Integrating Prior Knowledge into Factorization Approaches for Relational Learning

Xueyan Jiang

München, 2014

Integrating Prior Knowledge into Factorization Approaches for Relational Learning

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

eingereicht von
Xueyan Jiang
aus Guangdong, China

München, 2014

1.Gutachter: Prof. Dr. Volker Tresp

2.Gutachter: Prof. Dr. Rudolf Kruse

Tag der mündlichen Prüfung: 16. Dezember 2014

Contents

List of Figures

List of Tables

List of Algorithms

Acknowledgements

Along my way to this thesis, I received a lot of support and encouragement. Hereby, I would like to express my deep gratitude.

First of all, I am grateful to my advisor, Prof. Dr. Volker Tresp. Volker is the most humorous advisor that I know. His bright smile always can influence people around him. He is enthusiastic and energetic in getting into the forefront of the research on machine learning. His open-minded attitude towards new scientific trends, together with his deep experience in scientific research and the accumulated wisdom let him walk quickly to new scientific approaches. Many insightful discussions with Volker deepen my understanding on my research, and bring a broader perspective of my research applications. I would Thank Volker again for all his guidance, his patience on my way of research, and his valuable and detailed advices on this thesis.

I also thank a lot to Prof. Dr. Rudolf Kruse. Prof. Kruse is the tutor of my master intern, the supervisor of my master thesis, and now he is the second advisor of this thesis. Prof. Kruse is very friendly and cheerful to his students. He brought me to the world of intelligent data analysis and machine learning. In his lectures he emphasizes the importance of both solid theoretical knowledge and the applied theory into practice. I am very grateful to Prof. Kruse that he recommended me to Siemens for my intern, to reinforce my understanding of some machine learning theory through practice. Thanks again to Prof. Kruse for all his support over the years and for supervising this thesis.

At the Ludwig Maximilian University, I would like to thank Prof. Hans-Peter Kriegel and Dr. Matthias Schubert for their support from the faculty side. I would also like to thank Prof. Dr. Hans Jürgen Ohlbach, Prof. Dr. Andreas Butz, Prof. Dr. Florian Alt as members of my promotion committee. At Siemens CT where I work for most of the time during my PhD, I would also like to thank my PhD colleague Dr. Maximilian Nickel. Max is my mentor. He has an amazing ability to theorize the problems and he is always creative in doing things. I am very grateful to him for handing me his code so that I could try new algorithms based on it. And I also want to thank the laboratory colleagues, Yi Huang, Denis Krompass, Cristobal Esteban, Stephan Baier, Jonathan Boidol and Alev Canoglu. They create a good atmosphere at the discussions in the regular meetings and at lunch time. Additionally, many thanks to Dr. Clemens Otte. Clemens is the supervisor of my master intern at the Siemens side. Thanks for his guidance during my intern. Through the project on Yaw error prediction for Wind turbines, Clemens has inducted me into useful practical skills and experience of dealing with real data. I would also like to thank Christian Moewes, for all his help through my master time and for his continued support. Moreover, I appreciate appreciate our research group manager Michal Skubacz to support my travels to ESWC, ECML, and NIPS. I appreciate my time with Stefan Hagen Weber in the nice office at Siemens. I appreciate the colleagues at Siemens side, Christa Singer, Bianca Starke, Martina Sommer and Ulrike Robeck, for their help through many official procedures. I would also like to thank Heiner Oberkampf, Sigurd Spieckermann, Nina Solomakhina, Markus Bundschus, Sebastian Mechs, Achim Rettinger, Mohammed Yahya, Yiyu Li, Isabelle Augenstein, Prof. Dr. Heiko Paulheim, Stephan Spiegel, Nan Li, Franz Krojer, Yi Zhong, Meiwen Jia, Pasquale Minervini, Werner Hauptmann, Thorsten Führing, Christian Kleegrewe, Kai Heesche, Sebnem Rusitschka, Thomas Hubauer, and many other friends that I got to know in different research activities or interesting cooperation.

I especially would like to thank my families for their love, tolerance and encouragement. My father Yazhu influences me to be optimistic and be positive. My mother Zhiying encourages me to be brave and educates me to be a content person. My brother Jianshan is an outstanding architect. He teaches me to be patient to dig into problems. My sister-in-law Caixia is a lovely girl and she brings happiness to my family. The best outcome from the past years is I met my fiance Christof. Christof is a very kind, funny and smart person. I want to give him all my thanks for bringing me happiness over the last years, for all his help and for his correction of this thesis. Thanks again to my gracious families, and thanks to my friends from private life (though not listed here), to let me feel pleasure throughout all these years, and therefore I was able to concentrate on my research.

Abstract

An efficient way to represent the domain knowledge is relational data, where information is recorded in form of relationships between entities. Relational data is becoming ubiquitous over the years for knowledge representation due to the fact that many real-world data is inherently interlinked. Some well-known examples of relational data are: the World Wide Web (WWW), a system of interlinked hypertext documents; the Linked Open Data (LOD) cloud of the Semantic Web, a collection of published data and their interlinks; and finally the Internet of Things (IoT), a network of physical objects with internal states and communications ability. Relational data has been addressed by many different machine learning approaches, the most promising ones are in the area of relational learning, which is the focus of this thesis. While conventional machine learning algorithms consider entities as being independent instances randomly sampled from some statistical distribution and being represented as data points in a vector space, relational learning takes into account the overall network environment when predicting the label of an entity, an attribute value of an entity or the existence of a relationship between entities. An important feature is that relational learning can exploit contextual information that is more distant in the relational network. As the volume and structural complexity of the relational data increase constantly in the era of Big Data, scalability and the modeling power become crucial for relational learning algorithms.

Previous relational learning algorithms either provide an intuitive representation of the model, such as Inductive Logic Programming (ILP) and Markov Logic Networks (MLNs), or assume a set of latent variables to explain the observed data, such as the Infinite Hidden Relational Model (IHRM), the Infinite Relational Model (IRM) and factorization approaches. Models with intuitive representations often involve some form of structure learning which leads to scalability problems due to a typically large search space. Factorizations are among the best-performing approaches for large-scale relational learning since the algebraic computations can easily be parallelized and since they can exploit data sparsity. Previous factorization approaches exploit only patterns in the relational data itself and the focus of the thesis is to investigate how additional prior information (comprehensive information), either in form of unstructured data (e.g., texts) or structured patterns (e.g., in form of rules) can be considered in the factorization approaches. The goal is to enhance the predictive power of factorization approaches by involving prior knowledge for the learning, and on the other hand to reduce the model complexity for efficient learning.

This thesis contains two main contributions:

The first contribution presents a general and novel framework for predicting relationships in multirelational data using a set of matrices describing the various instantiated relations in the network. The instantiated relations, derived or learnt from prior knowledge, are integrated as entities' attributes or entity-pairs' attributes into different adjacency matrices for the learning. All the information available is then combined in an additive way. Efficient learning is achieved using an alternating least squares approach exploiting sparse matrix algebra and low-rank approximation. As an illustration, several algorithms are proposed to include information extraction, deductive reasoning and contextual information in matrix factorizations for the Semantic Web scenario and for recommendation systems. Experiments on various data sets are conducted for each proposed algorithm to show the improvement in predictive power by combining matrix factorizations with prior knowledge in a modular way.

In contrast to a matrix, a 3-way tensor is a more natural representation for the multirelational data where entities are connected by different types of relations. A 3-way tensor is a three dimensional array which represents the multirelational data by using the first two dimensions for entities and using the third dimension for different types of relations. In the thesis, an analysis on the computational complexity of tensor models shows that the decomposition rank is key for the success of an efficient tensor decomposition algorithm, and that the factorization rank can be reduced by including observable patterns. Based on these theoretical considerations, a second contribution of this thesis develops a novel tensor decomposition approach - an Additive Relational Effects (ARE) model - which combines the strengths of factorization approaches and prior knowledge in an additive way to discover different relational effects from the relational data. As a result, ARE consists of a decomposition part which derives the strong relational learning effects from a highly scalable tensor decomposition approach

RESCAL and a Tucker 1 tensor which integrates the prior knowledge as instantiated relations. An efficient least squares approach is proposed to compute the combined model ARE. The additive model contains weights that reflect the degree of reliability of the prior knowledge, as evaluated by the data. Experiments on several benchmark data sets show that the inclusion of prior knowledge can lead to better performing models at a low tensor rank, with significant benefits for run-time and storage requirements. In particular, the results show that ARE outperforms state-of-the-art relational learning algorithms including intuitive models such as MRC, which is an approach based on Markov Logic with structure learning, factorization approaches such as Tucker, CP, Bayesian Clustered Tensor Factorization (BCTF), the Latent Factor Model (LFM), RESCAL, and other latent models such as the IRM. A final experiment on a Cora data set for paper topic classification shows the improvement of ARE over RESCAL in both predictive power and runtime performance, since ARE requires a significantly lower rank.

Zusammenfassung

Relationale Daten sind ein leistungsfähiges Werkzeug um Domänenwissen zu repräsentieren. Informationen werden als Relationen zwischen Entitäten erfasst. Da man sich in verschiedenen Domänen zunehmend mit verketteten Daten beschäftigt, gewinnen Relationale Daten generell an Bedeutung. Einige bekannte Beispiel relationaler Daten sind: das World Wide Web (WWW), ein System verketteter Hypertext Dokumente; die Linked Open Data (LOD) Cloud des Semantischen Web, eine Kollektion veröffentlichter relationaler Daten; das Internet der Dinge, ein Netzwerk physischer Objekte, die interne Zustände messen und kommunizieren können. Relationale Daten sind durch viele verschiedene Verfahren des maschinellen Lernens adressiert worden. Die aussichtsreichsten sind im Bereich des Relationalen Lernens zu finden, dem Schwerpunkt dieser Doktorarbeit. Im Unterschied zu konventionellem Maschinellem Lernen, wo die Entitäten oft als unabhängige Instanzen betrachtet werden, die per Stichprobe gesammelt werden und als Datenpunkte im Vektorraum dargestellt werden, berücksichtigt Relationales Lernen die gesamte Netzwerk-Umgebung bei der Vorhersage der Markierung/Klasse einer Entität oder eines Attributwertes oder einer existierenden Verbindung. Eine wichtige Eigenschaft Relationalen Lernens ist, dass auch weiter entfernte kontextuelle Information im relationalen Netzwerk ausgenutzt werden kann. Da das Volumen und die strukturelle Komplexität der Daten im Zeitalter des Big Data ständig wachsen, werden Skalierbarkeit und Modellierungsvermögen zunehmend entscheidend für die Algorithmen Relationalen Lernens.

