bounds for the number of sign patterns, we also provide a bound for real-valued loss functions, which is the more commonly used setting for tensor factorizations. However, these loss functions, and therefore also their associated discrepancies, are not only determined by the sign of an entry $x_{\boldsymbol{i}}$ but are also determined by the value of this entry. We will therefore derive bounds for the pseudodimension of low-rank tensors.

Lemma 2. *Let $\mathbf{Y} \in \{\pm 1\}^{\boldsymbol{n}}$ be any binary tensor with $n_i > 2$. Furthermore, let $|\Delta(\cdot;\cdot)| \leq b$ be a bounded monotone loss function, let Ω be a set of $|\Omega|$ uniformly chosen entries of $\mathbf{Y}$, let $\delta > 0$, and let $\boldsymbol{r} \in \mathbb{N}_+^{\mathrm{ord}(\mathbf{Y})}$. Then, it holds with probability at least $1 - \delta$*

$$\forall \mathbf{X} \in \mathcal{X}_{\boldsymbol{r}} : \mathcal{D}(\mathbf{X}, \mathbf{Y}) < \mathcal{D}_\Omega(\mathbf{X}, \mathbf{Y}) + \sqrt{32 \frac{\log|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r},\mathbf{T}}| \log \frac{b|\Omega|}{\mathrm{var}(\mathbf{X})} - \log\delta}{|\Omega|}}$$

Proof. Again, the following proof is analogue to the matrix case (Srebro et al., 2005), hence we will outline it only briefly. As mentioned in section 4.2.2, tensor factorizations can be interpreted as real-valued functions, which map from tuples of indices to entries of the tensor, i.e. a multilinear function $\gamma : \mathcal{V}^{(1)} \times \cdots \times \mathcal{V}^{(m)} \mapsto \mathbb{R}$, where $\mathcal{V}^{(i)}$ indexes the i-th mode. This allows to use the pseudodimension of classes of real-valued functions to obtain similar generalization error bounds as for matrices. The difference to the matrix case is that for tensors the domain of the function γ ranges of tuples of fixed length m, while for matrices it ranges over ordered pairs. Therefore, we first bound the pseudodimension of n-rank tensors via the number of sign patterns *relative to a threshold tensor* $\mathbf{T} \in \mathbb{R}^{\prod \boldsymbol{n}}$. The equivalence

classes for these relative sign patterns are given by the set

$$\mathcal{S}_{\boldsymbol{n},\boldsymbol{r},\mathbf{T}} = \left\{ \operatorname{sgn}(\mathbf{X} - \mathbf{T}) \in \{-1, 0, +1\}^{\prod \boldsymbol{n}} \,\middle|\, \mathbf{X} \in \mathbb{R}^{\prod \boldsymbol{n}}, \text{n-rank}(\mathbf{X}) \leq \boldsymbol{r} \right\}.$$

The concrete bound for $|S_{\boldsymbol{n},\boldsymbol{r},\mathbf{T}}|$ will be given in section 4.3.3. Using (Srebro, 2004, Theorem 44) we can then obtain the desired bound. □

4.3.3. Bounds on the Number of Sign Patterns

Following the discussion in section 4.3.1 and section 4.3.2, we now seek to bound the number of possible sign patterns $|\mathcal{S}_{n,r}|$ and the number of relative sign patterns $|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r},\mathbf{T}}|$ for tensors $\mathbf{X} \in \mathcal{X}_{\boldsymbol{r}}$. For this purpose, consider the polynomial form of the Tucker decompositions as given in equation (4.1). Due to the multilinearity of tensor factorizations, the degree of the polynomial in equation (4.1) is equal to $\operatorname{ord}(\mathbf{X}) + 1$. Furthermore, for tensors of fixed size and n-rank, the quantities $\operatorname{pol}(\mathbf{X})$ and $\operatorname{var}(\mathbf{X})$ are also fixed. Using this property of multilinear factorizations, we can bound the number of possible sign patterns of tensors with $\text{n-rank}(\mathbf{X}) = \boldsymbol{r}$ by using their polynomial representation. Following Warren (1968) it has been shown, that the number of possible sign patterns for polynomials are bounded by

Theorem 4 (Srebro, 2004, Theorem 34, 35)**.** *The number of sign patterns of m polynomials, each of degree at most d, over q variables is at most*

$$\left(\frac{4edm}{q}\right)^q$$

for all $m > q > 2$.

By combining the polynomial form of tensor factorizations equation (4.1) and theorem 4, we can immediately derive the following lemma which bounds the number of possible sign patterns for n-rank tensors.

Lemma 3 (Upper Bound for Sign Patterns)**.** *The number of possible sign patterns of a m-th order tensor $\mathbf{X} \in \mathbb{R}^{\prod \boldsymbol{n}} = \mathbf{W} \times_1 U^{(1)} \times_2 \cdots \times_m U^{(m)}$ with $\text{n-rank}(\mathbf{X}) = \boldsymbol{r}$ is at most*

$$|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}| \leq \left(\frac{4e\,(\operatorname{ord}(\mathbf{X}) + 1)\,\operatorname{pol}(\mathbf{X})}{\operatorname{var}(\mathbf{X})} \right)^{\operatorname{var}(\mathbf{X})}$$

for $\operatorname{pol}(\mathbf{X}) > \operatorname{var}(\mathbf{X}) > 2$.

Furthermore, the number of relative sign patterns, i.e. $|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r},\mathbf{T}}|$, can be bounded in the same

way, since for

$$y_{i_1,\ldots,i_m} - t_{i_1\ldots,i_m} = \sum_{j_1=1}^{r_1}\sum_{j_2=1}^{r_2}\cdots\sum_{j_m=1}^{r_m} w_{j_1,\ldots,j_m}\prod_{k=1}^{m} u^{(k)}_{i_k j_k} - t_{i_1\ldots,i_m}$$

we have again $\mathrm{pol}(\mathbf{X})$ polynomials of degree $\mathrm{ord}(\mathbf{X}) + 1$ over $\mathrm{var}(\mathbf{X})$ variables.

Next, we provide a lower bound on the number of sign patterns, by interpreting tensor factorization as multiple simultaneous linear classifications.

Lemma 4 (Lower Bound for Sign Patterns)**.** *The number of possible sign patterns of a m-th order tensor* $\mathbf{X} \in \mathbb{R}^{\prod \boldsymbol{n}} = \mathbf{W} \times_1 U^{(1)} \times_2 \cdots \times_m U^{(m)}$ *with* $\text{n-rank}(\mathbf{X}) = \boldsymbol{r}$ *is at least*

$$|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}| \geq \left(\frac{n_i}{r_i - 1}\right)^{\frac{1}{n_i}(r_i-1)\,\mathrm{pol}(\mathbf{X})}$$

Proof. First, consider the Tucker decomposition in its unfolded variant, i.e.

$$X_{(i)} = U^{(i)} W_{(i)} \left(U^{(m)} \otimes \cdots \otimes U^{(i+1)} \otimes U^{(i-1)} \otimes \cdots \otimes U^{(1)}\right)^T$$

Let $B = U^{(m)} \otimes \cdots \otimes U^{(i+1)} \otimes U^{(i-1)} \otimes \cdots \otimes U^{(1)} \in \mathbb{R}^{\prod \boldsymbol{n}/n_i \times \prod \boldsymbol{r}/r_i}$, and fix $U^{(k)} \in \mathbb{R}^{n_k \times r_k}$ with rows in general position for all $k = 1 \ldots m$. We now consider the number of possible sign patterns of matrices $U^{(i)} W_{(i)} B^T$. It follows from the rows being in general position that $\mathrm{rank}\left(U^{(k)}\right) = r_k$ for all $k = 1 \ldots m$ (Hassoun, 1995, Sec. 1.3.2). Furthermore, since the tensor product preserves the linear independence of vectors, it follows that $\mathrm{span}(B) = \mathbb{R}^{\prod \boldsymbol{r}/r_i}$ (Anthony and Harvey, 2012, Sec. 6.1.4). Although B is highly structured, it follows that the matrix product $W_{(i)} B^T$ varies over all possible $r_i \times \prod \boldsymbol{n}/n_i$ matrices. Therefore, each column of $\mathrm{sgn}(U^{(i)} W_{(i)} B^T)$ can be considered an independent homogeneous linear classification of n_i vectors in $\mathbb{R}^{r_i}$, for which exactly

$$2\sum_{k=0}^{r_i-1}\binom{n_i}{k} > \left(\frac{n_i}{r_i - 1}\right)^{r_i-1}$$

such classifications exists. Consequently, this many sign patterns exist for each of the $\prod \boldsymbol{n}/n_i = \mathrm{pol}(\mathbf{X})/n_i$ columns of $U^{(i)} W_{(i)} B^T$. □

Next we analyze the tightness of bounds in lemma 3 and lemma 4. Let $m = \mathrm{ord}(\mathbf{X})$, let $\alpha = 4e(m+1)$, let $\forall i : r_{min} \leq r_i$, and similarly let $\forall i : n_{max} \geq n_i$. Then, for $r_{min} \geq \alpha^{1/m}$ it follows from lemma 3 that

$$|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}| \leq \left(\frac{\alpha n_{max}^m}{r_{min}^m}\right)^{\mathrm{var}(\mathbf{X})} \leq \left(\frac{\alpha^{1/m} n_{max}}{r_{min}}\right)^{m\ \mathrm{var}(\mathbf{X})} \leq n_{max}^{m\ \mathrm{var}(\mathbf{X})}$$

Furthermore, for low-rank factorizations with $n_i > r_i^2$ and $\text{pol}(\mathbf{X}) > \frac{m}{r_i - 1} \text{var}(\mathbf{X})$ it follows from lemma 4 that

$$|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}| \geq \left(\frac{n_i}{r_i - 1}\right)^{\frac{1}{n_i}(r_i-1)\,\text{pol}(\mathbf{X})} \geq \sqrt{n_i}^{\frac{1}{n_i}(r_i-1)\,\text{pol}(\mathbf{X})} \geq n_i^{\frac{1}{2n_i} m\,\text{var}(\mathbf{X})}$$

Hence, the bound is tight up to a multiplicative factor in the exponent.

4.4. The Effect of Structure and Constraints

In section 4.3 we derived bounds on the generalization error of tensor factorizations. In this section we discuss what conclusions can be drawn from the derived bounds. In particular, we are interested in how additional structure or constraints affect the generalization ability of tensor factorizations. For this purpose, we will first present a setting in which it is reasonable to compare tensor factorizations of different order. Furthermore, we will evaluate experimentally how the generalization ability of tensor factorizations behaves with the change of structure and constraints. At last, we will discuss how these results can be interpreted with respect to the derived generalization bounds.

4.4.1. Comparable Tensors

Since it is not reasonable to compare arbitrary tensor factorizations, consider the following setting: Let $\mathcal{T} = (\mathcal{V}, \mathcal{E}, \phi, m)$ be a weighted tuple-set of order m and let $\mathbf{Y}$ be the tensor representation of $\mathcal{T}$. Furthermore, let $\mathbf{Y}^-$ be a tensor representation of $\mathcal{T}$ such that the k-th mode of $\mathbf{Y}^-$ is indexed by the set

$$\mathcal{V}^{(k)-} = \begin{cases} \mathcal{V}^{(k)} & ,k \neq i \neq j \\ \mathcal{V}^{(i)} \times \mathcal{V}^{(j)} & ,k = i. \end{cases}$$

This means that for two index sets $\mathcal{V}^{(i)}, \mathcal{V}^{(j)}$ of $\mathcal{T}$ only a single vector space representation is used in $\mathbf{Y}^-$. Consequently, it holds that $\text{ord}(\mathbf{Y}^-) = \text{ord}(\mathbf{Y}) - 1$. This setting corresponds, for example, to propositionalization in multi-relational learning. We will refer to $\mathbf{Y}^-$ as an *understructured representation* of $\mathcal{T}$. The opposite setting would be an *overstructured representation* where the tensor $\mathbf{Y}^-$ is the correct representation of $\mathcal{T}$, while $\mathbf{Y}$ represents one true index set $\mathcal{V}^{(i)-}$ of $\mathcal{T}$ by two modes. For both, the under- and the overstructured case, we are interested in seeing how the generalization ability of a tensor factorization changes by factorizing $\mathbf{Y}$ compared to $\mathbf{Y}^-$. Without loss of generalization, let $i = m-1$ and let $j = m$ where

$m = \text{ord}(\mathbf{Y})$ and $\ell = \text{ord}(\mathbf{Y}^-) = m - 1$. Furthermore, let $\mathbf{X} = \mathbf{W} \times_1 U^{(1)} \times_2 \cdots \times_m U^{(m)} \in \mathbb{R}^{\boldsymbol{n}}$ and $\mathbf{X}^- = \mathbf{W}^- \times_1 U^{(1)-} \times_2 \cdots \times_\ell U^{(\ell)-} \in \mathbb{R}^{\boldsymbol{n}^-}$ be factorizations of $\mathbf{Y}$ and $\mathbf{Y}^-$. Since we are only interested in the effect that the order of data representation has on the generalization ability, we want to exclude the effect of different ranks. Therefore, analogously to section 4.3, we restrict the tensors $\mathbf{X}$ and $\mathbf{X}^-$ to be of similar n-rank, in order to get comparable models. Since it holds for the Kronecker product that $\text{rank}(V \otimes W) = \text{rank}(V)\,\text{rank}(W)$, we require that

$$r_k^- = \begin{cases} r_k & ,k \neq m \neq \ell \\ r_m r_\ell & ,k = \ell \end{cases}$$

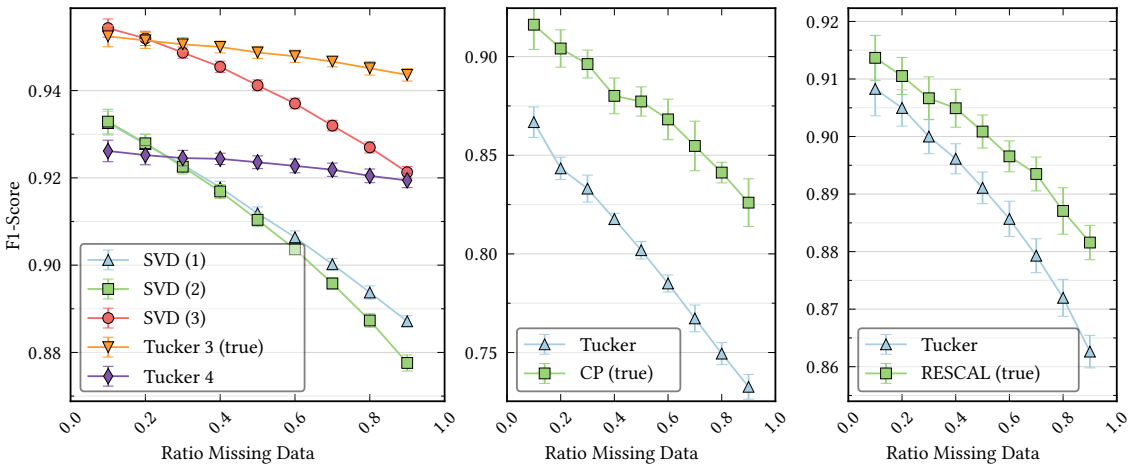
It also follows immediately from the construction of $\mathbf{Y}$ and $\mathbf{Y}^-$ and the properties of the Cartesian product that

$$n_k^- = \begin{cases} n_k & ,k \neq m \neq \ell \\ n_m n_\ell & ,k = \ell \end{cases}$$

In the following, we will refer to tensors $\mathbf{X}$ and $\mathbf{X}^-$ that have these properties as *comparable tensors*. Please note that for comparable tensors, it holds that $\text{var}(\mathbf{X}^-) > \text{var}(\mathbf{X})$, since $n_m n_\ell r_m r_\ell > n_m r_m + n_\ell r_\ell$. Furthermore, it holds that $\text{ord}(\mathbf{X}^-) + 1 = \text{ord}(\mathbf{X})$ and $\text{pol}(\mathbf{X}^-) = \text{pol}(\mathbf{X})$.

4.4.2. Experimental Results

Given comparable tensors, we evaluated experimentally how tensor factorization behaves under the change of structure and constraints. The experiments were carried out on synthetic data with different amounts of missing data. To evaluate the *effects of structure*, we created a third-order tensor $\mathbf{T} = \mathbf{W} \times_1 A \times_2 B \times_3 C$, where $\mathbf{W} \in \mathbb{R}^{5\times10\times2}$, $A \in \mathbb{R}^{50\times5}$, $B \in \mathbb{R}^{100\times10}$, $C \in \mathbb{R}^{20\times2}$ and where all entries of the core tensor and the factor matrices had been drawn from the standard normal distribution $\mathcal{N}(0,1)$. From $\mathbf{T}$ we created the target tensor $\mathbf{Y}$ by setting $y_{ijk} = \text{sgn}(t_{ijk})$. Furthermore, the set of observed entries Ω has been drawn uniformly at random, where we increased the ratio of missing entries from $[0.1, 0.9]$. To evaluate the effects of under- and overstructured representations, we compared three models: a Tucker-3 decomposition, which is the correct model, the singular value decomposition which is an understructured model and a Tucker-4 decomposition, which is an overstructured model. Moreover, the SVD has been computed on all possible unfoldings $Y_{(i)}$, where $i \in \{1,2,3\}$. For the Tucker-4 decomposition, we split the second mode of $\mathbf{Y}$ into two size-10 modes, such that $\mathbf{Y}_4 \in \mathbb{R}^{50\times10\times10\times20}$. For each model and each ratio of missing entries we computed 100 factorizations and recorded the F1-score for the classification of the missing entries compared to the ground truth. Figure 4.1a shows the results of these experiments. As expected, the



(a) Third-Order Model *(b)* CP Model *(c)* RESCAL Model

Figure 4.1.: Mean and standard error of the F1-Score over 100 iterations per percentage of missing data. SVD (i) denotes the singular value decomposition of $Y_{(i)}$, i.e. the unfolding of the i-th mode of $\mathbf{Y}$.

true model provides the best overall performance. One understructured model, i.e. SVD (3), shows comparable results to the true model for low amounts of missing entries but scales significantly worse as the missing data increases. The overstructured model displays the opposite behaviour; it shows reduced overall generalization ability compared to the true model but is more stable with the amount of missing data.

In similar experiments we also evaluated the *effects of constraints*. For this purpose, we created synthetic CP and RESCAL models under similar conditions as in the previous experiment. However, in this experiment we evaluated how the correct model compared to an unconstrained TUCKER model. Figure 4.1b and figure 4.1c show the results of these experiments. Again, the true models show the best overall performance in both experiments. Furthermore, in both settings, the constrained models scale better with the amount of missing data than the unconstrained TUCKER model.

4.4.3. Discussion

The previously derived generalization bounds can provide insight in how to interpret these experimental results. First, note that both terms in equation (4.4), i.e. $\mathcal{D}_\Omega(\mathbf{X}, \mathbf{Y})$ and ε, are influenced by the number of sign patterns that a factorization can express. For $\mathcal{D}_\Omega(\mathbf{X}, \mathbf{Y})$ this is the case because the discrepancy will increase when a model $\mathbf{X}$ is not expressive enough to model the sign patterns of a target tensor $\mathbf{Y}$. Furthermore, it has been shown in section 4.3

that the term ε grows with the number of sign patterns. Since it has also been shown that the upper bound on the number of sign configurations in lemma 3 is tight at least up to a multiplicative factor in the exponent, we consider how this bound changes with the order of the data representation – to see what possible effects the change of structure can have in terms of the generalization ability.

Corollary 1. *For comparable tensors* $\mathbf{X} \in \mathbb{R}^{\boldsymbol{n}}$, $\mathbf{X}^- \in \mathbb{R}^{\boldsymbol{n}^-}$ *with* $\operatorname{ord}(\mathbf{X}) = \operatorname{ord}(\mathbf{X})^- + 1$, $\text{n-rank}(\mathbf{X}) = \boldsymbol{r}$ *and* $\text{n-rank}(\mathbf{X})^- = \boldsymbol{r}^-$, *the ratio of upper bounds on then number of possible sign patterns is at most*

$$1 < \frac{O(|\mathcal{S}^-_{\boldsymbol{n},\boldsymbol{r}}|)}{O(|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}|)} < \left(\frac{4e\ (\operatorname{ord}(\mathbf{X}^-) + 1)\ \operatorname{pol}(\mathbf{X})}{\operatorname{var}(\mathbf{X}^-)}\right)^{\upsilon}$$

where $\upsilon = n_m n_\ell r_m r_\ell - (n_\ell r_\ell + n_m r_m) > 0$

Proof. It follows straight from the definition of comparable tensors that $\operatorname{var}(\mathbf{X}^-)$ can be rewritten as $\operatorname{var}(\mathbf{X}^-) = \operatorname{var}(\mathbf{X}) + \upsilon$. Furthermore, let

$$\alpha = 4e\ (\operatorname{ord}(\mathbf{X}^-) + 1)\ \operatorname{pol}(\mathbf{X})$$
$$\beta = 4e\ (\operatorname{ord}(\mathbf{X}) + 1)\ \operatorname{pol}(\mathbf{X}) = \alpha + 4e\ \operatorname{pol}(\mathbf{X})$$

Then, it holds that

$$\begin{aligned}\frac{O(|\mathcal{S}^-_{\boldsymbol{n},\boldsymbol{r}}|)}{O(|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}|)} &= \frac{\alpha^{\operatorname{var}(\mathbf{X})+\upsilon}}{\operatorname{var}(\mathbf{X}^-)^{\operatorname{var}(\mathbf{X})+\upsilon}} \frac{\operatorname{var}(\mathbf{X})^{\operatorname{var}(\mathbf{X})}}{\beta^{\operatorname{var}(\mathbf{X})}} \\ &= \left(\frac{\alpha}{\operatorname{var}(\mathbf{X}^-)}\right)^{\upsilon} \frac{\alpha^{\operatorname{var}(\mathbf{X})}}{\beta^{\operatorname{var}(\mathbf{X})}} \frac{\operatorname{var}(\mathbf{X})^{\operatorname{var}(\mathbf{X})}}{(\operatorname{var}(\mathbf{X}) + \upsilon)^{\operatorname{var}(\mathbf{X})}} \leq \left(\frac{\alpha}{\operatorname{var}(\mathbf{X}^-)}\right)^{\upsilon}\end{aligned}$$

□

The main result of corollary 1 for this discussion is that the bound increases as we decrease the order of the tensor. This suggests that as we increase the order of the data representation, we will reduce the term ε in equation (4.4). As the amount of missing data increases, it is therefore likely to see increasingly severe overfitting for $\mathbf{X}^-$ compared to $\mathbf{X}$. However, when $\mathbf{X}^-$ is the correct and $\mathbf{X}$ is an understructured representation, $O(|\mathcal{S}^-_{\boldsymbol{n},\boldsymbol{r}}|) > O(|\mathcal{S}_{\boldsymbol{n}}, \boldsymbol{r}|)$ also suggests that the model $\mathbf{X}$ might not be expressive enough to model the sign patterns of $\mathbf{Y}^-$. This corresponds nicely to the experimental results shown in figure 4.1a. The understructured models are expressive enough to model the sign patterns of $\mathbf{Y}$, as seen in the case of SVD (3). However, they also scale significantly worse than the correct model with the amount of

missing data. The overstructured TUCKER-4 model scales even better with missing data than the true model, but at the same time gives significantly worse overall results, what suggests that it might not be expressive enough. A possible interpretation is therefore, that the ratio between expressiveness and overfitting is superior for a correct model specification. Since the correct model $\mathbf{X}$ has a much smaller number of variables, it should also be noted that the memory complexity of $\mathbf{X}$ is significantly reduced compared to $\mathbf{X}^{-}$.