Bisherige relationale Lernalgorithmen basieren entweder auf einer intuitiven Repräsentation des Models, wie z.B. bei der Induktiven Logischen Programmierung (ILP) und Markov Logic Networks, oder sie gehen von einer Menge latenter Variablen aus, um beobachtete Daten zu erklären, Vertreter sind Infinite Hidden Relational Model (IHRM), Infinite Relational Model (IRM) und Faktorisierungs-Verfahren. Die Modelle mit intuitiver Repräsentation involvieren häufig eine Form des Struktur-

Lernens, was oftmals zu Skalierbarkeits-Problemen durch den typischerweise großen Suchraum führt. Faktorisierungen gehören zu den performantesten Verfahren für large-scale relationales Lernen, da sie mit niedriger Datendichte gut umgehen können und die algebraischen Berechnungen leicht parallelisiert werden können. Bisherige Faktorisierungs-Verfahren können nur Muster zentral in den relationalen Daten selbst betrachten. Weitergehend - und das ist der Kern dieser Doktorarbeit - kann zusätzliche A-priori-Information entweder durch unstrukturierte Daten (z.B. Text) oder in strukturierter Form (z.B. Regeln) durch Faktorisierungs-Verfahren ausgenutzt werden. Das Ziel ist, die Vorhersagegüte der Faktorisierungsverfahren durch die Einbeziehung von a-Priori Wissen zu verbessern, und gleichzeitig die Komplexität des Modells für ein effizientes Lernen zu reduzieren.

Diese Doktorarbeit enthält zwei wesentliche Beiträge:

Der erste Beitrag präsentiert ein allgemeines und neuartiges Rahmenwerk zur Vorhersage von Zusammenhängen in multi-relationalen Daten unter Verwendung einer Reihe von Matrizen, die die verschiedenen instanziierten Beziehungen im Netzwerk beschreiben. Die instanziierten Beziehungen, abgeleitet oder gelernt aus Vorwissen, werden als Attribute von Entitäten oder Attribute von Entitäten-Paaren in verschiedene Adjazenzmatrizen für das Lernen integriert. Die gesamte verfügbare Information wird dann in additiver Weise kombiniert. Effizientes Lernen wird durch eine alternierende Methode der kleinsten Fehlerquadrate erreicht über eine Ausnutzung von Sparse Matrix Algebra und Low Rank Approximation. Zur beispielhaften Illustration werden verschiedene Algorithmen vorgeschlagen, die Informationsextraktion, deduktives Reasoning und kontextuelle Information in Matrix-Faktorisierungen für Semantische Web Szenarien und Empfehlungssysteme mit einbeziehen. Zu allen vorgeschlagenen Algorithmen werden Experimente auf verschiedenen Datensätzen durchgeführt, um die Verbesserung der Vorhersagegüte durch das Kombinieren von Matrix-Faktorisierung mit Vorwissen aufzuzeigen.

Im Gegensatz zu einer Matrix bietet ein Tensor dritter Stufe eine viel natürlichere Repräsentation für multi-relationale Daten, wo Entitäten durch verschiedene Typen von Relationen verbunden sein können. Ein dreifach Tensor ist ein drei-dimensionales Array, das multi-relationale Daten in folgender Weise repräsentieren kann: Die ersten beiden Dimensionen stehen für Entitäten und die dritte Dimension für verschiedene Typen von Relationen. In dieser Arbeit wird durch eine eingehende Analyse der Berechnungskomplexität von Tensor-Modellen gezeigt, dass der Dekompositionsrang der Schlüssel zum Erfolg eines effizienten Tensor-Dekompositions Algorithmus darstellt und dass der Faktorisierungsrang durch Einbeziehung beobachtbarer Muster reduziert werden kann. Auf Basis dieser theoretischen Betrachtungen ist ein zweiter Beitrag dieser Arbeit die Entwicklung eines Tensor-Dekompositions Verfahrens, das Additive Relational Effects (ARE) Modell, das die Stärken von Faktorisierungsverfahren und beobachtbaren Mustern in additiver Weise verknüpft, zur Entedeckung verschiedener relationaler Effekte in den relationalen Daten. ARE besteht aus einem Dekompositions-Teil, der starke relationalen Lerneffekte durch das hoch-skalierbare Tensor-Dekompositions Verfahren RESCAL zeigt und einem Tucker 1 Tensor, der das Vorwissen durch instanziierte Relationen integriert. Ein effizienter Kleinste-Fehlerquadrate Ansatz wird zur Berechnung des kombinierten Modells ARE vorgestellt. Das additive Modell enthält Gewichte, die den Grad der Zuverlässigkeit des Vorwissens, evaluiert auf den Daten, reflektiert. Experimente mit verschiedenen Benchmark-Datensätzen zeigen, dass die Einbettung von Vorwissen zu leistungsfähigeren Modellen mit kleinerem Tensor-Rang führen mit signifikanten Nutzen für die Laufzeit und für Speicheranforderungen. Insbesondere zeigen die Ergebnisse, dass ARE state-of-the-art Verfahren des Relationalen Lernens einschließlich intuitiver Modelle wie MRC (ein Verfahren basierend auf Markov Logik Netzwerken mit Strukturlernen), CP, Bayesian Clustered Tensor Factorization (BCTF), das Latent Factor Model (LFM), RESCAL, und andere latente Modelle wie IRM übertrifft.

Ein abschließendes Experiment auf dem Cora Datensatz für Topic Klassifizierung zeigt eine Verbesserung von ARE gegenüber RESCAL sowohl in der Vorhersagegüte als auch in der Laufzeitperformanz, da ARE mit signifikant weniger Rängen auskommt.

Chapter 1

Introduction

This chapter outlines the general domains for relational learning and the specific tasks to be solved by relational models. An exhaustive survey on relational information will be given, which can be derived from the network structure and can be useful for the learning. The previous relational learning approaches that are most relevant to the work presented in the thesis will be discussed. The strengths and weaknesses of these approaches will be pointed out during discussions, as a foundation for introducing the contributions of this thesis.

1.1 Relational Data

Most of the real-world data is inherently relational, consisting of a set of entities related to each other in complex ways. Examples of relational data are

- Relational enterprise data, often explicitly defined by an Entity-Relationship model.
- Industry 4.0, where the structure of a plant and the automation process are described in terms of the involved components (entities) and their relationships in a specific context.
- The Linked Open Data (LOD) cloud of the Semantic Web, where knowledge from different domains is published in form of Resource Description Framework (RDF) triples.

- The World Wide Web (WWW), where hypertext documents are connected through hyperlinks.
- Google's knowledge graph, which enhances the search results by providing structured and detailed information about topics gathered from a wide variety of sources.
- Logfile data, such as event logs which contain huge amount of related events.
- Social networks, where people are linked by social relations such as friendship.
- Biology domains, where the gene-disease interactions or relationships among proteins, genes and chemical compounds are modeled.

For more examples of relational data, one can refer to [25].

Relational data represents information in form of the relationships between entities, which is an efficient way of knowledge representation in many domains such as the aforementioned examples. In the course of this paper, relational data from any domain can be represented as a labeled graph where the entities are nodes $\mathcal{V}$ and the relationships are labeled edges $\mathcal{E}$. A typical example is the Semantic Web area where information elements are represented as subject-predicate-object (s, p, o) triples. Entities (i.e. subjects and objects) are represented as nodes and statements are represented as directed labeled links from subject node to object node. This thesis follows the terms of subject-predicate-object triples in Semantic Web to define relational graphs.

Definition 1 (*Relational Graph*)**.** *A set of triples of the form* $\mathcal{G} = \{(\mathrm{s}, \mathrm{p}, \mathrm{o})\}$ *defines a relational graph over a domain* $\mathcal{D}$*, where* $\mathrm{s} \in \mathcal{V}$ *is a subject entity,* $p \in \mathcal{P}$ *denotes a relation type and* $\mathrm{o} \in \mathcal{V}$ *is an object entity or in case of a relation whose objects are the values, a literal. The vertices* $\mathcal{V}$ *are all the entities in* $\mathcal{D}$*.* $\mathcal{P}$ *is defined as the set of all the relation types in* $\mathcal{D}$*. Each edge* $\boldsymbol{e} \in \mathcal{E}$ *in the graph corresponds to a relationship and is defined by a triple* $(\mathrm{s}, \mathrm{p}, \mathrm{o})$*, i.e.* $\boldsymbol{e}$ *is a link starting from subject entity* s *to object entity* o*, labeled by* p*.*

In addition, $\mathcal{G}$ as defined in Definition 1 is called a multi-relational graph if it contains more than one distinct relation types, i.e. when $|\mathcal{P}| > 1$. Since from the definition of $\mathcal{G}$ the order of subject entity s and object entity o matters, $\mathcal{G}$ is a directed graph. An example of a directed graph is YAGO[1] knowledge base with RDF triples. The

[1]http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/

statement (s,p,o) is read as “s is p-related to o”. The existence of a relationship is modeled as a random variable $Z_{\text{s,p,o}}$, where

$$Z_{\text{s,p,o}} = \begin{cases} 1 & \text{if entity s is p-related to entity o} \\ 0 & \text{otherwise} \end{cases}$$

In most cases $Z_{\text{s,p,o}} \in \{0,1\}$ will be considered, but one can also use $Z_{\text{s,p,o}} \in [0,1]$ and it stands for the confidence value in the statement “entity s is p-related to entity o”.

Although the definition of the relational graph $\mathcal{G}$ covers only dyadic relations, note that any relation can be converted into dyadic form when more than two entities are involved, by introducing auxiliary entities. For example, a triadic relation under the schema of “table_match(teamA, teamB, score)” can be converted into three dyadic relations defined by “table_teamA(match, teamA)”, “table_teamB(match, teamB)” and “table_score(match, score)” after introducing new entities of type “match”. In the course of this thesis, discussions on relational data will refer to dyadic relational data without additional statement. Additionally, the term relationship will be specific to binary relationships between entities, while relationships between an entity and its attributes will be called attributes.

In the emerging era of Big Data, a rapidly increasing amount of relational data is derived from diverse sources such as text, audio, video, optical character recognition (OCR), sensor data, stream data, and etc.. Using the LOD cloud of the Semantic Web as an example, data sets in the LOD cloud consisted of over 2 billion RDF triples in October 2007. By the time of writing the thesis (summer of 2014), this number has grown to more than 73 billion RDF triples and the number keeps growing since there are increasing efforts from different communities sharing their data via the LOD cloud. The rapid increasing amount of relational data, which carries rich information and knowledge from different domains, can in turn benefit development of the applications in their domains. For example, the machine-readable RDF triples enable software driven processes based on expert knowledge, which relieves the burden of manually checking the knowledge and is crucial for domains with tremendous information such as biomedicine.

1.2 Machine Learning on Relational Data

Since relational data is being generated in ubiquitous domains, with increasing volume and structural complexity, the problem of learning in relational domains becomes more difficult and has received a considerable amount of attention from the machine learning community. This section will first list the main tasks that can be carried out with machine learning on relational data. What follows is a thorough analysis on the relational information that can be derived or exploited from the relational graph. Consequently, an introduction of two groups of relational models will be given, i.e. relational models learning from observed variables and relational models learning from latent variables.