Similar arguments apply for the effect of constraints. Here, the key insight is that both CP-type and RESCAL-type constraints decrease the number of variables in a model. Models like CP or the Block-Term Decompostion require that $\mathbf{W}$ is superdiagonal or block-superdiagonal and therefore set most entries in the core tensor to $w_{\boldsymbol{i}} = 0$. Models like RESCAL on the other hand, decrease the number of variables through the constraint that some factor matrices $U^{(i)}$, $U^{(j)}$ have to be identical. Since $O(|\mathcal{S}_{\boldsymbol{n},\boldsymbol{r}}|)$ depends exponentially on $\mathrm{var}(\mathbf{X})$, conclusions similar to the effects of structure can be drawn with regard to the effects of constraints. It suggests that a model with a larger number of variables, i.e. fewer constraints, has more capacity to model sign patterns, but at the same time is more likely to overfit as the amount of missing data increases. Again, this corresponds nicely to the experimental results in figure 4.1b and figure 4.1c.

4.5. Related Work

We are not aware of any previous generalization error bounds for tensor factorizations or of any theoretical results that relate the order of a tensor and the order of structured data to the generalization ability of factorizations. Our derivation of error bounds for the tensor case builds strongly on the work of Srebro et al. (Srebro et al., 2005; Srebro, 2004), which provided error bounds for matrix factorizations with zero-one loss and general loss functions. Wolf et al. (2007) derived similar bounds in the context of rank-k SVMs. For general matrices, Candès and Recht (2009) and Candès and Plan (2010) show that under suitable conditions a low-rank matrix can be recovered from a minimal set of entries via convex optimization and also provide theoretical bounds. Gandy et al. (2011) and Tomioka et al. (2012) extend these methods to tensor completion, although without providing error bounds. It has also been shown experimentally that by adding structure to the vector space representations via the tensor product, the amount of data needed for exact recovery can be greatly reduced (Tomioka et al., 2012; Tomioka et al., 2010).

4.6. Conclusion

To obtain a deeper understanding of the generalization ability of tensor factorizations, we derived the first known generalization error bounds based on the number of sign patterns that a tensor factorization can express. Using a general framework to describe structured data based on weighted tuple-sets, we analyzed how tensor factorizations behave when their order does not match the true order of the data. We showed experimentally that structuring vector space representations via the tensor product, up to the true order of the data, adds important information such that tensor models often scale better with sparsity or missing data than their understructured counterparts. We also discussed analytically how this behaviour can be explained in the light of the newly derived generalization bounds. In this work, we only considered binary values for the target tensor $\mathbf{Y}$, which corresponds to a classification setting. For future work, it would prove very valuable to also derive error bounds for the more general case of real-valued weight functions. Since the current error bounds are based on the assumption that the observed entries are independently and identically distributed, what – especially for structured data – might not hold, it might also be useful to consider techniques as developed by Mohri and Rostamizadeh (2009), to overcome this limitation.

Chapter 5

Conclusion

In this thesis, we examined several aspects of tensor factorization for relational learning, which we want to summarize in this chapter and furthermore discuss interesting directions for future research.

5.1. Summary

In chapter 2 we proposed RESCAL, a novel tensor factorization for relational learning, and discussed its properties and applications. Due to its latent variable structure, RESCAL is able to capture global dependencies in relational data and also shows a strong collective learning effect such that information that is more distant in the relational graph can be efficiently exploited. Furthermore, the model does not require deep domain knowledge such that it can be applied easily to most relational domains. These factors contributed all to a very good performance of the factorization for different relational learning tasks. On various benchmark data sets, RESCAL was usually able to outperform related tensor factorization approaches and moreover, to provide better or on-par results when compared to state-of-the-art relational learning methods. Another feature of RESCAL is that it renders relational learning very manageable, what is less quantifiable than its prediction performance but also very important for its applicability. As previously noted, no prior knowledge about a domain of application is needed for the functioning of the model, what facilitates the learning process significantly. Moreover, RESCAL enables the application of non-relational algorithms to relational data via its latent representation of entities for a wide range of tasks. This way, existing and proven machine learning methods that solve these tasks for vector space representations can be

reused. At last, we also want to make a case for the simplicity of implementing RESCAL-ALS, which requires less than 200 lines of code in PYTHON, NUMPY and SCIPY (Jones et al., 2001; T. E. Oliphant, 2007). Moreover, any implementation of RESCAL-ALS can directly exploit highly optimized linear algebra libraries via the BLAS and LAPACK interface (Anderson et al., 1999) since the algorithm uses only standard matrix operations in its computations.

In chapter 3 we showed that an efficient computation of RESCAL scales linearly with the number of entities, the number of relations and the number of known facts, such that complete knowledge bases of the size of the YAGO2 core ontology can be factorized on a single desktop computer. Moreover, we showed how attributes of entities can be handled efficiently via coupled tensor-matrix factorization and also reduced the runtime as well as the memory complexity of RESCAL-ALS significantly. Both of these improvements also increased the applicability of RESCAL to Linked Data, where learning tasks such as instance matching or taxonomy learning require algorithms to scale up to the size of complete knowledge bases and where much of the information about entities exists in form of attribute values.

One of the main results of this thesis is the fact that the results of chapter 2 and chapter 3 were achieved with a single relational learning approach that combines both, state-of-the-art learning results *and* linear scalability with the size of relational data. This property of RESCAL, in combination with its versatility to approach different learning tasks, can be an important step towards relational learning from complete knowledge bases in the web of data.

In chapter 4 we discussed tensor factorization as a principled way for learning from structured data. For this purpose, we derived the first known generalization error bounds for tensor factorizations in a classification setting on structured data, which also subsumes link prediction on relational data as a special case. Moreover, these bounds are not only interesting with regard to the analysis of the generalization ability of tensor factorizations in this setting, but they also give valuable insight how the improved scalability with missing data of higher-order factorizations can be interpreted. While the bounds presented in this thesis are currently confined to a classification setting where the observed entries uniformly distributed, it can be a first step to a rigorous analysis of the advantages of higher-order data representations.

5.2. Outlook

The RESCAL model has shown very good results, both in its ability to learn from relational data and in its scalability with the size of relational data. In the following, we want to briefly outline

interesting directions for future research that can further improve its relational learning capabilities and its applicability to large knowledge bases.

Learning Directly from Observable Variables

A limiting factor for the scalability of Rescal is its computational complexity and its memory complexity with regard to the number of latent components r, which is cubic for Rescal-ALS. Hence, relational domains that require a very large number of latent components are currently out of reach for computational reasons. A very interesting and valuable line of research is therefore to reduce the complexity of the factorization with regard to this parameter. An interesting observation in this context is that relations that require a large number of latent components can often be explained efficiently via simple patterns on the observable variables of the data. For instance, consider a simplified instance of the relation `marriedTo`, where each person is married to exactly one other person. The adjacency matrix for this relation – after appropriate reordering of columns and rows – would be a square matrix A with entries a_{ij} such that

$$a_{ij} = \begin{cases} 1, & \text{if } j = i + 1 \vee j = i - 1 \\ 0, & \text{else.} \end{cases}$$

Hence, all super- and subdiagonal entries of A are set to one and all other entries are set to zero. It is easy to show that this adjacency matrix can be created via $A = P \otimes I$, where

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

and where I is the identity matrix whose size corresponds to the number of marriages in the data. Since both matrices P and I are of full rank, and since $\operatorname{rank}(A \otimes B) = \operatorname{rank}(A)\operatorname{rank}(B)$, the matrix $A = P \otimes I$ would also be of full rank. Therefore, while Rescal would certainly be able to *learn* the model for such a high-rank relation – what we also verified experimentally on synthetic data – it would also need a large number of latent components to *reconstruct* the data exactly, what limits its scalability on such relations. Interestingly, although such an instance of `marriedTo` would have a high complexity when explained via latent variables, it could be explained very easily via the simple rule

$$\texttt{marriedTo}(a, b) \Leftrightarrow \texttt{marriedTo}(b, a),$$

which operates on the observable variables of the data. Being able to exploit these kind of patterns on observable variables efficiently could therefore be a very valuable extension to

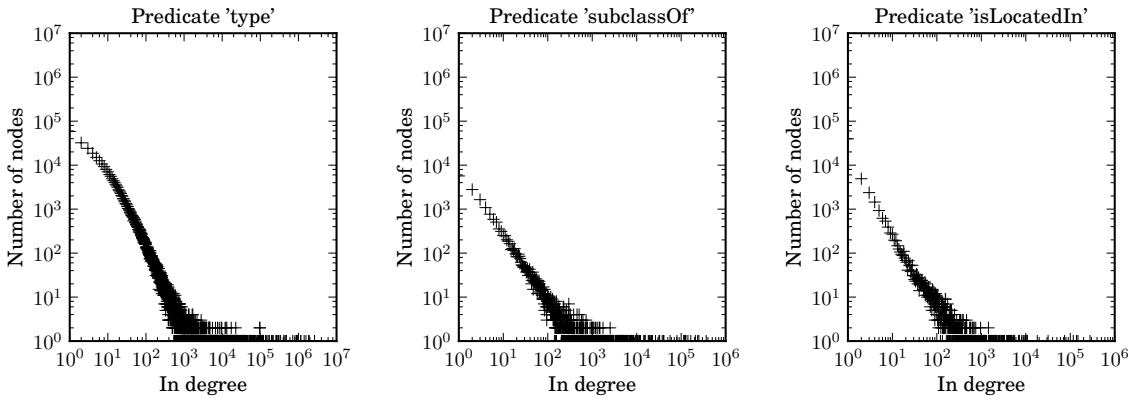


Figure 5.1.: Log-log plots of the in-degree distribution for various relations in the YAGO2 ontology. In such log-log plots, the degree distribution in scale-free networks tends to form a straight line for large k with slope $-\gamma$ (Jeong et al., 2000).

the RESCAL model, for which we examine in ongoing research the use of additive tensor factorizations.

SCALE-FREE REGULARIZATION

Many networks in social network analysis, computer science or biology are assumed to be scale-free and heavy-tailed (Clauset et al., 2009). In a scale-free network, the probability of a randomly selected node having exactly k links to other nodes in the network follows a power-law distribution

$$P(k) \sim k^{-\gamma},$$

where the scaling parameter γ lies typically within $2 \leq \gamma \leq 3$. Interestingly, similar degree distributions can be observed on large-scale multi-relational data. For instance, figure 5.1 shows a few examples of relations in the YAGO2 ontology, where the in-degrees of entities, i.e. their occurrences of objects in the particular relation, are distributed very similar to a power-law distribution. In its present form, RESCAL does not explicitly account for the scale-free nature of many networks, what can potentially diminish its ability to model such networks. An interesting extension of RESCAL could therefore be to include power-law distributed bias terms in the factorization, such that

$$P(x_{ijk} = 1) \propto \boldsymbol{a}_i^T R_k \boldsymbol{a}_j + b_i^{in} + b_j^{out},$$

where b_i^{in} and b_j^{out} represent the general probability of nodes i and j to be the subject or the object in a relationship respectively. By grouping these bias terms into vectors $\boldsymbol{b}^{in}$ and $\boldsymbol{b}^{out}$

any by enforcing a power-law distribution of their entries, the desired effect of scale-free degree distributions could be modeled. Possible ways to enforce such power-law distributions have been considered in form of reweighted ℓ_1 regularization (Q. Liu and Ihler, 2011) and Pitman-Yor processes (Teh, 2006) which could, similarly as t-logistic regression (Ding and Vishwanathan, 2010), be interesting starting points to extend the RESCAL model to scale-free networks.

Scalable Logistic Model

It has been discussed in section 2.2 that by using the least-squares loss to compute the RESCAL factorization, we implicitly assume that the random variation in an observed tensor $\mathbf{X}$ is normally distributed, i.e. that

$$x_{ijk} \sim \mathcal{N}(\boldsymbol{a}_i^T R_k \boldsymbol{a}_j, \sigma^2).$$

However, for binary-valued variables x_{ijk} as in relational data, the Bernoulli distribution would usually be the preferred distribution to model the random variation of these variables. Conveniently, the Bernoulli distribution is also a member of the family of exponential distributions, such that under an appropriate loss function, RESCAL can be interpreted as computing the MAP estimates of A and $\mathbf{R}$, where

$$x_{ijk} \sim \mathit{Bernoulli}(\boldsymbol{a}_i^T R_k \boldsymbol{a}_j).$$

In the theory of the exponential family, the logistic function describes the inverse parameter mapping of the Bernoulli distribution. In Nickel and Tresp (2013b), we examined therefore a logistic variant of the RESCAL factorization to describe the random variation in the data via a Bernoulli distribution, by changing the loss function to the logistic loss and by optimizing

$$\arg\max_{A,\mathbf{R}} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} x_{ijk} \log h(\boldsymbol{a}_i^T R_k \boldsymbol{a}_j) + (1-x_{ijk}) \log(1-h(\boldsymbol{a}_i^T R_k \boldsymbol{a}_j)) + \lambda_A \|A\|^2 + \lambda_{\mathbf{R}} \|\mathbf{R}\|^2 \quad (5.1)$$

where

$$h(\boldsymbol{a}_i^T R_k \boldsymbol{a}_j) := \frac{1}{1+\exp(-\boldsymbol{a}_i^T R_k \boldsymbol{a}_j)}$$

denotes the logistic function. Independently, a similar logistic extension of the RESCAL factorization has also been proposed by London et al. (2013). Unfortunately, there exists no closed form solution to compute equation (5.1), such that we employed a gradient-based approach via quasi-Newton optimization this objective. Using the logistic extension of RESCAL, it was possible to improve the link prediction results compared to RESCAL-ALS as shown

Table 5.1.: Area under the precision-recall curve on the Kinships, Nations, and U.S. Presidents data sets for logistic and least-squares RESCAL models.

	AUC-PR		
	Kinships	Nations	U.S. Presidents
RESCAL-ALS	0.966	0.843	0.805
RESCAL-Logit	0.981	0.851	0.800

in table 5.1. Especially the improvements on Kinships are noteworthy, considering the already very good results of RESCAL-ALS on this data set. Unfortunately, the partial gradients of equation (5.1) include terms of the form

$$\left(h\left(AR_kA^T\right) - X_k\right)A,$$

which can not be reduced to a significantly simpler form, due to the logistic function $h(\cdot)$. Currently, the logistic variant of RESCAL requires therefore to compute the dense matrix AR_kA^T, what limits its scalability as it results in a quadratic runtime and memory complexity in the number of entities. Hence, a valuable direction of future work would be to improve the scalability of logistic RESCAL, as it is currently too limited for practical use on larger data sets but simultaneously shows encouraging results on benchmark data.

Appendix A

Block-Partitioned Matrix Multiplication

In RESCAL, products of block-partitioned matrices occur in the model itself as well as in the alternating least-squares algorithm to compute the factorization. For general block-partitioned matrices $A \in \mathbb{R}^{p \times q}$, $B \in \mathbb{R}^{m \times n}$

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1q} \\ \vdots & \ddots & \vdots \\ A_{p1} & \cdots & A_{pq} \end{bmatrix}, \qquad B = \begin{bmatrix} B_{11} & \cdots & B_{1n} \\ \vdots & \ddots & \vdots \\ B_{m1} & \cdots & B_{mn} \end{bmatrix}$$

with compatible partitions it holds that the matrix product $C = AB$ is a block-partitioned matrix with (Golub and Van Loan, 1996, Theorem 1.3.2)

$$C_{ij} = \sum_{k=1}^{q} A_{ik} B_{kj} \tag{A.1}$$

In the following, we will use this property of block-partitioned matrices to explain the equivalence of the tensor and matrix view of RESCAL as well as to simplify the computations in RESCAL-ALS.

A.1. Tensor and Matrix View of RESCAL

The equivalence of equation (2.2) and equation (2.3) follows immediately from equation (A.1) and the unfolded representation of a TUCKER-2 model: According to equation (1.16), the unfolding of equation (2.2) in the first mode is equivalent to $X_{(1)} \approx AR_{(1)}(I \otimes A)^T$. Please note, that the unfolding of an arbitrary third-order tensor $\mathbf{T} \in \mathbb{R}^{n \times m \times \ell}$ in its first mode is a block-partitioned matrix where each block correspond to a frontal slices of $\mathbf{T}$, i.e.

$$T_{(1)} = \begin{bmatrix} T_1 & T_2 & \ldots & T_\ell \end{bmatrix}$$

Furthermore, $(I \otimes A)^T$ is also a block-partitioned matrix, where

$$(I \otimes A)^T = \begin{bmatrix} A^T & 0 & \cdots & 0 \\ 0 & A^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A^T \end{bmatrix}$$

Consequently, it follows from the mode-1 unfolding of equation (2.2) and equation (A.1) that

$$\begin{aligned} X_{(1)} = \begin{bmatrix} X_1 & X_2 & \dots & X_m \end{bmatrix} &\approx AR_{(1)}(I \otimes A)^T \\ &= A \begin{bmatrix} R_1 & R_2 & \dots & R_m \end{bmatrix} \begin{bmatrix} A^T & 0 & \cdots & 0 \\ 0 & A^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A^T \end{bmatrix} \\ &= \begin{bmatrix} AR_1A^T & AR_2A^T & \dots & AR_mA^T \end{bmatrix}, \end{aligned}$$

what is equivalent to equation (2.3).

A.2. Simplification of Rescal-ALS

To simplify the matrix products $\underline{X}M$ and MM^T in the update step of A, we use similar properties as in appendix A.1. First, recall that M equals $\underline{R}(I \otimes \underline{A})^T$ and that the matrices $\underline{X}$, $\underline{R}$ are block partitioned, where each block corresponds to a frontal slices of $\mathbf{X}$ and $\mathbf{R}$ or its transpose. As in appendix A.1, the matrix $(I \otimes \underline{A})^T$ is block-partitioned with

$$(I \otimes \underline{A})^T = \begin{bmatrix} \underline{A}^T & 0 & \cdots & 0 \\ 0 & \underline{A}^T & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \underline{A}^T \end{bmatrix}$$

Applying Equation A.1 to $\underline{X}M^T$ and MM^T respectively, it follows that

$$\underline{X}M = \sum_{k=1}^{K} X_k \underline{A} R_k^T + X_k^T \underline{A} R_k, \qquad MM^T = \sum_{k=1}^{K} R_k \underline{A}^T \underline{A} R_k^T + R_k^T \underline{A}^T \underline{A} R_k$$

This concludes the full derivation of the update step of A in section 2.5.

Appendix B

MLN on U.S. Presidents

The MLN experiments in section 2.6.1 have been carried out using the current "Aug 23, 2010" release of the Alchemy toolbox.[1]. To learn the weights of manually specified formulas, we used Alchemy's `learnwts` command with the parameters listed in table B.1.

Table B.1.: Parameters for MLN weight learning with `learnwts`.

Parameter	Description	Value
`-ne`	Non-evidence predicates	`party`
`-d`	Discriminative Learning	True

For structure learning, we tried different combinations of parameter settings; what produced a quite diverse set of formulas, but in all settings we did not succeed to learn a good set of formulas. In the following we list an example for one of the better parameter settings in table B.3 and the formulas that have been learned by Alchemy's `learnstruct` in table B.2. These are also the settings used for structure learning in the experiments of section 2.6.1. All parameters not listed in table B.3 were set to default values by Alchemy.

Table B.2.: Formulas learned with MLN structure learning.

Formula	Weight
`party(a1,a2)`	0.332
`¬party(a1,a2) ∨ ¬party(a1,a3)`	1.645
`¬party(a1,a2) ∨ ¬party(a3,a2) ∨ ¬president(a3,a1)`	-4.00777
`party(a1,a2) ∨ ¬party(a1,a3) ∨ ¬party(a4,a1)`	0.128603

[1] Available from `http://alchemy.cs.washington.edu`

Table B.3.: Parameters for MLN structure learning with `learnstruct`.

Parameter	Description	Value
`-minWt`	Minimum Weight to accept formulas	0
`-startFromEmptyMLN`	Start structure learning from an empty MLN	True
`-ne`	Non-evidence predicates	`party`
`-penalty`	Penalty for each difference between previous and current candidate clauses	0.001

Appendix C

Complexity Analysis of CP and Tucker Algorithms

In the following, we will provide a brief complexity analysis of state-of-the-art algorithms to compute the CP and Tucker factorizations, in which we will show important differences to the complexity of Rescal-ALS.

C.1. Computational Complexity of CP-ALS

First, we consider the computational complexity of CP-ALS, which is a state-of-the art approach to compute the CP factorization based on alternating least squares updates. For a tensor $\mathbf{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ it computes updates for factors $A \in \mathbb{R}^{n_1 \times r}$, $B \in \mathbb{R}^{n_2 \times r}$, $C \in \mathbb{R}^{n_3 \times r}$ as follows (Kolda and Bader, 2009, Section 3.4):

$$A \leftarrow X_{(1)}(C \odot B)(C^T C * B^T B)^{-1} \tag{C.1}$$

$$B \leftarrow X_{(2)}(C \odot A)(C^T C * A^T A)^{-1} \tag{C.2}$$

$$C \leftarrow X_{(3)}(B \odot A)(B^T B * A^T A)^{-1}. \tag{C.3}$$

The symbol "$\odot$" denotes the column-wise Khatri-Rao product, which is defined as:

Definition 12 (Column-wise Khatri-Rao Product)**.** *Let $A \in \mathbb{R}^{p \times r}$, $B \in \mathbb{R}^{q \times r}$ be two matrices with an identical number of rows. The column-wise Khatri-Rao product of A and B, in the following denoted by $A \odot B$, is a matrix of size $pq \times r$, such that*

$$A \odot B := \begin{bmatrix} \boldsymbol{a}_{:,1} \otimes \boldsymbol{b}_{:,1} & \boldsymbol{a}_{:,2} \otimes \boldsymbol{b}_{:,2} & \dots & \boldsymbol{a}_{:,r} \otimes \boldsymbol{b}_{:,r} \end{bmatrix}.$$

It follows from definition 12 that the matrix $B \odot A$ in equation (C.3) is of size $n^2 \times r$.

Algorithm 2 Higher-Order Orthogonal Iterations (De Lathauwer et al., 2000b)

1: **function** TUCKER_HOOI($\mathbf{X}, r_1, \ldots, r_N$)
2: $A^{(1)}, \ldots, A^{(N)} \leftarrow$ initialize with HOSVD
3: **while** not converged **do**
4: **for** $i = 1 \ldots N$ **do**
5: $\mathbf{Y} \leftarrow \mathbf{X} \times_1 A^{(1)T} \times_2 \cdots \times_{i-1} A^{(i-1)T} \times_{i+1} A^{(i+1)T} \times_{i+1} \cdots \times_N A^{(N)T}$
6: $A^{(i)} \leftarrow r_n$ leading left singular vectors of $Y_{(i)}$
7: **end for**
8: **end while**
9: $\mathbf{G} \leftarrow \mathbf{X} \times_1 A^{(1)T} \times_2 \cdots \times_N A^{(N)T}$
10: **return G**, $A^{(1)}, \ldots, A^{(N)}$
11: **end function**

According to table 3.1, the computational complexity for updates of C would therefore be $O(n^2r + nr^2 + r^3)$ for an adjacency tensor of size $n \times n \times m$. It follows that CP-ALS would scale quadric with the number of objects in the data instead of linearly as in the case of Rescal-ALS.

C.2. Computational Complexity of the Tucker decomposition

Higher-Order Orthogonal Iterations (HOOI) are the state-of-the-art method to compute the Tucker decomposition. The detailed algorithm for HOOI is listed in algorithm 2. A particular problem regarding the computation of the Tucker decomposition is that the intermediate computations within line 5 of algorithm 2 can become top large to fit into memory as these intermediate results are relatively large dense tensors. This is known as the "intermediate blowup problem" and can be approached with a memory efficient version to calculate the tensor-times-matrix product under the cost of computational efficiency (Kolda and Sun, 2008). However, the requirements for orthogonal factor matrices are even more demanding. In the Tensor Toolbox (Bader and Kolda, 2012; Bader and Kolda, 2007), which is a state-of-the-art toolkit to compute tensor decompositions, this constraint leads to the computation of the first r eigenvectors of the matrix $Y_{(i)}Y_{(i)}^T$, where $Y_{(i)}$ denotes the unfolding of the tensor $\mathbf{Y}$ in line 5 of algorithm 2. This unfolding $Y_{(i)}$ is a *dense* matrix of size $n_i \times \prod_{j \neq i} r_i$, such that the product $Y_{(i)}Y_{(i)}^T$ would already have a computational complexity of $O(n^2r^2)$ for $r = r_1 = r_2 = r_3$. For a Tucker-2 model such as Rescal, where $r_3 = m$ and $r = r_1 = r_2$, the complexity of this operation would amount to $O(n^2mr)$. An orthogonal Tucker decomposition would therefore become at least quadratic in the number of entities. Moreover, since $Y_{(i)}$ is dense, the memory complexity would also be quadratic in the number of entities, as $Y_{(i)}Y_{(i)}^T$ is a $n \times n$ matrix.