1.2.1 Relational Learning Tasks

There are three popular tasks defined on relational data, i.e. link prediction to predict the existence of relationships between entities, attribute predictions to predict attribute values, and link-based clustering to group entities in the relational graph based on their relational similarity. This thesis will address the link prediction problem, although the proposed models can be used for attribute prediction and link-based clustering, as well.

1.2.1.1 Link Prediction

The objective of link prediction is to determine whether a relationship holds between two entities. Equivalently, the goal is to determine the state of the random variable $Z_{\mathrm{s,p,o}}$ as defined in Definition 1.1. One can predict, e.g., friendships among social network users, or the existence of triples in a knowledge base. As described before, the state of $Z_{\mathrm{s,p,o}}$ can be either binary or it can be a nonnegative scalar value representing a confidence value for the existence of the relationship. A link prediction problem can thus correspond to a recommendation system in terms of ranking the predicted relevance of the object candidates to a given subject entity. A typical example is to recommend items to a given user. In another family of applications, link prediction can be used for entity resolution, when it is applied to identifying the identical underlying entities, i.e. when the query relation type p equals to “isEqualTo” or “sameAs”.

A third special application of link prediction is collective classification [41, 89], where the task is to classify the class labels of entities in the relational graph, and p equals to “isClassOf”.

Link prediction has been applied in many existing projects and applications. It has been used to predict relationships between different topics in Google’s knowledge graph, in the recommendation systems of Amazon[2] and Netflix[3], for the task of entity resolution from natural texts, and to build large and clean knowledge bases [10, 15, 67].

1.2.1.2 Attribute Prediction

Attribute prediction is similar to link prediction, the objective of attribute prediction is to predict an attribute value of a given entity. For example, predicting the click-through rate (CTR) of an online advertisement is a form of attribute prediction used in ad-placement.

1.2.1.3 Link-Based Clustering

Different from the traditional clustering approaches which partition the entities based on the similarity of their observed attribute values, link-based clustering groups the entities based on their connectivity in the relational graph, i.e. based on the similarity of their relationships. The task of link-based clustering is also referred to as community detection in graphs [17], and has been included in applications such as detecting social groups in social networks, uncovering modules (which correspond to functional groups and associate to cancer and metastasis) in protein-protein interaction networks [46], grouping web pages with similar topics to enhance the PageRank [76] value of websites, and grouping system elements in food webs based on their predator-prey relationships to understand the system as a whole [17]. Link-based clustering can be used as another way for entity resolution. While link prediction gives a similarity measure for the likelihood that the two entities refer to the same entity, link-based clustering provides clusters learned from the entities’ connectivity and uses the clusters as a basics to group the similar or identical entities [71]. All in all, link-based clustering draws a readable map of large networks to help users

[2]http://www.amazon.com/gp/feature.html?docId=487250
[3]https://www.netflix.com/global

understand the structure of the networks.

1.2.2 Learning with Relational Information

Traditional statistical machine learning approaches assume that each sample is a random subset of a population of entities, and that associated random variables are independently and identically distributed (I.I.D). As a consequence, they solve the aforementioned tasks by learning a model from a fixed-length feature vector derived merely from the attribute values of the involved entities. However, with a relational setting, additional pieces of information are available, referred to as relational information, that helps to understand the complex structure of the underlying network. In [24] two systematic ways are reviewed to construct relational features from a relational graph, i.e. one is based on entity-specific measures to detect similar entities, and the other is based on entity-pair measures to detect similar links between pairs of entities. Both measures can be useful for the learning. Section 1.2.2.1 and Section 1.2.2.2 will extent these two categories of measures by giving concrete examples and further related measures.

1.2.2.1 Entity-Specific Information

Given a relational graph $\mathcal{G}$ as defined in Definition 1 and an entity represented as a node v in $\mathcal{G}$, there are two distinct information sources, i.e. attribute aggregations and structural properties, that can be simply extracted from the subgraph around v and can be utilized to determine its class label.

a) **Attribute Aggregations** Attribute aggregations summarize attributes of the neighborhood to describe local information about entity v. In [78] a hierarchy with four levels of aggregation concepts are defined over different complexity of the aggregations.
   - Level 1 corresponds to the observed attributes without aggregation.
   - Level 2 aggregates over the same kind of attributes of directly connected entities of v, such as the average age of friends, and the class labels of related neighbors. It has been shown that a simple predictive model based on only the class labels of neighbor entities surprisingly outperforms some complex models [63].

- Level 3 aggregates over different kinds of attributes of directly connected entities of v, such as the age of the youngest female friend that could be expressed using conditioning on the attribute of gender and an aggregation function min on the attribute of age.
- Level 4 aggregates attributes from entity v and the attributes of its directly connected entities, such as the common interest shared by a user and the user's friends.

Besides these approaches, more complex attribute aggregations and extensive studies on aggregations can be found in [63, 79].

b) **Structural Properties** Structural properties capture characteristics of the network structure. Heuristics from graph theory and social network analysis, such as centrality, cohesion and etc., can be used to exploit this information.

- Centrality of entity v measures its relative importance in the relational graph. There are four main types of centrality measures of an entity v.
  - Degree centrality, which is defined as the number of neighbors of v.
  - Closeness centrality, which is a distance metric defined as the inverse of the sum of the length from v to all other nodes [84].
  - Betweenness centrality, which calculates how many times entity v acts as a bridge along the shortest path between two other entities [19].
  - Eigenvector centrality, which assigns a score to each entity based on their influences in the network. Google's PageRank is a kind of the eigenvector centrality measure.
- Cohesion measures the node connectivity in the graph. Examples are local clustering coefficient which quantifies how close the neighbor of entity v can be a clique, and stability which measures the consistency of class labels in triads.

1.2.2.2 Entity-Pair Information

Given a relational graph $\mathcal{G}$ as defined in Definition 1, a relation type p and an entity-pair represented as (v_1, v_2), three distinct types of measures, i.e. attribute-based measures, link-based measures and neighborhood similarity measures, can be utilized to determine existence of the relationship (v_1, p, v_2). These three types of entity-pair measures summarize properties of the subgraph around the subject entity v_1 and the object entity v_2.

a) **Attribute-Based Measures** Attribute-based measures, such as boolean match, hamming distance, string similarity, cosine and etc., can be applied to the common attributes of entity-pairs to define the match of the two entities. For example, sharing the same email address indicates the identical user with a high probability.

b) **Link-Based Measures** Link-based measures assign scores to links (i.e. relationships) in the relational graph, based on the links of another relation type or the associated attribute values of the entities involved. A typical example is if a particular social relation holds for two persons, it might imply that there is a high probability that they share some other social activities. Another example is that male teenagers like action movies can be expressed using gender attribute of the users and genre information of the movies [44].

c) **Neighborhood Similarity Measures** Neighborhood similarity measures capture the overlapping neighborhoods. In [61] neighborhood methods are suggested, based on node neighborhoods and neighborhood similarity, can be utilized to define the neighborhood similarity of two entities.

- Node neighborhoods measures include:
  - Common Neighbors, which simply counts the number of neighbors the two entities have in common.
  - Jaccard's coefficient, which is a commonly used similarity metric in information retrieval [85] to compute ratio of the overlapping neighborhoods between two entities.
  - Adamic/Adar measure, which weights based on the inverse number of common neighbors [2] and suggests the similarity if two entities share more neighbors that are overall less frequent.
  - Preferential attachment score, which is based on the idea that entities with more neighbors have higher probabilities to be involved in a new relationship.
- Neighborhood similarity measures are path-based and include:
  - Katz score, which assumes two entities are similar if they are connected by short paths.
  - Hitting time, which measures the expected number of steps required

for a random walk between two entities.

– SimRank, which assumes two entities are similar if they are related to similar objects [39].

Beside these three types of entity-pair measures, the family of kernels can be used to determine existence of the relationship between two entities. Kernel functions defined over the attributes of the entities involved show the similarity of the entity-pairs. Graph kernels can be defined for various specific graph structures such as walks [21], cycles [36] and trees [90]. For example, a kernel representing the similarity of two entities can be defined by counting the common sub trees in the intersection graphs of the entities' neighborhoods [62].

Relational information, as just described, provides extra information from the network environment, which has been proved to be able to improve the prediction performance in some use cases [82, 108].

There are different ways of integrating relational information to a machine learning model.

Relational classifiers are one of the straightforward ways to make use of relational information in a model. Relational classifiers define relational information as fixed-length feature vectors and apply conventional classification or regression algorithms on them. These relational features are predefined and computed before the learning process. The advantage is that it enables a large number of well-established classification or regression algorithms for learning and prediction. But the I.I.D assumption still holds, and the relational features are only based on observed evidence (observed attribute values and observed relationships) which excludes the information derived from the attributes or relationships being predicted [24]. When the entities are partitioned into subgroups based on their connectivity in the network, the labels of entities can differ considerably from any classifier that makes an I.I.D assumption on the entities. Traditional classification or regression algorithms ignore the relationships among entities and are, therefore, unable to explore the extra piece of information from the network environment. It has been shown that modelling the dependencies between entities can greatly improve the predictive performance [25, 91]. A well known example is collaborative filtering, where users' preferences are predicted based on preferences of similar users. Furthermore, relational features are usually high-dimensional and the success of a relational classifier highly depends on the feature informativeness. As high-dimensional features usually suffer from the curse of dimensionality[4] for most of the relational classifiers, an additional feature selection procedure is thus required before the training process.

Beside relational classifiers, a second way of integrating relational information to a model is to perform a feature search during the learning process. This feature search is in the context of relational graph formed by the related entities, which is considered as an essential distinction to the previous non-relational approaches [99] such as the relational classifiers.

1.2.3 Relational Models

Relational data provides more information about an entity via its links. Relational models take into account this network environment and perform a feature search during learning, while non-relational models use more traditional features (such as the entities' attributes) and predefined relational features in a flat way. The goal of relational learning is to derive a model that can make predictions (typically with probabilities) for a large number of, or even all, possible relationships in a relational domain. One of the many possible groupings of relational models is based on their model representations. Relational models learning from observed variables make use of relational representations, such as logical expressions, schema information and ontological information, to guide the search on the relational feature space. Relational models that learn from latent variables associate a set of latent variables to the entities. The relational models are then expressed in form of the states of these latent variables. This subsection will review some state-of-the-art relational models from the groups of relational models learning from observed variables, and of relational models learning from latent variables. The representations, learning problems and inference methods of these models will be discussed.