Appendix D

Number of Variables for Comparable Tensors

Let $\mathbf{X}$ and $\mathbf{X}^-$ be comparable tensors with $\operatorname{ord}(\mathbf{X}) = o = \operatorname{ord}(\mathbf{X}^-) + 1$. Also, recall that for comparable tensors it holds that $\operatorname{var}(\mathbf{X}^-) > \operatorname{var}(\mathbf{X})$. Then, we can rewrite $\operatorname{var}(\mathbf{X})$ in the following way: The number of variables for $\mathbf{X}^-$ is given by

$$\operatorname{var}(\mathbf{X}^-) = \prod_{i=1}^{o-1} r_i^- + \sum_{i=1}^{o-1} n_i^- r_j^- .$$

From the construction of comparable tensors it follows that

$$\begin{aligned}\operatorname{var}(\mathbf{X}^-) &= r_{o-1} r_o \prod_{i=1}^{o-2} r_i + \sum_{i=1}^{o-2} n_i r_i + n_{o-1} n_o r_{o-1} r_o \\ &= \prod_{i=1}^{o} r_i + \sum_{i=1}^{o} n_i r_i + n_{o-1} n_o r_{o-1} r_o - n_{o-1} r_{o-1} - n_o r_o\end{aligned}$$

By substituting $\operatorname{var}(\mathbf{X}) = \prod_i r_i + \sum_i n_i r_i$ it follows immediately that

$$\operatorname{var}(\mathbf{X}^-) = \operatorname{var}(\mathbf{X}) + n_{o-1} n_o r_{o-1} r_o - n_{o-1} r_{o-1} - n_o r_o$$

Furthermore, since $n_o > r_o > 2$ and $n_{o-1} > r_{o-1} > 2$, it follows that

$$n_{o-1} n_o r_{o-1} r_o - (n_{o-1} r_{o-1} + n_o r_o) > 0$$

List of Figures

1.1. Probabilistic Relational Models 11
1.2. Markov Logic Networks 13
1.3. Tensor Product of Vectors 17
1.4. Fibers and slices of a third-order tensor **X**. 20
1.5. Unfolding Operation 20
1.6. The CANDECOMP / PARAFAC Decomposition 22
1.7. The TUCKER Decomposition 24

2.1. The RESCAL factorization 32
2.2. Graphical Model of RESCAL 35
2.3. Entity-specific Relationships in an Adjacency Tensor 38
2.4. Collective Learning Example 39
2.5. The DEDICOM factorization 45
2.6. Visualization of U.S. Presidents Data 52
2.7. Experimental Results for Link Prediction on U.S. Presidents Data 54
2.8. Relations between types of entities in the Cora data set. 55
2.9. Experimental Results for Link Prediction on Benchmark Data 57
2.10. Experimental Results for Link-based Clustering on Nations Data 58
2.11. Experimental Results on Synthetic Social Network 60

3.1. Matrix Structure of Attribute Data in an Adjacency Tensor 77
3.2. Experimental Results for Scalability Synthetic Data 80
3.3. Experimental Results for Collective Learning on YAGO2 writers data 86

4.1. Experimental Results for Missing Data on Comparable Tensors 105

5.1. Distribution of In-degrees in Relations of the YAGO2 ontology 112

List of Tables

2.1. Data statistics of recent recommendation challenges. 30
2.2. Tensor decomposition algorithms used in experiments 50
2.3. Experimental Results for Entity Resolution on Cora 56
2.4. Experimental Results on Synthetic Social Network 59

3.1. Computation Complexity of Standard Matrix Operations 69
3.2. Computational Complexity of Update Steps in RESCAL-ALS 71
3.3. Computational Complexity of Scalable Updates for the **R** in RESCAL-ALS . . 76
3.4. Computational Complexity of Coupled Tensor-Matrix Factorization 79
3.5. Parameter values for scalability experiments 81
3.6. Runtime comparison on benchmark data 82
3.7. Statistics for YAGO2 83
3.8. Statistics for selected classes in YAGO2 core 84
3.9. Experimental Results for Link-Prediction on YAGO2 core 85
3.10. Experimental Results for Taxonomy Learning 88

5.1. Experimental Results for Logistic RESCAL 114

B.1. Parameters for MLN weight learning with `learnwts`. 117
B.2. Formulas learned with MLN structure learning. 117
B.3. Parameters for MLN structure learning with `learnstruct`. 118

List of Algorithms

1. Scalable computation of updates for R 75

2. Higher-Order Orthogonal Iterations (De Lathauwer et al., 2000b) 120

Bibliography

[1] E. Acar and B. Yener. "Unsupervised Multiway Data Analysis: A Literature Survey". In: *IEEE Transactions on Knowledge and Data Engineering* 21.1 (2009), pp. 6–20. ISSN: 1041-4347. DOI: 10.1109/TKDE.2008.112.

[2] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. "Scalable tensor factorizations for incomplete data". In: *Chemometrics and Intelligent Laboratory Systems* (2010). ISSN: 01697439. DOI: 10.1016/j.chemolab.2010.08.004. URL: http://linkinghub.elsevier.com/retrieve/pii/S0169743910001437.

[3] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. "Mixed Membership Stochastic Blockmodels". In: *Journal of Machine Learning Research* 9 (Sept. 2008), pp. 1981–2014. URL: http://jmlr.csail.mit.edu/papers/v9/airoldi08a.html.

[4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8 (paperback).

[5] M. Ankerst, M. Breunig, H. Kriegel, and J. Sander. "OPTICS: ordering points to identify the clustering structure". In: *ACM SIGMOD Record*. Vol. 28. 1999, pp. 49–60.

[6] M. Anthony and M. Harvey. *Linear Algebra: Concepts and Methods*. Cambridge University Press, May 2012. ISBN: 9780521279482.

[7] G. Antoniou and F. van Harmelen. *A Semantic Web Primer*. Cambridge, MA, USA: MIT Press, Apr. 2004. ISBN: 0262012103. URL: http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=10140.

[8] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. "DBpedia: A nucleus for a web of open data". In: *The Semantic Web* (2008), pp. 722–735.

[9] S. Auer, I. Ermilov, J. Lehmann, and M. Martin. *LODStats: A statement-stream based approach for gathering comprehensive statistics about RDF datasets*. 2013. URL: http://aksw.org/Projects/LODStats.html (accessed 05/29/2013).

[10] S. Auer and J. Lehmann. "Creating knowledge out of interlinked data". In: *Semantic Web* 1.1 (Jan. 2010), pp. 97–104. DOI: 10.3233/SW-2010-0019. URL: http://dx.doi.org/10.3233/SW-2010-0019.

[11] B. W. Bader, R. A. Harshman, and T. G. Kolda. "Temporal Analysis of Semantic Graphs Using ASALSAN". In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. Omaha, NE, USA, Oct. 2007, pp. 33–42. DOI: 10.1109/ICDM.2007.54. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4470227.

[12] B. W. Bader and T. G. Kolda. "Efficient MATLAB computations with sparse and factored tensors". In: *SIAM Journal on Scientific Computing* 30.1 (Dec. 2007), pp. 205–231. DOI: 10.1137/060676489.

[13] B. W. Bader and T. G. Kolda. *MATLAB Tensor Toolbox Version 2.5*. Jan. 2012. URL: http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/.

[14] T. Berners-Lee and M. Fischetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*. en. Paw Prints, June 2008. ISBN: 9781439500361.

[15] I. Bhattacharya and L. Getoor. "Collective entity resolution in relational data". In: *ACM Trans. Knowl. Discov. Data* 1.1 (Mar. 2007). ISSN: 1556-4681. DOI: 10.1145/1217299.1217304. URL: http://doi.acm.org/10.1145/1217299.1217304.

[16] V. Bicer, T. Tran, and A. Gossen. "Relational Kernel Machines for Learning from Graph-Structured RDF Data". In: *The Semantic Web: Research and Applications* (2011), pp. 47–62.

[17] C. Bizer, T. Heath, and T. Berners-Lee. "Linked data-the story so far". In: *International Journal on Semantic Web and Information Systems* 5.3 (2009), pp. 1–22.

[18] S. Bloehdorn and Y. Sure. "Kernel methods for mining instance data in ontologies". In: *Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference*. ISWC'07/ASWC'07. Springer Berlin Heidelberg, 2007, pp. 58–71.

[19] A. Bordes, J. Weston, R. Collobert, and Y. Bengio. "Learning structured embeddings of knowledge bases". In: *Proceedings of the 25th Conference on Artificial Intelligence*. AAAI'11. San Francisco, USA, 2011. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI11/paper/viewPDFInterstitial/3659/3898.

[20] L. Bottou and O. Bousquet. "The Tradeoffs of Large Scale Learning". In: *Advances in Neural Information Processing Systems*. Ed. by J. C. Platt, D. Koller, Y. Singer, and S. Roweis. Vol. 20. NIPS'07. Cambridge, MA, USA: MIT Press, 2008, pp. 161–168.

[21] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN: 0521833787. URL: http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20&path=ASIN/0521833787.

[22] S. Brin and L. Page. "The anatomy of a large-scale hypertextual Web search engine". In: *Computer networks and ISDN systems* 30.1 (1998), pp. 107–117. URL: http://www.sciencedirect.com/science/article/pii/S016975529800110X.

[23] A. Buluç, J. T. Fineman, M. Frigo, J. R. Gilbert, and C. E. Leiserson. "Parallel sparse matrix-vector and matrix-transpose-vector multiplication using compressed sparse blocks". In: *Proceedings of the twenty-first annual symposium on Parallelism in algorithms and architectures*. SPAA '09. New York, NY, USA: ACM, 2009, pp. 233–244. ISBN: 978-1-60558-606-9. DOI: 10.1145/1583991.1584053. URL: http://doi.acm.org/10.1145/1583991.1584053.

[24] D. S. Burdick. "An introduction to tensor products with applications to multiway data analysis". In: *Chemometrics and intelligent laboratory systems* 28.2 (1995), pp. 229–237. URL: http://www.sciencedirect.com/science/article/pii/016974399580060M.

[25] E. J. Candès and Y. Plan. "Matrix completion with noise". In: *Proceedings of the IEEE* 98.6 (2010), pp. 925–936. ISSN: 0018-9219. DOI: 10.1109/JPROC.2009.2035722. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5454406.

[26] E. J. Candès and B. Recht. "Exact matrix completion via convex optimization". In: *Foundations of Computational Mathematics* 9.6 (2009), pp. 717–772. ISSN: 1615-3375. DOI: 10.1007/s10208-009-9045-5. URL: http://www.springerlink.com/index/w18k8252q4548477.pdf.

[27] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. "Toward an architecture for never-ending language learning". In: *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*. Vol. 2. 2010, pp. 3–3. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/download/1879/2201.

[28] J. D. Carroll and J. J. Chang. "Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition". In: *Psychometrika* 35.3 (1970), pp. 283–319. ISSN: 0033-3123.

[29] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu. "SVDFeature: A Toolkit for Feature-based Collaborative Filtering". In: *Journal of Machine Learning Research* 13 (2012), pp. 3619–3622. URL: http://jmlr.csail.mit.edu/papers/volume13/chen12a/chen12a.pdf.

[30] D. M. Chickering. "Learning Bayesian networks is NP-complete". In: *Learning from data*. Springer, 1996, pp. 121–130. URL: http://link.springer.com/chapter/10.1007/978-1-4612-2404-4_12.