1.2.3.1 Learning from Observed Variables

Early relational learning approaches, including Inductive Logic Programming (ILP) methods such as FOIL [80], assume a consistent hypothesis (with deterministic or

[4]With a fixed number of training samples, the predictive power of a classifier reduces as the dimensionality increases.

close-to-deterministic dependencies) that logically describes the data structure, such as

$$\text{Smokes}(var\_1) \leftarrow \text{Friends}(var\_1, var\_2) \wedge \text{Smokes}(var\_2)$$

which represents the hypothesis that if any friend is a smoker, then this person is also a smoker. Unfortunately ILP approaches show little robustness towards noise (not all the smoker's friends are also smokers) and towards missing information (whether the friends smoke is unknown) in the data.

Statistical Relational Learning (SRL) approaches overcome these problems by working with link probabilities. Most of the existing SRL approaches define their model based on graphical models [47] over the random variables, which can represent observable attribute values, observable relationships, or latent variables. Graphical models provide a principled way to dealing with uncertainty and complexities in the relational data through the use of both probability theory and graph theory. Generally, random variables are represented as nodes in the graph and the edges represent the statistical dependencies between random variables. The goal of SRL is to model the joint distribution $P(Z_1, \ldots, Z_n)$ over all random variables $\{Z_i | i = 1, \ldots, n\}$ in a relational domain, where n is the number of random variables and each Z_i corresponds to $Z_{\mathrm{s,p,o}}$ as defined in Section 1.1. When modelling joint distribution of a relational graph, the number of possible assignments in $Z_1, \ldots, Z_n$ is exponential to the number of random variables n. If random variables are defined on the level of observable attribute values or observable relationships, and the variables are binary-valued, it requires 2^n parameters to describe the joint distribution $P(Z_1, \ldots, Z_n)$. For example, starting with just 3 entities and 2 possible relation types among the entities, it ends up with a fully connected graph with the number of states $2^{3\times3\times2}$. This high-dimensional probability distributions becomes very problematic in computations for a relational graph as the number of entities and relation types grows. However, graphical models can better exploit model structure by defining random variables as shared parameters between the observable attribute values or relationships, as a way to exploit the independence properties that exists in many real world applications [53]. The two most common graphical representations of distributions are Bayesian Networks (BNs) for acyclic directed graphs, and Markov Networks (MNs) for undirected graphs, also known as Markov Random Field. Here briefly reviews three important SRL approaches, i.e. Probabilistic Relational Models, Bayesian Logic Programming Models, and Markov Logic Networks.

a) **Probabilistic Relational Models** Probabilistic Relational Models (PRMs) were one of the earliest successful approaches for statistical relational learning [20, 25, 54].

The representation of a PRM, like a Bayesian network, consists of the network dependency structure and the parameters quantifying the dependencies. PRMs extend Bayesian Networks with the concept of objects from the schema information in a relational database, and define templates for the dependency structure. Random variables in PRMs stand for ground atoms, either based on unary predicates (attributes), or based on dyadic relations (links). Direct edges in the relational templates indicate direct dependencies in the ground Bayesian Network. The parameters associating with the nodes are typically conditional probabilities tables, as in a Bayesian Network.

There are two distinguished types of learning problems for PRMs. When both the dependency structure and the observable attribute values are known, parameter learning is performed to estimate the likelihood of data given the dependency structure, i.e. to estimate the conditional probabilities. When the dependency structure is unknown, structure learning is required to find the set of edges. It is known that for Bayesian Networks in general, the task of finding the optimal structure is NP hard. PRMs use marginal probabilities or other score functions like BIC [32] to evaluate the candidate structures. A greedy search that iteratively adds, removes or reverses edges is used to find the dependency structure with the highest score. Overall, structure learning is a far more challenging task than parameter learning.

Exact inference in PRMs requires inference over the instantiated PRMs, which is usually computationally intractable due to the large number of random variables and their possible states in most real-world applications. Often approximate inference, like loopy belief propagation, is employed instead [66].

b) **Bayesian Logic Programming** Bayesian Logic Programming Models (BLPs), implemented in a software called BALIOS [49], are another class of successful SRL models.

BLPs unify Bayesian Networks with logic programming. The representation of

BLPs consists of a set of Bayesian clauses. A Bayesian clause differs from a logical clause (with a deterministic value) in that it uses a conditional probability table to present the probability of the head of the clause given its body. It is possible that the same random variable is included in the head of different clauses. Combination rules, such as the “noisy-or” rule, are used to combine probabilistic rules with the same head variables.

Learning in BLPs is a probabilistic extension of learning in ILP. Structure learning in BLPs has similar procedure as rule learning in ILP systems [73] which consider all possible operations such as adding or deleting new literals to the clause for each iteration. A score function based on maximum likelihood is defined to evaluate the clauses.

Inference in BLPs is done via standard Bayes Net inference on the grounded (instantiated) Bayesian clauses, which is intractable for large data sets as in PRMs.

c) **Markov Logic Networks** Markov Logic Networks (MLNs) [83] are the best known SRL models. An MLN is a probabilistic extension of logic which combines Markov Networks and first-order logic. First-order logic is also known as first-order predicate calculus, which is a formal logical system containing constants, variables, functions and predicates.

The representation of MLNs is a set of first-order logic formulae and their corresponding weights, i.e. (F_i, w_i). The set of formulae in MLNs describe the dependency structure on the class level. A grounding is defined as assigning a value from its domain to a variable in the formula. The set of formulae and a finite set of entities in a domain can be viewed as a template to construct such a grounded Markov network: Each node is a binary random variable corresponding to a ground predicate. The state of a node equals to 1 if the ground predicate (also known as ground atom in MLNs) is true, and 0 otherwise. Furthermore, a feature function f_{i_k} is defined for each possible grounding of a formula F_i, which has the value of 1 if the grounding formula is true, and 0 otherwise. An edge is created between two nodes if the ground predicates occur in at least one formula. As a result, each grounding formula forms a clique in the grounded Markov network. As each node is for a decision, connecting dependent decisions in cliques gives collective attribute prediction. A possible world represents a state of all the random variables, i.e. the truth values of all the ground predicates. The probability distribution over possible worlds z can be written as:

$$P(Z = z) = \frac{1}{\Psi} \exp \left(\sum_i w_i n_i(x) \right)$$

where $n_i(x)$ represents the number of true groundings for formula F_i in z, the same weight w_i is assigned to all the possible groundings of the same formula F_i, and Ψ is the partition function that guarantees the probabilities of all possible groundings sum up to one.

When the set of first-order logic formulae are known, parameter learning in MLNs corresponds to estimating the weights w_i from one or more relational databases. MLNs make a close-world assumption [23] that any absent ground predicate in the database is assumed to be false. The weights w_i are computed by maximizing the log likelihood of the data. Pseudo-likelihood is used for calculating the probability of a possible world, which approximates the distribution of z based on the product of the probabilities of each node given its Markov blanket[5]. Structure learning is employed when the set of first-order logic formulae are unknown, which is similar to BLPs. MLNs use the CLAUDIEN [81] system to learn the first-order clauses and not just Horn clauses.

Inference in a MLN corresponds to predicting the truth value of a ground predicate given the data. An MLN usually results in a large Markov network, which immediately poses a challenge on computing the inference. Traditional approaches on MLNs inference consider a subset of the ground Markov network and compute the conditional probability on it using Gibbs sampling. A more recent and efficient approach on MLN inference Tuffy [74] partitions an MLN program into high-level tasks (such as classification) that can be solved with specialized algorithms [75].

In addition to PRMs, BLPs and MLNs, various SRLs have been proposed over the decades, such as Relational Dependency Network [68] which is an extension of Dependency Networks [30], Relational Markov Network [97] which is a conditional Markov network, and DAPER [33, 31] which is an extension of entity-relationship model. For

[5]In a Markov network, the Markov blanket of a node is the set of its neighboring nodes

more SRL models, one can refer to the work [25]. SRLs have been generally successful on relational learning. However, the computational complexity of inference is the essential limitation of most SRLs. Although approximate methods make inference tractable, it is still infeasible for large-scale data. Another limitation of most SRLs is that they often require structure learning when lacking the full knowledge about the dependency of the data. Structure learning in SRLs is usually computational costly and state-of-the-art approaches have been only tested on small to medium sized data sets [12, 51].

1.2.3.2 Learning from Latent Variables

All those afore described relational learning approaches model the statistical dependencies in the data with random variables that can be observed from the data. Another group of relational learning approaches assume a set of latent variables as random variables that are associated to the entities. The advantage is that no structure learning is needed. The independence properties of the network are exploited in that the observed variables (such as relationships) are conditionally independent given the latent variables. Once the states of the latent variables are inferred from the data, predictions can be made via the latent representations of the corresponding entities. Examples are Infinite Hidden Relational Model (IHRM) [107], Infinite Relational Model (IRM) [48] and the factorization approaches.

a) **Infinite Hidden Relational Model and Infinite Relational Model** IHRM assigns each entity to one latent class. The state of the random variable $Z_{\mathrm{s,p,o}}$, as defined in Section 1.1 which corresponds to the probability of a relationship $(\mathrm{s}, \mathrm{p}, \mathrm{o})$ between a subject entity s and an object entity o being true, depends on the relation type p and the latent classes L(s), L(o) of the two entities:

$$P\big(Z_{\mathrm{s,p,o}} = 1 | \mathrm{L(s)}, \mathrm{L(o)}\big) = \theta_{\mathrm{p,L(s),L(o)}}$$

where $0 \leqslant \theta_{\mathrm{p,L(s),L(o)}} \leqslant 1$. Learning in IHRM is to infer the assignments of the latent classes and the class-relationship probabilities from the data. The class assignments are drawn from a Chinese restaurant process (CRP) and the number of latent classes is allowed to be infinite. Inference in IHRM uses Gibbs sampling on only a finite number of latent classes. The underlying idea of IRM is similar to IHRM, while the mixed membership stochastic blockmodel [3] generalizes it to allow more than one latent classes associated to an entity. An important advantage of these latent-classes based models is that the latent classes summarize the information of the related entities. Since the latent classes encode information of the represented entities participating in different relationships, information is propagated in the network via the latent classes. However, these methods suffer the same scalability problem as most SRLs because of the sophisticated inference procedure, which hinders their application on large-scale data.

b) **Factorization Approaches** Another family of approaches that learn from latent variables are based on the factorization of matrices and 3-way tensors, which partition the relational graph into subgroups based on their connectivity and neighborhood similarities.

Matrix factorization approaches represent the relational graph $\mathcal{G}$ as defined in Definition 1 by an adjacency matrix.

Definition 2 (*Adjacency Matrix*)**.** *An adjacency matrix* $X \in \mathbb{R}^{|\mathcal{S}|\times(|\mathcal{P}|*|\mathcal{O}|)}$ *for the relational graph* $\mathcal{G}$ *is a two dimensional array, where each row represents a subject entity* $s \in \mathcal{S}$*, each column represents a pair of relation type* $p \in \mathcal{P}$ *and object entity* $o \in \mathcal{O}$*, i.e.* (p, o)*. The confidence value of the relationship* (s, p, o) *being true is therefore represented as an entry in* X*, denoted as* $x_{s,p,o}$.

During the learning process of matrix factorizations, the adjacency matrix X is decomposed into a multiplication of some smaller-sized component matrices. Accordingly, the subject entities $\mathcal{S}$ and the pairs of relation type - object entity $(\mathcal{P}, \mathcal{O})$ are mapped to a joint latent factor space with a much lower dimensionality, such that the interactions of $\mathcal{S}$ and $(\mathcal{P}, \mathcal{O})$ are modelled as inner products in that latent space, and predictions can be made by simple calculations on the latent factors. Examples of matrix factorization techniques are collaborative filtering, Singular Value Decomposition (SVD) which is a well-established approach for information retrieval in identifying latent semantic factors, and SUNS [37, 98] which is a regularized SVD and will be described in detail in section 2.1.2. Take SUNS for example, the learning process with a single relation type where $|\mathcal{P}| = 1$, can be viewed as to decompose X into factors of a latent representation $U \in \mathbb{R}^{|\mathcal{S}|\times r}$ which summarizes the subject entities, a latent representation $V \in \mathbb{R}^{|\mathcal{O}|\times r}$ which summarizes the object entities, and a diagonal interaction matrix $D \in \mathbb{R}^{r \times r}$ where each entry d_{ij} describes the degree of interaction between the i-th latent component of U and the j-th latent component of V. r is the predefined rank, as well the number of latent components.

A 3-way adjacency tensor is a more natural representation for the multi-relational graph $\mathcal{G}$ as defined in Definition 1.

Definition 3 (*Adjacency Tensor*). *A 3-way tensor* $\mathbf{X} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{P}| \times |\mathcal{O}|}$ *for the relational graph* $\mathcal{G}$ *is a three dimensional array, where the subject entities* $\mathcal{S}$*, the object entities* $\mathcal{O}$*, and the relation types* $\mathcal{P}$ *are respectively represented by the first, the second and the third mode of* $\mathbf{X}$*. The confidence value of the relationship* $(\mathrm{s}, \mathrm{p}, \mathrm{o})$ *being true is therefore represented as an entry in* X*, denoted as* x_{spo}*, where* $s \in \mathcal{S}$*,* $p \in \mathcal{P}$ *and* $o \in \mathcal{O}$*.*

Similar to matrix factorizations, tensor decomposition approaches use component matrices to represent the latent space of the corresponding dimension, and a core tensor to represent the interactions among these component matrices. Predictions are made via generalized matrix operations to reconstruct the tensor from the latent factors. A tensor provides greater descriptive flexibility for multi-relational data than a matrix [1]. Each frontal slice in the tensor models the interaction of the subject entities and the object entities in a specific relation. Modelling the interactions of entities under different relation types with a tensor comes with a computational cost compared to a matrix, as the unfoldings of a tensor from different modes correspond to three matrices. This extra modelling complexity allows to discover different combinatorial possibilities of latent vectors respectively from the subject entities, the object entities and the relation types, as the interactions among the three are frequently lost when a tensor is reshaped into only one matrix. In other words, instead of using linear models for matrix factorizations, tensor decompositions enhance the modelling power using multi-linear models and simultaneously retain the scalability compared to non-linear models [69] such as Support Vector Machines with RBF kernels. Tensor decompositions have been used early in the field of chemometrics and psychometrics [104], and in recent years they received more attention also from machine learning and data mining community. A number of tensor decomposition approaches have been proposed over the last decades. Tucker [101] is one of the early tensor models and it decomposes $\mathbf{X}$ into factors of a latent representation $A \in \mathbb{R}^{|\mathcal{S}| \times r}$ of the subject entities, a latent representation $\boldsymbol{B} \in \mathbb{R}^{|\mathcal{O}| \times r}$ of the object entities, a latent representation $\boldsymbol{C} \in \mathbb{R}^{|\mathcal{O}| \times r}$ of the relation types, and an interaction tensor $\mathbf{G} \in \mathbb{R}^{r \times r \times r}$ where each entry g_{ijk} represents the degree of interaction between the corresponding latent components. PARAFAC (also called CP) [34, 35], can be considered as a special case of Tucker, where the interaction tensor $\mathbf{G}$ is diagonal and facilitates analysis on the latent components. Bayesian Clustered Tensor Factorization (BCTF) [94] embeds a factorized representation of relations in a non-parametric Bayesian clustering framework. RESCAL [71, 72], which represents the subject entities $\mathcal{S}$ and object entities $\mathcal{O}$ with the same latent representation, provides high scalability and shows strong relational learning ability. Particularly, RESCAL will be introduced in detail later in section 3.1.2.3. The Latent Factor Model (LFM) is a work based on RESCAL but it models additionally the correlations among the relation types given a subject - object entity pair. These are the most important tensor models and there are more, which can refer to a survey on tensors [52].

For all these latent variable models, factorizations are among the most-promising approaches as compared to IHRM and IRM for large-scale relational learning, because of the highly paralleled computations via linear algebra (for matrices, multi-linear algebra for tensors) and being capable of exploiting the data sparsity.

1.3 Conclusions and Motivations

So far it has been shown that given a relational graph representing relational data in a domain, there exists huge amount of possible relational information derived from the graph structure. As a consequence, a challenge immediately posed to all the relational models is on the search of a large feature space. Relational models learning from observed variables although provide an intuitive representation of the model, they often require structure learning which is usually intractable for large-scale data due to the large number of random variables and their possible states. Relational models learning from latent variables overcome this problem by associating a set of latent variables to the related entities and the learning only consists of inferring the states of these latent variables from the data.

A second observation is that different relational models learn the rule-like relational information in different ways. ILP-based relational models such as PRMs, BLPs and MLNs, can explain rules with long distance dependencies in the network structure. Factorization of a matrix or a tensor normally considers one-step dependencies during learning, i.e. the direct neighbor in the relational graph. If parts of the rule are not in a big latent component, a high factorization rank is required or collective learning is needed. Collective learning is an evolutionary process of sharing, disseminating and further developing individual knowledge. RESCAL is a well-established tensor model and has been proved to have collective learning effect.

A third comparison is on how to include relational information or prior knowledge to different relational models. Relational models such as PRMs, BLPs and MLNs, that represent the network structure with logical expressions, schema information and ontological information, can easily include relational information into the model template. For example, the attribute aggregations can be defined as an additional node in PRMs, BLPs, and as an additional literal in MLNs. For factorization approaches, aggregated attributes need to be instantiated and be added to a matrix as columns or to a tensor as slices.

A fourth observation is regarding to information propagation. Existing relational models either include an sophisticated inference procedure (e.g. Gibbs sampling) for propagating information through the dependency structure, such as shown in PRMs, BLPs, MLNS, IHRM and etc., or perform a global optimization to infer the model parameters such as factorization approaches. Furthermore, for relational models that include an inference procedure, the information derived from predicted attribute values or relationships is passed along the dependency structure, whereas for latent variable models, this piece of information is encrypted in the latent variables.

All in all, due to the high demand on performance and scalability of a relational model for large-scale data, this thesis will focus on factorization approaches, including factorization of matrices and 3-way tensors. In many cases partial knowledge about the data is known, which can be expressed as relational information (such as aggregated attributes and some path-based measures) or rules in form of Horn clauses. These elements of prior knowledge may be only available for parts of the network, and applying the corresponding expression of relational information or rules to the whole network can bring redundance. Therefore, simple but efficient extensions of factorization approaches are proposed in this thesis for integrating prior knowledge to factorization approaches for improving the predictions, while at the same time the model retains good scalability and has tolerance with redundant or wrong prior information.

1.4 Chapter Map

Chapter 2 and Chapter 3 respectively present the two main contributions of this thesis, with the goal of deploying methodologies to integrate prior knowledge into factorization approaches for relational learning.

The first contribution described in Chapter 2 presents a general and novel framework for predicting relationships in multi-relational data. The chapter starts with an introduction on a regularized SVD, as the basics for the proposed additive framework. An extended additive model using a set of matrices describing the various instantiated relations in the network is discussed, as a theoretical support for the efficient learning. The proposed additive model combines all the information derived or learnt from prior knowledge as entities' attributes or entity-pairs' attributes in form of different adjacency matrices for the learning. The scenario of relation prediction in the Semantic Web and the use case of recommendation systems are used to evaluate the proposed framework. In the Semantic Web scenario, the three most common approaches in that area (information extraction, deductive reasoning and machine learning) are combined under the proposed framework for relation prediction. In the use case of recommendation systems, both the collaborative filtering effects and the contextual information (as prior knowledge) are utilized in the proposed framework for the learning. Experiments on various data sets are conducted for each use case to show the improvement in predictive power by combining matrix factorizations with prior knowledge in a principled way.

A novel tensor decomposition model is presented in Chapter 3 as the second contribution of this thesis. A 3-way tensor is a more natural representation for the multi-relational data compared to a matrix. The chapter starts with an introduction on tensors and tensor models. Before proposing the new tensor model, an analysis on the computational complexity of tensor models shows that the decomposition rank is key for the success of an efficient tensor decomposition algorithm. An analysis on synthetic data illustrates that the factorization rank can be reduced by including observable patterns. Based on these theoretical considerations, a novel tensor decomposition approach ARE, an Additive Relational Effects (ARE) model is proposed. ARE combines the strengths of factorization approaches and prior knowledge in an additive way to discover different relational effects from the relational data. As a result, ARE consists of a decomposition part which derives the strong relational learning effects from a highly scalable tensor decomposition approach RESCAL, and a Tucker 1 tensor which integrates the prior knowledge (path-based patterns or neighborhood-based patterns) as instantiated relations. An efficient least squares approach is proposed to compute the combined model ARE. The additive model contains weights that reflect the degree of reliability of the prior knowledge, as evaluated by the data. Experiments on several benchmark data sets show that the inclusion of prior knowledge can lead to better performing models at a low tensor rank, with significant benefits for run-time and storage requirements. In particular, the results show that ARE outperforms many state-of-the-art relational learning algorithms. A final experiment on paper topic classification shows the improvement of ARE over RESCAL in both predictive power and runtime performance, since ARE requires a significantly lower rank.

The last chapter concludes this thesis, discusses the possible extensions and the application domains.

The two aforementioned contributions of this thesis are related to the following publications:

[43] Xueyan Jiang, Volker Tresp and Denis Krompass. “A Logistic Additive Model for Relation Prediction in Multi-relational Data”. In: *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)* 2013 *Workshop on Tensor Methods for Machine Learning*, 2013.