[31] A. Clauset, C. R. Shalizi, and M. E. Newman. "Power-law distributions in empirical data". In: *SIAM review* 51.4 (2009), pp. 661–703. URL: http://epubs.siam.org/doi/abs/10.1137/070710111.

[32] E. F. Codd. "A relational model of data for large shared data banks". In: *Pioneers and Their Contributions to Software Engineering*. Springer, 2001, pp. 61–98. URL: http://link.springer.com/chapter/10.1007/978-3-642-48354-7_4.

[33] R. Cyganiak and A. Jentzsch. *The Linking Open Data cloud diagram*. Sept. 2011. URL: http://lod-cloud.net/ (accessed 05/29/2013).

[34] C. d'Amato, N. Fanizzi, and F. Esposito. "Non-parametric Statistical Learning Methods for Inductive Classifiers in Semantic Knowledge Bases". In: *Proceedings of the 2008 IEEE International Conference on Semantic Computing*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 291–298. ISBN: 978-0-7695-3279-0. DOI: 10.1109/ICSC.2008.28. URL: http://dl.acm.org/citation.cfm?id=1446294.1446383.

[35] J. Davis and M. Goadrich. "The relationship between Precision-Recall and ROC curves". In: *Proceedings of the 23rd International Conference on Machine learning*. ICML'06. New York, NY, USA: ACM, 2006, pp. 233–240. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143874.

[36] J. Davis, E. Burnside, I. Dutra, D. Page, R. Ramakrishnan, V. S. Costa, and J. Shavlik. "View learning for statistical relational learning: With an application to mammography". In: *Proceeding of the 19th International Joint Conference on Artificial Intelligence. Edinburgh, Scotland*. 2005. URL: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA457195#page=205.

[37] J. Davis and P. Domingos. "Bottom-up learning of Markov network structure". In: *Proceedings of the 27th International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 271–280. URL: http://www.icml2010.org/papers/537.pdf.

[38] L. De Lathauwer. "Decompositions of a higher-order tensor in block terms—Part II: Definitions and uniqueness". In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (2008), pp. 1033–1066.

[39] L. De Lathauwer, B. De Moor, and J. Vandewalle. "A multilinear singular value decomposition". In: *SIAM Journal on Matrix Analysis and Applications* 21.4 (2000), pp. 1253–1278. URL: http://epubs.siam.org/doi/abs/10.1137/S0895479896305696.

[40] L. De Lathauwer, B. De Moor, and J. Vandewalle. "On the Best Rank-1 and Rank-(R1, R2,. . .,RN) Approximation of Higher-Order Tensors". In: *SIAM J. Matrix Anal. Appl.* 21.4 (2000), pp. 1324–1342. ISSN: 0895-4798. DOI: 10.1137/S0895479898346995. URL: http://dx.doi.org/10.1137/S0895479898346995.

[41] V. De Silva and L.-H. Lim. "Tensor rank and the ill-posedness of the best low-rank approximation problem". In: *SIAM Journal on Matrix Analysis and Applications* 30.3 (2008), pp. 1084–1127. URL: http://epubs.siam.org/doi/abs/10.1137/06066518X.

[42] W. Denham. "The detection of patterns in Alyawarra nonverbal behavior". PhD thesis. Seattle, WA: University of Washington, 1973.

[43] N. Ding and S. V. N. Vishwanathan. "t-logistic regression". In: *Advances in Neural Information Processing Systems 23*. Ed. by J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta. 2010, pp. 514–522.

[44] G. Dror, N. Koenigstein, Y. Koren, and M. Weimer. "The yahoo! music dataset and kdd-cup'11". In: *KDD-Cup Workshop*. Vol. 2011. 2011. URL: http://jmlr.csail.mit.edu/proceedings/papers/v18/dror12a/dror12a.pdf.

[45] S. Dzeroski. "Relational data mining applications: an overview". In: *Relational Data Mining*. 2001, pp. 339–360. URL: http://dl.acm.org/citation.cfm?id=567240.

[46] A. Fader, S. Soderland, and O. Etzioni. "Identifying relations for open information extraction". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 1535–1545. ISBN: 978-1-937284-11-4. URL: http://dl.acm.org/citation.cfm?id=2145432.2145596.

[47] N. Fanizzi, C. D'Amato, and F. Esposito. "DL-FOIL Concept Learning in Description Logics". In: *Proceedings of the 18th international conference on Inductive Logic Programming*. ILP '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 107–121. ISBN: 978-3-540-85927-7. DOI: 10.1007/978-3-540-85928-4_12. URL: http://dx.doi.org/10.1007/978-3-540-85928-4_12.

[48] A. Ferrara, D. Lorusso, S. Montanelli, and G. Varese. "Towards a Benchmark for Instance Matching." In: *Ontology Matching*. Ed. by P. Shvaiko, J. Euzenat, F. Giunchiglia, and H. Stuckenschmidt. Vol. 431. OM'08. CEUR Workshop Proceedings, 2008.

[49] A. Ferrara, A. Nikolov, and F. Scharffe. "Data Linking for the Semantic Web". In: *International Journal on Semantic Web and Information Systems* 7.3 (2011), pp. 46–76. ISSN: 1552-6283, 1552-6291. DOI: 10.4018/jswis.2011070103. URL: http://www.igi-global.com/article/data-linking-semantic-web/62562.

[50] S. Fortunato. "Community detection in graphs". In: *Physics Reports* 486.3–5 (Feb. 2010), pp. 75–174. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2009.11.002. URL: http://www.sciencedirect.com/science/article/pii/S0370157309002841.

[51] T. Franz, A. Schultz, S. Sizov, and S. Staab. "Triplerank: Ranking semantic web data by tensor decomposition". In: *The Semantic Web-ISWC 2009* (2009), pp. 213–228.

[52] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. "Learning probabilistic relational models". In: *International Joint Conference on Artificial Intelligence*. Vol. 16. Stockholm, Sweden: Morgan Kaufman, 1999, pp. 1300–1309. URL: http://www.robotics.stanford.edu/~koller/Papers/Friedman+al:IJCAI99.pdf.

[53] S. Gandy, B. Recht, and I. Yamada. "Tensor completion and low-n-rank tensor recovery via convex optimization". In: *Inverse Problems* 27.2 (2011), p. 025010. URL: http://iopscience.iop.org/0266-5611/27/2/025010.

[54] L. Getoor and B. Taskar, eds. *Introduction to statistical relational learning*. MIT Press, 2007. ISBN: 0262072882. URL: http://www.cs.umd.edu/srl-book/.

[55] L. Getoor and C. P. Diehl. "Link mining: a survey". In: *ACM SIGKDD Explorations Newsletter* 7.2 (2005), pp. 3–12. URL: http://dl.acm.org/citation.cfm?id=1117456.

[56] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. "Probabilistic Relational Models". In: *Introduction to statistical relational learning*. Ed. by L. Getoor and B. Taskar. Cambridge, MA, USA: MIT Press, 2007, pp. 129–174. URL: http://www.cs.umd.edu/srl-book/.

[57] G. H. Golub and C. F. V. Loan. *Matrix Computations*. en. Fourth Edition. JHU Press, 2013. ISBN: 1421407949.

[58] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Third Edition. Johns Hopkins University Press, 1996. ISBN: 0-8018-5414-8. URL: http://www.cs.cornell.edu/cv/Books/GVL/.

[59] P. R. Halmos. *Naive set theory*. Springer, 1998.

[60] H. Halpin, P. Hayes, J. McCusker, D. Mcguinness, and H. Thompson. "When owl: sameAs isn't the same: An analysis of identity in linked data". In: *The Semantic Web–ISWC 2010* (2010), pp. 305–320.

[61] R. A. Harshman. "Models for analysis of asymmetrical relationships among N objects or stimuli". In: *First Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology, McMaster University, Hamilton, Ontario, August*. 1978.

[62] R. A. Harshman. "Foundations of the PARAFAC procedure: models and conditions for an" explanatory" multimodal factor analysis". In: *UCLA Working Papers in Phonetics* 16 (1970). URL: http://www.psychology.uwo.ca/faculty/harshman/wpppfac0.pdf.

[63] M. H. Hassoun. *Fundamentals of Artificial Neural Networks*. en. MIT Press, 1995. ISBN: 9780262082396.

[64] J. Håstad. "Tensor rank is NP-complete". In: *Journal of Algorithms* 11.4 (1990), pp. 644–654. ISSN: 0196-6774. DOI: 10.1016/0196-6774(90)90014-6. URL: http://www.sciencedirect.com/science/article/pii/0196677490900146.

[65] D. Heckerman, C. Meek, and D. Koller. "Probabilistic Entity-Relationship Models, PRMs, and Plate Models". In: *Introduction to statistical relational learning*. Ed. by L. Getoor and B. Taskar. Cambridge, MA, USA: MIT Press, 2007, pp. 201–238. URL: http://www.cs.umd.edu/srl-book/.

[66] D. Heckerman, C. Meek, and D. Koller. *Probabilistic models for relational data*. Tech. rep. Technical Report MSR-TR-2004-30, Microsoft Research, 2004. URL: http://research.microsoft.com/pubs/70050/tr-2004-30.pdf.

[67] S. Hellmann, J. Lehmann, and S. Auer. "Learning of OWL class descriptions on very large knowledge bases". In: *Int. J. Semantic Web Inf. Syst* 5.2 (2009), pp. 25–48.

[68] F. L. Hitchcock. "The expression of a tensor or a polyadic as a sum of products". In: *Journal of Mathematics and Physics* 6 (1927), pp. 39–79.

[69] P. Hitzler and F. van Harmelen. "A reasonable semantic web". In: *Semantic Web* 1.1 (2010), pp. 39–44.

[70] P. D. Hoff, A. E. Raftery, and M. S. Handcock. "Latent space approaches to social network analysis". In: *Journal of the American Statistical Association* 97.460 (2002), pp. 1090–1098. URL: http://pubs.amstat.org/doi/abs/10.1198/016214502388618906.

[71] P. D. Hoff. "Modeling homophily and stochastic equivalence in symmetric relational data". In: *Advances in Neural Information Processing Systems*. Vol. 20. NIPS'07. Cambridge, MA, USA: MIT Press, 2008, pp. 657–664.

[72] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum. "YAGO2: exploring and querying world knowledge in time, space, context, and many languages". In: *Proceedings of the 20th international conference companion on World wide web*. WWW '11. New York, NY, USA: ACM, 2011, pp. 229–232. ISBN: 978-1-4503-0637-9. DOI: 10.1145/1963192.1963296. URL: http://doi.acm.org/10.1145/1963192.1963296.

[73] A. Hogan, A. Harth, A. Passant, S. Decker, and A. Polleres. "Weaving the pedantic web". In: *Proceedings of the WWW2010 Workshop on Linked Data on the Web*. Vol. Linked Data on the Web. LDOW'10. Raleigh, North Carolina, USA: CEUR Workshop Proceedings, 2010. URL: CEUR-WS.org/Vol-628/.

[74] Y. Huang, M. Nickel, V. Tresp, and H.-P. Kriegel. "A Scalable Kernel Approach to Learning in Semantic Graphs with Applications to Linked Data". In: *Proceedings of the 1st Workshop on Mining the Future Internet*. CEUR Workshop Proceedings, 2010. URL: http://www.csw.inf.fu-berlin.de/mifi2010/.

[75] Y. Huang, V. Tresp, M. Bundschus, A. Rettinger, and H.-P. Kriegel. "Multivariate prediction for learning on the semantic web". In: *Proceedings of the 20th international conference on Inductive logic programming*. ILP'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 92–104. ISBN: 978-3-642-21294-9. URL: http://dl.acm.org/citation.cfm?id=2022735.2022750.

[76] Y. Huang, V. Tresp, M. Nickel, A. Rettinger, and H.-P. Kriegel. "A scalable approach for statistical learning in semantic graphs". In: *Semantic Web* (2013), to appear. DOI: 10.3233/SW-130100. URL: http://dx.doi.org/10.3233/SW-130100.

[77] R. Jenatton, N. Le Roux, A. Bordes, and G. Obozinski. "A latent factor model for highly multi-relational data". In: *Advances in Neural Information Processing Systems*. Ed. by P. Bartlett, F. C. N. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger. Vol. 25. NIPS'12. Lake Tahoe, Nevada, USA: MIT Press, 2012, pp. 3176–3184.