[44] Xueyan Jiang, Volker Tresp, Yi Huang, and Maximilian Nickel. “Link Prediction in Multi-relational Graphs using Additive Models”. In: *Proceedings of International Workshop on Semantic Technologies meet Recommender Systems* & *Big Data at the* 11*th International Semantic Web Conference (ISWC)*, 2012.

[42] Xueyan Jiang, Yi Huang, Maximilian Nickel, and Volker Tresp. “Combining Information Extraction, Deductive Reasoning and Machine Learning for Relation Prediction”. In: *Proceedings of the* 9*th Extended Semantic Web Conference (ESWC)*, 2012.

[45] Xueyan Jiang, Volker Tresp, Yi Huang, Maximilian Nickel, and Hans-Peter

Kriegel. “Scalable Relation Prediction Exploiting Both Intrarelational Correlation and Contextual Information”. In: *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD)*, 2012.

[70] Maximilian Nickel, Xueyan Jiang, and Volker Tresp. “Reducing the Rank of Relational Factorization Models by Including Observable Patterns”. In: *Advances in Neural Information Processing Systems (NIPS)*, 2014.

The extended additive model in Chapter 2 was proposed in Section 2.1 of [43]. In this thesis, Chapter 2 generalizes the extended additive model from Section 3 to Section 5 of [44] for combining all the entity-specific and entity-pair information. As illustrations, use cases in the Semantic Web and for recommendation systems are shown. A set of experiments for these two use cases are respectively from the experimental parts of [42] and [45]. The ARE model in Chapter 3 is an additive model similar to the model from [44]. The model formulation of ARE and the related experiments are from Section 3 and Section 4 of [70].

Chapter 2

Additive Models for Relation Prediction

This chapter extends the traditional additive models by using a set of relational adjacency matrices as components describing the various instantiated relations in the relational network. Prior knowledge is represented in form of matrices as entities' attributes or entity-pair's attributes. The low-rank approximation of the query relation based on these matrices are then combined in an additive way for the learning. Two scenarios, in the Semantic Web area and for recommendation systems, are presented to show the improvement in predictive performance by adding prior knowledge to matrix factorizations.

The chapter begins with an introduction to matrix factorizations. Particularly, the SUNS model will be discussed in section 2.1. Section 2.2 will present the additive models whose solution is based on regularized SVD. Section 2.3 will describe in general how to apply the additive models for link prediction in multi-relational graphs. As illustrations, section 2.4 will demonstrate by using the additive models, how to include information extraction, deductive reasoning and machine learning for relation prediction in the Semantic Web area; Sections 2.5 will demonstrate how to exploit both intrarelational correlation and contextual information for recommendation systems.

2.1 A Review on SUNS

In 2006, the online DVD rental company Netflix organized a contest to improve its recommendation system [5]. A training set of more than 100 million ratings was released, with respect to about 500,000 anonymous customers and their ratings on more than 17,000 movies, where each movie was rated on a scale of 1 to 5 stars. Participating teams submit predicted ratings for a test set consisting of approximately 3 million ratings, and the evaluation is done by calculating a root-mean-square-error (RMSE) on the held-out truth. The majority of winning entries of this competition are models based on matrix factorizations [4, 9, 96]. Since then, matrix factorizations have become more and more popular because of their good scalability and high predictive accuracy [56]. The basic form of the matrix factorization models infers vectors of factors from movie rating patterns. Accordingly, both users and movies are mapped to a joint latent factor space of a much lower dimensionality, to discover the most descriptive dimensions for predicting movie preferences. Such a modelling can correspond to singular value decomposition (SVD), a well-established technique for identifying latent semantic factors in information retrieval. Applying SVD to this recommendation system requires factorizing the user-movie rating matrix, which usually has very high sparsity since users have rated only a small percentage of all the movies. The unknown ratings are treated as missing values in the user-movie matrix. This incomplete knowledge raises difficulty to conventional SVD. Many existing works solve it either by filling in missing ratings to make the rating matrix dense [88], or by modelling only the observed ratings through a regularized model [55, 77, 95]. Particularly in relational domains, a relational adjacency matrix $X \in \mathbb{R}^{|\mathcal{S}| \times (|\mathcal{P}| * |\mathcal{O}|)}$ as defined in Definition 2 is used to represent a relational graph with $|\mathcal{S}|$ subject entities, $|\mathcal{O}|$ object entities and $|\mathcal{P}|$ relation types. Instead of ratings and missing values in the user-movie matrix, each entry in the adjacency matrix X can be either 1 or 0 to represent existence of a relationship. Thus the missing values for the user movie case are treated as untrue for relation prediction in the relational domain. For example, each entry in a relational database represents a true relationship, while the absent relationships are most likely to be untrue. This close-world assumption also holds for the Semantic Web area, where the known triples represent true relationships and the absent triples represent untrue relationships.

One of the first line of research where matrix representations were used for relation prediction is the SUNS framework [37, 98], which is a regularized SVD-based low-rank approximation technique. SUNS is selected as the basic matrix factorization model for the proposed approaches in this chapter. SVD has been implemented in many high performance software packages such as LAPACK[1], Apache Mahout[2], scikit-learn[3], MLlib[4], GraphLab[5], Teradata Aster[6] and these implementations can easily be extended to SUNS.

2.1.1 Data Representation

As defined in Definition 1, labeled links in a relational graph $\mathcal{G}$ are represented as triples of the form (s, p, o) where subject s and object o stand for entities in a domain and where p is the relation type, i.e. the link label. The statement (s, p, o) stands for the relationship that "s is p-related to o". Definition 2 defines an adjacency matrix X where each entry $x_{\mathrm{s,p,o}}$ represents the truth value of relationship (s, p, o). Moreover, a random variable $Z_{\mathrm{s,p,o}}$ is defined in Section 1.1 and is associated with the triple (s= i, p= j, o= k). Consider that each entry in X corresponds to a random variable $Z_{\mathrm{s,p,o}}$, $x_{i,j,k} = 1$ is set when the relationship (s, p, o) is known to exist, otherwise $x_{i,j,k} = 0$. , i.e.

$$x_{i,j,k} = \begin{cases} 1 & \text{if an edge labeled as } k \text{ exists between node } i \text{ and node } j \\ 0 & \text{otherwise} \end{cases},$$

where in multi-relational graphs, the entities including subject entities and object entities form the nodes.

Usually, a large relational graph $\mathcal{G}$ comprises a sparse adjacency matrix $X \in \mathbb{R}^{I \times JK}$ ($I = |\mathcal{S}|$, $J = |\mathcal{O}|$, $K = |\mathcal{P}|$), since only a small percentage of all possible relationships exist. By using this matrix representation X, the relational graph $\mathcal{G}$ is accordingly represented by a bipartite graph Γ with I subject nodes and JK pairs of relation type - object node, i.e. (p, o) pairs. By the definition of a bipartite graph, the vertex set $\mathcal{V}$ with $(I + JK)$ vertex can be divided into mutually exclusive vertex set $\mathcal{V}_1$ (with I subject nodes) and vertex set $\mathcal{V}_2$ (with JK pairs of (p, o)), where $\mathcal{V}_1 \cap \mathcal{V}_2 = \{\}$

[1]http://www.netlib.org/lapack/
[2]https://mahout.apache.org/
[3]http://scikit-learn.org/stable/
[4]https://spark.apache.org/mllib/
[5]http://graphlab.org/projects/index.html
[6]http://www.teradata.de/Teradata-Aster/overview

and $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$. In the adjacency matrix X, the row vector $\boldsymbol{x}_{i,:,:} \in \mathbb{R}^{JK}$ shows the neighbors of subject node i, $i = 1, \ldots, I$, where each neighbor is related to an object node and a relation type. Similarly, the column vector $\boldsymbol{x}_{:,j,k} \in \mathbb{R}^{I}$ shows the neighbors of object node j with a specific relation type k, where $j = 1, \ldots, J$, $k = 1, \ldots, K$.

2.1.2 A Regularized SVD-based Low-rank Approximation

2.1.2.1 Standard Singular Value Decomposition

Standard Singular Value Decomposition (SVD) is based on a matrix factorization method which decomposes the adjacency matrix $X \in \mathbb{R}^{I \times JK}$ as following [27]:

$$X = UDV^T \ . \tag{2.1}$$

SVD causes a linear transformation that diagonalizes matrix X into the product of an orthogonal matrix U, a diagonal matrix D and an orthogonal matrix V. When discarding the null space, $U \in \mathbb{R}^{I \times r}$ and $V \in \mathbb{R}^{JK \times r}$ are orthogonal bases respectively spanning the column space $\mathbb{R}^{JK}$ and spanning the row space $\mathbb{R}^{I}$ of X, where r is the rank of X. The diagonal entries of $D \in \mathbb{R}^{r \times r}$ are nonnegative real numbers organized in a decreasing order of magnitude. Furthermore, the columns of U are the eigenvectors of the covariance matrix XX^T, and the columns of V are the eigenvectors of X^TX. The diagonal entries in D contains singular values with scaling information about how a vector is stretched or shrunk when it goes from the column space $\mathbb{R}^{JK}$ to the row space $\mathbb{R}^{I}$.

Following [92], one can obtain from Equation 2.1

$$X\boldsymbol{v}_{:p} = d_p \boldsymbol{u}_{:p} \tag{2.2}$$

where $p = 1, \ldots, r$, $\boldsymbol{v}_{:p}$ and $\boldsymbol{u}_{:p}$ are respectively column vectors of V and U, and d_p is a diagonal entry of D. Equivalently

$$XV = UD.$$

The left-hand side of Equation 2.2 tells that the dot product between a row vector of X (i.e. $\boldsymbol{x}_{i,:,:}$) and the p-th latent component of V (i.e. $\boldsymbol{v}_{:p}$) is to project $\boldsymbol{x}_{i,:,:}$ to

$\boldsymbol{v}_{p:}$, as a measure of how much the subject node i points in the same direction as $\boldsymbol{v}_{:p}$. Obviously, if two subject nodes share many common neighbors (pairs of relation type - object node) or highly correlate with each other, they will have similar measure in each eigenvector $\boldsymbol{v}_{:p}$. The right-hand side of Equation 2.2 tells that this projection from row vector $\boldsymbol{x}_{i,:,:}$ to eigenvector $\boldsymbol{v}_{:p}$ can be expressed in terms of the corresponding column basis $\boldsymbol{u}_{:p}$ and the singular value d_p. As V is orthogonal, all vectors of subject nodes that share many common neighbors will point in the same direction in the row space spanned by V. Similar analysis can be done with

$$X^T U = VD \; .$$

The observation is then the pairs of relation type - object node sharing many common neighbors (subject nodes) will point in the same direction in the column space spanned by U and this direction can be expressed by a linear combination of the corresponding row basis in V and the singular values in D.