[78] D. Jensen and J. Neville. "Linkage and Autocorrelation Cause Feature Selection Bias in Relational Learning". In: *Proceedings of the Nineteenth International Conference on Machine Learning*. ICML '02. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 259–266. ISBN: 1-55860-873-7. URL: http://dl.acm.org/citation.cfm?id=645531.655828.

[79] D. Jensen, J. Neville, and B. Gallagher. "Why collective inference improves relational classification". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '04. New York, NY, USA: ACM, 2004, pp. 593–598. ISBN: 1-58113-888-1. DOI: 10.1145/1014052.1014125. URL: http://doi.acm.org/10.1145/1014052.1014125.

[80] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. "The large-scale organization of metabolic networks". en. In: *Nature* 407.6804 (Oct. 2000), pp. 651–654. ISSN: 0028-0836. DOI: 10.1038/35036627. URL: http://www.nature.com/nature/journal/v407/n6804/full/407651a0.html.

[81] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001. URL: http://www.scipy.org/.

[82] M. I. Jordan. *Learning in graphical models*. en. MIT Press, 1998. ISBN: 0262600323.

[83] A. Karatzoglou, A. Smola, and M. Weimer. "Collaborative filtering on a budget". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. AISTATS. 2010, pp. 389–396. URL: http://cs.markusweimer.com/pub/2010/2010-AISTATS.pdf.

[84] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. "Learning systems of concepts with an infinite relational model". In: *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*. AAAI'06. Boston, Massachusetts: AAAI

Press, 2006, pp. 381–388. ISBN: 978-1-57735-281-5. URL: http://dl.acm.org/citation.cfm?id=1597538.1597600.

[85] K. Kersting and L. De Raedt. *Bayesian Logic Programs*. Tech. rep. Albert-Ludwigs University at Freiburg, 2001.

[86] K. Kersting and L. De Raedt. "Bayesian Logic Programming: Theory and Tool". In: *Introduction to statistical relational learning*. Ed. by L. Getoor and B. Taskar. Cambridge, MA, USA: MIT Press, 2007, pp. 291–321. URL: http://www.cs.umd.edu/srl-book/.

[87] C. Kiefer, A. Bernstein, and A. Locher. "Adding data mining support to SPARQL via statistical relational learning methods". In: *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*. ESWC'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 478–492. ISBN: 3-540-68233-3, 978-3-540-68233-2.

[88] J. M. Kleinberg. "Authoritative sources in a hyperlinked environment". In: *J. ACM* 46.5 (Sept. 1999), pp. 604–632. ISSN: 0004-5411. DOI: 10.1145/324133.324140. URL: http://doi.acm.org/10.1145/324133.324140.

[89] S. Kok and P. Domingos. "Learning the structure of Markov logic networks". In: *Proceedings of the 22nd international conference on Machine learning*. ICML '05. New York, NY, USA: ACM, 2005, pp. 441–448. ISBN: 1-59593-180-5. DOI: 10.1145/1102351.1102407. URL: http://doi.acm.org/10.1145/1102351.1102407.

[90] S. Kok and P. Domingos. "Statistical predicate invention". In: *Proceedings of the 24th international conference on Machine learning*. ICML '07. New York, NY, USA: ACM, 2007, pp. 433–440. ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273551. URL: http://doi.acm.org/10.1145/1273496.1273551.

[91] S. Kok and P. Domingos. "Learning Markov logic network structure via hypergraph lifting". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. New York, NY, USA: ACM, 2009, pp. 505–512. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553440. URL: http://doi.acm.org/10.1145/1553374.1553440.

[92] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, J. Wang, A. Nath, and P. Domingos. *The Alchemy System for Statistical Relational AI*. Technial Report. Seattle, WA, USA: Department of Computer Science and Engineering, University of Washington, 2005. URL: http://alchemy.cs.washington.edu.

[93] T. G. Kolda and B. W. Bader. "Tensor Decompositions and Applications". In: *SIAM Review* 51.3 (2009), pp. 455–500. ISSN: 00361445. DOI: 10.1137/07070111X. URL: http://link.aip.org/link/SIREAD/v51/i3/p455/s1&Agg=doi.

[94] T. G. Kolda, B. W. Bader, and J. P. Kenny. "Higher-order web link analysis using multilinear algebra". In: *Proceedings of the Fifth International Conference on Data Mining*. IEEE Computer Society, 2005, pp. 242–249.

[95] T. G. Kolda and J. Sun. "Scalable Tensor Decompositions for Multi-aspect Data Mining". In: *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. ICDM '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 363–372. ISBN: 978-0-7695-3502-9. DOI: 10.1109/ICDM.2008.89. URL: http://dx.doi.org/10.1109/ICDM.2008.89.

[96] D. Koller, N. Friedman, L. Getoor, and B. Taskar. "Graphical Models in a Nutshell". In: *Introduction to statistical relational learning*. Ed. by L. Getoor and B. Taskar. Cambridge, MA, USA: MIT Press, 2007, pp. 13–55. URL: http://www.cs.umd.edu/srl-book/.

[97] D. Koller and A. Pfeffer. "Probabilistic frame-based systems". In: *Proceedings of the National Conference on Artificial Intelligence*. 1998, pp. 580–587. URL: http://www.aaai.org/Papers/AAAI/1998/AAAI98-082.pdf.

[98] Y. Koren. "The bellkor solution to the netflix grand prize". In: *Netflix prize documentation* (2009). URL: http://www.stat.osu.edu/~dmsl/GrandPrize2009_BPC_BellKor.pdf.

[99] S. Kramer, N. Lavrac, and P. Flach. *Propositionalization approaches to relational data mining*. Springer-Verlag New York, Inc., 2001. URL: http://dl.acm.org/citation.cfm?id=567236.

[100] P. M. Kroonenberg. *Applied multiway data analysis*. Wiley-Interscience, 2008. ISBN: 978-0-470-16497-6. URL: http://three-mode.leidenuniv.nl/.

[101] P. M. Kroonenberg and J. De Leeuw. "Principal component analysis of three-mode data by means of alternating least squares algorithms". In: *Psychometrika* 45.1 (1980), pp. 69–97. URL: http://link.springer.com/article/10.1007/BF02293599.

[102] K. Lange. *Numerical analysis for statisticians*. Springer Verlag, 2010.

[103] A. J. Laub. *Matrix analysis for scientists and engineers*. Society for Industrial and Applied Mathematics, 2004.

[104] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. "Kronecker Graphs: An Approach to Modeling Networks". In: *Journal of Machine Learning Research* 11 (2010), pp. 985–1042.

[105] D. Liben-Nowell and J. Kleinberg. "The link-prediction problem for social networks". In: *Journal of the American society for information science and technology* 58.7 (2007), pp. 1019–1031. URL: http://onlinelibrary.wiley.com/doi/10.1002/asi.20591/full.

[106] H. T. Lin, N. Koul, and V. Honavar. "Learning relational bayesian classifiers from RDF data". In: *Proceedings of the 10th international conference on The semantic web - Volume Part I*. ISWC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 389–404. ISBN: 978-3-642-25072-9.

[107] J. Liu, P. Musialski, P. Wonka, and J. Ye. "Tensor completion for estimating missing values in visual data". In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE Computer Society, 2009, pp. 2114–2121. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5459463.

[108] Q. Liu and A. T. Ihler. "Learning scale free networks by reweighted l1 regularization". In: *International Conference on Artificial Intelligence and Statistics*. 2011, pp. 40–48.

[109] B. London, T. Rekatsinas, B. Huang, and L. Getoor. "Multi-relational Learning Using Weighted Tensor Decomposition with Modular Loss". In: *arXiv preprint arXiv:1303.1733* (2013). URL: http://arxiv.org/abs/1303.1733.

[110] C. D. Meyer, ed. *Matrix analysis and applied linear algebra*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2000. ISBN: 0-89871-454-0.

[111] T. P. Minka. *Old and new matrix algebra useful for statistics*. MIT Media Lab Note. 2000.

[112] M. Mohri and A. Rostamizadeh. "Rademacher complexity bounds for non-iid processes". In: *Advances in Neural Information Processing Systems*. Vol. 21. NIPS. Cambridge, MA, USA: MIT Press, 2009, pp. 1097–1104. URL: http://www.cs.nyu.edu/~rostami/papers/rad.pdf.

[113] K. P. Murphy, Y. Weiss, and M. I. Jordan. "Loopy belief propagation for approximate inference: An empirical study". In: *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*. 1999, pp. 467–475. URL: http://dl.acm.org/citation.cfm?id=2073849.

[114] N. Nakashole, G. Weikum, and F. Suchanek. "Discovering and exploring relations on the web". In: *Proceedings of the VLDB Endowment* 5.12 (2012), pp. 1982–1985.

[115] N. Nakashole, G. Weikum, and F. Suchanek. "PATTY: a taxonomy of relational patterns with semantic types". In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 2012, pp. 1135–1145.

[116] A. Neumaier. "Solving ill-conditioned and singular linear systems: A tutorial on regularization". In: *SIAM Review* 40.3 (1998), pp. 636–666. URL: http://epubs.siam.org/doi/abs/10.1137/S0036144597321909.

[117] J. Neville and D. Jensen. "Collective classification with relational dependency networks". In: *Proceedings of the Second International Workshop on Multi-Relational Data Mining*. 2003, pp. 77–91.

[118] J. Neville and D. Jensen. "Relational dependency networks". In: *The Journal of Machine Learning Research* 8 (2007), pp. 653–692.

[119] A. Y. Ng, M. I. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm". In: *Advances in neural information processing systems*. Vol. 2. NIPS'01. MIT Press, 2002, pp. 849–856.

[120] M. Nickel and V. Tresp. "Three-Way DEDICOM for Relational Learning". In: *NIPS 2010 Workshop - Tensors, Kernels and Machine Learning*. Whistler, Canada, 2010. URL: http://csmr.ca.sandia.gov/~dfgleic/tkml2010/.

[121] M. Nickel and V. Tresp. "Learning Taxonomies from Multi-Relational Data via Hierarchical Link-Based Clustering". In: *NIPS Workshop - Learning Semantics*. Granada, Spain, 2011. URL: http://learningsemanticsnips2011.wordpress.com/.

[122] M. Nickel and V. Tresp. *Machine Learning on Linked Data: Tensors and their Applications in Graph-Structured Domains*. Tutorial at the 11th International Semantic Web Conference. Boston, MA, USA, 2012. URL: http://www.cip.ifi.lmu.de/~nickel/iswc2012-learning-on-linked-data/.

[123] M. Nickel and V. Tresp. "An Analysis of Tensor Models for Learning on Structured Data". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013*. Ed. by H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezny. ECML/PKDD'13. Springer, 2013, to appear.

[124] M. Nickel and V. Tresp. "Logistic Tensor-Factorization for Multi-Relational Data". In: *ICML Workshop - Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*. Atlanta, GA, USA, 2013.

[125] M. Nickel and V. Tresp. "Tensor Factorization for Multi-Relational Learning". In: *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2013*. Ed. by H. Blockeel, K. Kersting, S. Nijssen, and F. Zelezny. Nectar track for "high-quality research related to machine learning". Springer, 2013, to appear.

[126] M. Nickel, V. Tresp, and H.-P. Kriegel. "A Three-Way Model for Collective Learning on Multi-Relational Data". In: *Proceedings of the 28th International Conference on Machine Learning*. ICML '11. Bellevue, WA, USA: ACM, 2011, pp. 809–816. ISBN: 978-1-4503-0619-5. URL: http://www.icml-2011.org.

[127] M. Nickel, V. Tresp, and H.-P. Kriegel. "Factorizing YAGO: scalable machine learning for linked data". In: *Proceedings of the 21st international conference on World Wide Web*. WWW '12. New York, NY, USA: ACM, 2012, pp. 271–280. ISBN: 978-1-4503-1229-5. DOI: 10.1145/2187836.2187874. URL: http://doi.acm.org/10.1145/2187836.2187874.

[128] T. E. Oliphant. "Python for Scientific Computing". In: *Computing in Science & Engineering* 9.3 (2007), pp. 10–20. URL: http://link.aip.org/link/?CSX/9/10/1.