Thus, the product of UD provides an abstract way of describing the subject nodes:

$$d_1 u_{i1} + d_2 u_{i2} + \cdots + d_r u_{ir}, \quad i = 1, \ldots, I \; . \tag{2.3}$$

Correspondingly, VD describes the pairs of relation type - object node:

$$d_1 v_{l1} + d_2 v_{l2} + \cdots + d_r v_{lr}, \quad l = 1, \ldots, JK \; .$$

Considering the columns of U as r clusters for the column space $\mathbb{R}^{JK}$ of X, each term in Equation 2.3 tells how much node i belongs to a cluster, i.e. the membership of a subject node i to the cluster $\boldsymbol{u}_{:p}$ is d_p. From Equation 2.2 and 2.3, the subject nodes sharing many common neighbors point in the same direction in the space formed by V, and have similar memberships to the clusters formed by U. Similarly, for a given relation type the object nodes that share many common neighbors will have close memberships to the clusters formed by V. As a result, the highly connected groups of nodes in the relational graph $\mathcal{G}$ will point in the same direction in space and can be viewed as belonging to the same cluster, due to sharing similar memberships to all the eigenvectors [87]. Figure 2.1 shows two bipartite graphs (respectively generated from two different relational graphs with a single relation type) and their SVDs. The observation is that the number of latent components, i.e. the rank of X, corresponds to the number of clusters for the links in the associated bipartite graph.

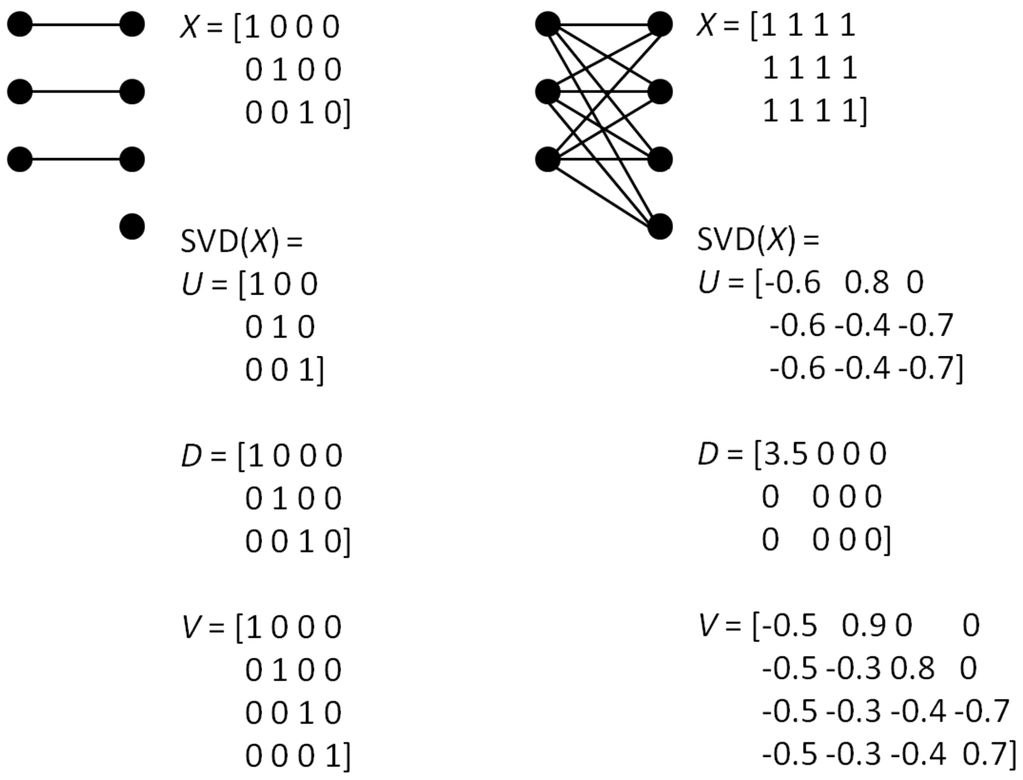


Figure 2.1: This picture is adopted from [87], which shows two extreme cases for SVD. The left part shows a bipartite graph generated from a relational graph where no common neighbors exist among the subject nodes or among the object nodes. All columns (or rows) in the corresponding adjacency matrix X are independent of each other. The rank of X is 3 which indicates 3 clusters in the graph, with each of them formed by one subject-object node pair. The right part shows a bipartite graph generated from a highly connected relational graph where all the nodes are connected to each other. All the columns (or rows) in X linearly depend on each other. All the nodes belong to exactly one cluster, as the same number for the rank of X.

2.1.2.2 Dimension Reduction

If the nodes in a relational graph share the same or most of the neighbors, there will be dependent rows or columns in X. In case of such redundancy, a low-rank approximation to the adjacency matrix X is suggested and preferred to represent the nodes in a reduced dimensional space [92].

$$X \approx U_k D_k V_k^T$$

where only the largest k singular values and the corresponding eigenvectors are retained to reconstruct X, usually $k << r$. By discarding the least important $r - k$ singular values and vectors as noise, the nodes that share many common neighbors will orient further in the same direction in the corresponding eigenvector space, and therefore, be more similar to each other.

2.1.2.3 A Regularized SVD for Relation Prediction

Relation prediction using SVD corresponds to complete the adjacency matrix X. After the lower dimensionality approximation compresses the space formed by X to a joint latent factor space formed by the major eigenvectors, the subject nodes and the relation type - object node pairs are respectively clustered by their similarity to the corresponding latent factors. These latent factors play the role of latent representations for the subject nodes and the (p,o) pairs. Accordingly, the i-th subject node has a latent representation of $\boldsymbol{u}_{i:}$, the j-th (p,o) pair has a latent representation of $\boldsymbol{v}_{j:}$, and the interaction between these two latent vectors is the vector $\boldsymbol{d}$ derived from the diagonal entries of $\boldsymbol{D}$. From the discussion in Section 1.2.3.2 on relational models that learn from latent variables, any query on a possible relationship with respect to a subject node and a pair of relation type - object node can be inferred from their latent representations. The prediction of a relationship denotes the confidence value of the truth of the relationship. Predictions of all the possible relationships among subject nodes and object nodes are typically represented by a densely filled matrix. Particularly, the SUNS model [37, 98] is chosen for relation prediction. SUNS is essentially a regularized SVD that applies regularization in the prediction phase to avoid overfitting. With a regularizer $\lambda > 0$, the adjacency matrix X can be reconstructed in three different ways.

The first way is to recover X from the joint latent factor space formed by U and V.

$$\tilde{X} = U_k \operatorname{diag}\left\{\frac{d_p^3}{d_p^2 + \lambda}\right\}_{p=1}^{k} V_k^T \tag{2.4}$$

The second way is to recover X from the column space spanned by U.

$$\tilde{X} = U_k \operatorname{diag}\left\{\frac{d_p^2}{d_p^2 + \lambda}\right\}_{p=1}^{k} U_k^T X \tag{2.5}$$

The third way is to recover X from the row space spanned by V.

$$\tilde{X} = X V_k \operatorname{diag}\left\{\frac{d_p^2}{d_p^2 + \lambda}\right\}_{p=1}^{k} V_k^T \tag{2.6}$$

2.1.2.4 Scalability

The expensive part of SVD is the eigen decomposition required in Equation 2.1. For the kind of relational data considered in this thesis, the adjacency matrix X is very sparse and the reduced-rank reconstruction can be calculated efficiently. By exploiting matrix sparsity, Figure 2.2 shows that the SVD decomposition scales super-linearly with the number of nonzeros in X. Note that for an X-matrix with 10^5 rows, 10^6 columns, 10^7 nonzero elements and a predefined rank of 50, the computation only takes approximately 10 minutes on a standard laptop[7].

2.2 Additive Models

As described in Section 2.1.2, matrix factorization approaches predict the unknown relationships by completing a query relational adjacency matrix. In domains with multiple relations or attributes, represented as multiple matrices, one can improve the predictive performance by exploiting the relational information from the relational graph. Additive models [8] has been long used to reduce the model complexity

[7]This set of experiments were carried out by the coauthor Yi Huang of [42].

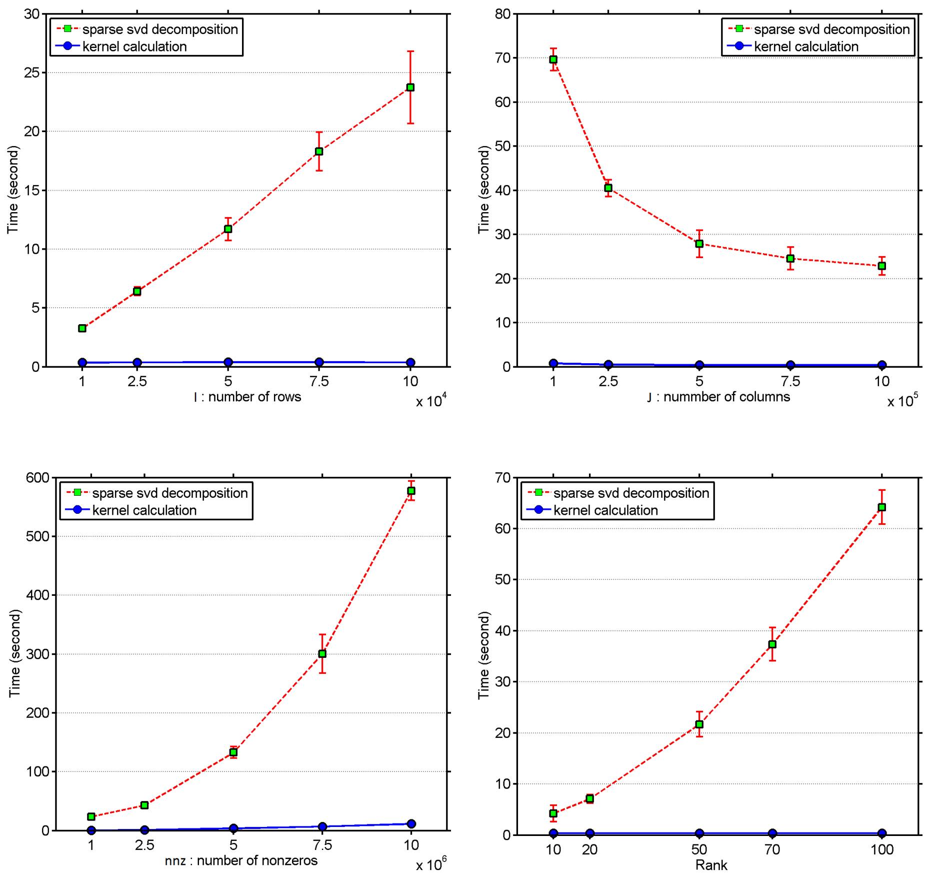


Figure 2.2: To evaluate the computation time for SVD decomposition on a sparse random $I \times J$ matrix X, four experiments were carried out respectively with regard to the number of rows (I), the number of columns (J), the number of nonzeros (nnz) and the predefined rank of X (Rank). The kernel matrix via $\text{ker} = XX^T$ was constructed and the sparse SVD was used on X. Each plot shows the experimental result on the computational time for SVD as a function with three fixed parameters, and a variational one. Basically, a sparse matrix X of size $10^5 \times 10^6$ with 10^6 nonzeros for a rank-50 SVD is under consideration. The top left figure (red dashed) shows computational time for the SVD as a function of I. An approximately linear dependency is observed which is related to the fact that the number of rows of U is I as well. The top right figure (red dashed) shows computation time for the SVD as a function of J. One can see a decrease. The reason is that with increasing J, XX^T becomes more sparse. The bottom left figure (red dashed) shows an approximately quadratic dependency of the computational time for the SVD on nnz. Note, that the last data point in the plot is a system with $nnz = 10^7$ requiring only 10 minutes of computation on a standard laptop. Finally, the bottom right figure (red dashed) shows the dependency on Rank. One can see that a 10 fold increase in Rank approximately displays a 10 fold increase in computational cost. Each figure also shows the computational time for calculating $\text{ker} = XX^T$, which, in comparison, is negligible (blue continuous).