[129] H. Pasula and S. Russell. "Approximate inference for first-order probabilistic languages". In: *Proceedings of the 17th international joint conference on Artificial intelli-*

gence - Volume 1. IJCAI'01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 741–748. ISBN: 1-55860-812-5, 978-1-558-60812-2. URL: http://dl.acm.org/citation.cfm?id=1642090.1642190.

[130] J. Pearl. *Causality: Models, Reasoning and Inference*. 2nd Revised edition. Cambridge University Press, 2009. ISBN: 052189560X.

[131] J. Platt. "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods". In: *Advances in large margin classifiers* 10.3 (1999), pp. 61–74.

[132] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. 3rd ed. New York, NY, USA: Cambridge University Press, 2007. ISBN: 0521880688, 9780521880688.

[133] L. d. Raedt. *Logical and relational learning*. en. Springer, 2008. ISBN: 3540688560.

[134] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. "Factorizing personalized Markov chains for next-basket recommendation". In: *Proceedings of the 19th international conference on World Wide Web*. ACM, 2010, pp. 811–820.

[135] A. Rettinger, H. Wermser, Y. Huang, and V. Tresp. "Context-aware tensor decomposition for relation prediction in social networks". en. In: *Social Network Analysis and Mining* 2.4 (2012), pp. 373–385. ISSN: 1869-5450, 1869-5469. DOI: 10.1007/s13278-012-0069-5. URL: http://link.springer.com/article/10.1007/s13278-012-0069-5.

[136] M. Richardson and P. Domingos. "Markov logic networks". In: *Machine Learning* 62.1 (2006), pp. 107–136. ISSN: 0885-6125.

[137] D. Roy, C. Kemp, V. Mansinghka, and J. Tenenbaum. "Learning annotated hierarchies from relational data". In: *Advances in neural information processing systems*. Vol. 19. NIPS'06. Cambridge, MA, USA: MIT Press, 2007, pp. 1185–1192. ISBN: 0-262-19568-2.

[138] R. J. Rummel. *Dimensionality of Nations Project: Attributes of Nations and Behavior of Nation Dyads, 1950-1965*. 1999. URL: http://dx.doi.org/10.3886/ICPSR05409.v1.

[139] A. Ruttenberg, J. A. Rees, M. Samwald, and M. S. Marshall. "Life sciences on the Semantic Web: the Neurocommons and beyond". en. In: *Briefings in Bioinformatics* 10.2 (Mar. 2009). PMID: 19282504, pp. 193–204. ISSN: 1467-5463, 1477-4054. DOI: 10.1093/bib/bbp004. URL: http://bib.oxfordjournals.org/content/10/2/193.

[140] R. Salakhutdinov and A. Mnih. "Probabilistic matrix factorization". In: *Advances in neural information processing systems*. Vol. 20. Cambridge, MA, USA: MIT Press, 2008, pp. 1257–1264.

[141] R. Salakhutdinov and A. Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo". In: *Proceedings of the 25th international conference on Machine learning*. ICML '08. New York, NY, USA: ACM, 2008, pp. 880–887. ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390267. URL: http://doi.acm.org/10.1145/1390156.1390267.

[142] S. Sarkar and A. Dong. "Community detection in graphs using singular value decomposition". In: *Physical Review E* 83.4 (Apr. 2011), p. 046114. DOI: 10.1103/PhysRevE.83.046114. URL: http://link.aps.org/doi/10.1103/PhysRevE.83.046114.

[143] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. "Collective classification in network data". In: *AI Magazine* 29.3 (2008), pp. 93–106. ISSN: 0738-4602.

[144] M. Signoretto, R. Van de Plas, B. De Moor, and J. A. Suykens. "Tensor versus matrix completion: a comparison with application to spectral data". In: *Signal Processing Letters, IEEE* 18.7 (2011), pp. 403–406. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5764499.

[145] A. P. Singh and G. J. Gordon. "A Unified View of Matrix Factorization Models". In: *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*. ECML PKDD '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 358–373. ISBN: 978-3-540-87480-5. DOI: 10.1007/978-3-540-87481-2_24. URL: http://dx.doi.org/10.1007/978-3-540-87481-2_24.

[146] A. P. Singh and G. J. Gordon. "Relational learning via collective matrix factorization". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. New York, NY, USA: ACM, 2008, pp. 650–658. ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401969. URL: http://doi.acm.org/10.1145/1401890.1401969.

[147] P. Singla and P. Domingos. "Entity Resolution with Markov Logic". In: *Proceedings of the Sixth International Conference on Data Mining*. ICDM '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 572–582. ISBN: 0-7695-2701-9. DOI: 10.1109/ICDM.2006.65. URL: http://dx.doi.org/10.1109/ICDM.2006.65.

[148] P. Singla and P. Domingos. "Memory-efficient inference in relational domains". In: *Proceedings of the 21st National Conference on Artificial intelligence - Volume 1*. AAAI'06. Boston, Massachusetts: AAAI Press, 2006, pp. 488–493. ISBN: 978-1-57735-281-5. URL: http://dl.acm.org/citation.cfm?id=1597538.1597617.

[149] P. Singla and P. Domingos. "Lifted first-order belief propagation". In: *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*. AAAI'08. Chicago, Illinois: AAAI Press, 2008, pp. 1094–1099. ISBN: 978-1-57735-368-3. URL: http://dl.acm.org/citation.cfm?id=1620163.1620242.

[150] A. Smilde, R. Bro, and P. Geladi. *Multi-way analysis: applications in the chemical sciences*. Wiley, 2005.

[151] S. Sra, S. Nowozin, and S. J. Wright. *Optimization for machine learning*. en. Cambridge, MA, USA: MIT Press, 2012. ISBN: 9780262016469.

[152] N. Srebro. "Learning with matrix factorizations". PhD thesis. Cambridge, MA, USA: Massachusetts Institute of Technology, 2004. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.146.4320&rep=rep1&type=pdf.

[153] N. Srebro, N. Alon, and T. S. Jaakkola. "Generalization Error Bounds for Collaborative Prediction with Low-Rank Matrices". In: *Advances in Neural Information Processing Systems*. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. Vol. 17. Cambridge, MA: MIT Press, 2005, pp. 1321–1328.

[154] F. M. Suchanek, G. Kasneci, and G. Weikum. "Yago: a core of semantic knowledge". In: *Proceedings of the 16th international conference on World Wide Web*. WWW '07. New York, NY, USA: ACM, 2007, pp. 697–706. ISBN: 978-1-59593-654-7. DOI: 10.1145/1242572.1242667. URL: http://doi.acm.org/10.1145/1242572.1242667.

[155] I. Sutskever, R. Salakhutdinov, and J. Tenenbaum. "Modelling Relational Data using Bayesian Clustered Tensor Factorization". In: *Advances in Neural Information Processing Systems 22*. Ed. by Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta. NIPS'09. 2009, pp. 1821–1828.

[156] P. Tan, M. Steinbach, V. Kumar, et al. *Introduction to data mining*. Pearson Addison Wesley Boston, 2006.

[157] B. Taskar, P. Abbeel, and D. Koller. "Discriminative probabilistic models for relational data". In: *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*. 2002, pp. 485–492. URL: http://dl.acm.org/citation.cfm?id=2073934.

[158] B. Taskar, M.-F. Wong, P. Abbeel, and D. Koller. "Link Prediction in Relational Data". In: *Advances in Neural Information Processing Systems*. Ed. by S. Thrun, L. Saul, and B. Schölkopf. Vol. 16. NIPS'03. Cambridge, MA: MIT Press, 2004.

[159] Y. W. Teh. "A hierarchical Bayesian language model based on Pitman-Yor processes". In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. ACL-44. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006, pp. 985–992. DOI: 10.3115/1220175.1220299. URL: http://dx.doi.org/10.3115/1220175.1220299.

[160] R. Tomioka, K. Hayashi, and H. Kashima. "Estimation of low-rank tensors via convex optimization". In: *arXiv preprint arXiv:1010.0789* (2010). URL: http://arxiv.org/abs/1010.0789.

[161] R. Tomioka, T. Suzuki, K. Hayashi, and H. Kashima. "Statistical performance of convex tensor decomposition". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor, R. S. Zemel, P. Bartlett, F. C. N. Pereira, and K. Q. Weinberger. Vol. 24. NIPS'11. 2012, pp. 972–980. URL: http://books.nips.cc/papers/files/nips24/NIPS2011_0596.pdf.

[162] L. N. Trefethen and D. Bau III. *Numerical linear algebra*. 50. Society for Industrial and Applied Mathematics, 1997.

[163] V. Tresp, Y. Huang, and M. Nickel. "Querying the Web with Statistical Machine Learning". In: *To be published as a chapter in a book reviewing the results of the THESEUS project*. 2013.

[164] V. Tresp and M. Nickel. "Relational Models". In: *Encyclopedia of Social Network Analysis and Mining*. Ed. by J. Rokne and R. Alhajj. Heidelberg: Springer, 2013, to appear.

[165] L. R. Tucker. "Some mathematical notes on three-mode factor analysis". In: *Psychometrika* 31.3 (1966), pp. 279–311. ISSN: 0033-3123.

[166] J. Van Haaren and J. Davis. "Markov network structure learning: A randomized feature generation approach". In: *Proceedings of the Twenty-Sixth National Conference on Artificial Intelligence. AAAI Press*. 2012. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/viewFile/5107/5534.

[167] J. Voelker and M. Niepert. "Statistical schema induction". In: *The Semantic Web: Research and Applications* (2011), pp. 124–138.

[168] M. J. Wainwright and M. I. Jordan. "Graphical Models, Exponential Families, and Variational Inference". In: *Foundations and Trends® in Machine Learning* 1.1–2 (2007), pp. 1–305. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000001. URL: http://www.nowpublishers.com/product.aspx?product=MAL&doi=2200000001.

[169] H. E. Warren. "Lower Bounds for Approximation by Nonlinear Manifolds". In: *Transactions of the American Mathematical Society* 133.1 (Aug. 1968), pp. 167–178. ISSN: 00029947. DOI: 10.2307/1994937. URL: http://www.jstor.org/discover/10.2307/1994937?uid=3737864&uid=2&uid=4&sid=21102114782543.

[170] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. "Feature hashing for large scale multitask learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. New York, NY, USA: ACM, 2009, pp. 1113–1120. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553516. URL: http://doi.acm.org/10.1145/1553374.1553516.

[171] S. E. Whang and H. Garcia-Molina. "Joint Entity Resolution". In: *Proceedings of the 28th IEEE International Conference on Data Engineering*. ICDE '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 294–305. ISBN: 978-0-7695-4747-3. DOI: 10.1109/ICDE.2012.119. URL: http://dx.doi.org/10.1109/ICDE.2012.119.

[172] P. Winston. “Learning structural descriptions from example”. In: *The Psychology of Computer Vision*. Ed. by P. Winston. New York, NY, USA: McGraw-Hill, 1975, pp. 157–209.

[173] L. Wolf, H. Jhuang, and T. Hazan. “Modeling Appearances with Low-Rank SVM”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. CVPR’07. IEEE Computer Society, 2007, pp. 1–6. DOI: 10.1109/CVPR.2007.383099.

[174] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. G. Carbonell. “Temporal collaborative filtering with bayesian probabilistic tensor factorization”. In: *Proceedings of SIAM Data Mining*. Vol. 2010. 2010, pp. 211–222. URL: http://www.cs.cmu.edu/~./xichen/images/Xi%20Chen%20SDM%202010.pdf.

[175] Z. Xu, V. Tresp, K. Yu, and H.-P. Kriegel. “Infinite Hidden Relational Models”. In: *Proceedings of the Twenty-Second Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*. Arlington, Virginia: AUAI Press, 2006, pp. 544–551.

[176] Z. Xu, V. Tresp, S. Yu, K. Yu, and H.-P. Kriegel. “Fast inference in infinite hidden relational models”. In: *Working Notes of the 5th International Workshop on Mining and Learning with Graphs (MLG’07), Florence, Italy*. 2007. URL: http://www.tresp.org/papers/ihrm_mlg07.pdf.