by combining simple models as components in an additive way. This section extends the traditional additive models with each component initialized based on the low-rank approximations of the query relation matrix. The linear combinations of the predictions from all these components form the overall predictions for the query relation matrix. While the traditional additive models assume that each component corresponds to a regression surface describing the targets[8], this section assumes that each component is formed by the joint latent space spanned by its input matrix. Particularly, relational information will be represented in form of matrices used as inputs for the additive models.

2.2.1 Problem Formulation

Following Section 2.1.1, a query relational adjacency matrix $X \in \mathbb{R}^{I \times K}$ is derived from a sub graph of the relational graph $\mathcal{G}$ (Definition 1). Particularly the relation type in the query relation matrix X is specific to p, i.e. $K = 1$.

The task of relation prediction is to predict the existence of a relationship of type p among a set of nodes, i.e. to predict the truth of $x_{ij} = 1$ for the zero entries for relation type p. Essentially, it is to derive a matrix with the same size of X but replacing the zeros by continuous numbers which can be interpreted as the confidence value of $P(x_{ij}|\mathcal{G} = 1)$ being true. These continuous values can then be the basis for further analysis to tackle the tasks of classification and ranking.

In the following, relational information are in form of matrices $X^{(h)}$, $h = 1, \dots, H$, where H different kinds of relational information are available. Examples of relational information in the matrix form are observed attributes of the subject nodes, observed attributes of the object nodes and the instantiated relation of the subject-object pairs. For details of possible relational information, see Section 1.2.2. Relational information in form of a set of matrices will then be incorporated in the additive models.

[8]A target relation of interest is defined and the relationships (triples) involving the target relation are called the targets or the target triples in this thesis.

2.2.2 The Extended Additive Least-squares Model and Its Normal Equations

The standard parametric method for additive models is to predefine the form of functions f_h (e.g., to polynomial) for the components, and then estimates the parameters by least-squares. Here a nonparametric method based on H smoothed matrices will be considered. A smoother matrix $S^{(h)} : \mathbb{R}^{I\times J} \rightarrow \mathbb{R}^{I\times J}$ is a linear mapping [8]

$$\tilde{X}^{(h)} = S^{(h)}X \tag{2.7}$$

where h is the index of the smoothed matrix or the component, $\tilde{X}^{(h)}$ is the smoothed prediction matrix from component h. Traditional additive models can be extended for relational learning in a way that, each smoother $S^{(h)}$ is assumed to depend on a corresponding matrix $X^{(h)}$ derived from the relational information.

The assumption of additive models is that the overall prediction is formulated as a linear combination of predictions from the H components.

$$\tilde{X} = \sum_{h=1}^{H} \tilde{X}^{(h)} = \sum_{h=1}^{H} S^{(h)}X \tag{2.8}$$

which minimize

$$\min_{\{\tilde{X}^{(h)}|h=1,\dots,H\}} \quad \|X - \sum_{h} \tilde{X}^{(h)}\|_F^2 \tag{2.9}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

For solving the optimization problem 2.9, one can start with the least-squares fit for only one component (Equation 2.7)

$$\min_{\tilde{X}^{(h)}} \quad \|X - \tilde{X}^{(h)}\|_F^2 \ .$$

Recall that X is a sparse matrix. Efficient low rank approximation on X can be computed via SVD on $X^{(h)}$ (see Section 2.1.2 for details), i.e.

$$\tilde{X}^{(h)} = U_r \ D_r V_r^T \tag{2.10}$$

where r is the predefined rank of $X^{(h)}$, U_r and V_r are orthogonal matrices, $D_r =$

$\mathrm{diag}\{d_i\}_{i=1}^r$ contains the singular values of $X^{(h)}$.

The smoother matrix $S^{(h)}$ can be viewed as a projection to the column space of $X^{(h)}$ and describes the regression space of component h.

$$\begin{aligned}
S^{(h)} &= \tilde{X}^{(h)} \left((\tilde{X}^{(h)})^T \tilde{X}^{(h)} + \lambda I\right)^{-1} \left(\tilde{X}^{(h)}\right)^T \\
&= U_r D_r V_r^T \left(V_r D_r U_r^T U_r D_r V_r^T + \lambda I\right)^{-1} V_r D_r U_r^T \\
&= U_r \,\mathrm{diag}\left\{\frac{d_i^2}{d_i^2 + \lambda}\right\}_{i=1}^r U_r^T \\
&= X^{(h)}\, V_r \,\mathrm{diag}\left\{\frac{1}{d_i^2 + \lambda}\right\}_{i=1}^r V_r^T {X^{(h)}}^T
\end{aligned}$$

where λ is a regularizer, I is an identity matrix of size $r \times r$.

Hence, the predictions from component h can be derived from the U-matrix, i.e.

$$\tilde{X}^{(h)} = S^{(h)} X = U_r \,\mathrm{diag}\left\{\frac{d_i^2}{d_i^2 + \lambda}\right\}_{i=1}^r U_r^T X \qquad (2.11)$$

which equals to Equation 2.5 of the SUNS model.

Alternatively, the predictions from component h can be derived from the V-matrix.

$$\tilde{X}^{(h)} = S^{(h)} X = X^{(h)}\, V_r \,\mathrm{diag}\left\{\frac{1}{d_i^2 + \lambda}\right\}_{i=1}^r V_r^T {X^{(h)}}^T X \qquad (2.12)$$

Now the solution (Equation 2.11 and 2.12) of one component can be extended to solve the additive models of objective function 2.9. A sufficient condition for a solution is that the space of residuals $(X - \sum_h \tilde{X}^{(h)})$ is orthogonal to the regression space of the additive model. Since the regression space of the additive model depends on the space of each component h, equivalently, the regression space of component h is orthogonal to the residuals [8], i.e.

$$S^{(h)} \left(X - \sum_{h'=1}^{H} \tilde{X}^{(h')} \right) = 0 \,,$$

thus

$$\tilde{X}^{(h)} + \sum_{\bar{h} \neq h} S^{(h)} \tilde{X}^{(\bar{h})} = S^{(h)} X \,.$$

Equivalently, the following systems of normal equations are necessary and sufficient to minimize objective function 2.9.

$$\begin{pmatrix} I & S^{(1)} & S^{(1)} & \dots & S^{(1)} \\ S^{(2)} & I & S^{(2)} & \dots & S^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ S^{(H)} & S^{(H)} & S^{(H)} & \dots & I \end{pmatrix} \begin{pmatrix} \tilde{X}^{(1)} \\ \tilde{X}^{(2)} \\ \vdots \\ \tilde{X}^{(H)} \end{pmatrix} = \begin{pmatrix} S^{(1)}X \\ S^{(2)}X \\ \vdots \\ S^{(H)}X \end{pmatrix}$$

In [8], the Gauss-Seidel algorithm to update each $\tilde{X}^{(h)}$ at a time is provided.

$$\tilde{X}^{(h)}\text{new} \leftarrow S^{(h)} \left(X - \sum_{\bar{h}<h} \tilde{X}^{(\bar{h})}\text{new} - \sum_{\bar{h}>h} \tilde{X}^{(\bar{h})}\text{old} \right) \tag{2.13}$$

2.3 Link Prediction in Multi-relational Graphs Using Additive Models

As described in Section 1.1, there is a growing amount of data published in multi-relational graphs where information elements are represented as subject - relation type - object triples, i.e. (s, p, o) triples. Entities (subjects and objects) are represented as nodes and statements are represented as directed labeled links from subject node to object node. A machine learning task of some generality is the prediction of labeled links between entities using patterns in known labeled links in the relational graph.

The rest of this section is organized as the following: Section 2.3.1 will present a general framework for link prediction, including the cost function and the alternating least squares solution for parameter learning. Section 2.3.2 will discuss information aggregation to the general framework via joint operations, the inclusion of unstructured information, interaction terms and kernels. Section 2.3.3 will describe the approach for hyperparameter tuning. Section 2.3.4 will propose several algorithms for reference. Empirically, Section 2.3.5 will show the experimental results on synthetic data sets.

2.3.1 A General Framework for Link Prediction in Multi-relational Graphs

Considering a domain with N entities and K relation types, a general framework $\mathrm{M}_{\mathrm{add}}$ assumes that the truth value of a triple (s= i, p= j, o= k) can be estimated as a linear combination of directly related triples, defined as all triples (s= i, p= k', o= j') where i is the subject, all triples (s= i', p= k', o= i) where i is the object, all triples (s= j, p= k', o= j') where j is the subject and all triples (s= i', p= k', o= j) where j is the object. Finally, triples are considered with arbitrary relation types but where two entities from the target triple are involved, i.e. (s= i, p= k', o= j), and (s= j, p= k', o= i). $x_{i,j,k}$ is as defined in Section 2.1.1 and represents the truth value of triple (s= i, p= k, o= j). If $x_{i,j,k} = 0$, the predicted $\tilde{x}_{i,j,k} \geqslant 0$ can then be interpreted as a likelihood that the triple is true based on the immediate context of the triple.

To formulate the problem, a matrix $M = (X, [X]_T)$ is formed as shown in Figure 2.3, where $X = [X_1, X_2, \ldots, X_K]$, and $[X]_T = (X_1^T, X_2^T, \ldots, X_K^T)$ denotes the in-place matrix transposed of X. Following the model assumption, an additive model $\mathrm{M}_{\mathrm{add}}$ is formulated as

$$\tilde{x}_{i,j,k} = \sum_{l=1}^{2KN} w_{l,\,j+(k-1)N}\, m_{i,l} + \sum_{l=1}^{2KN} r_{l,\,i+(k-1)N}\, m_{j,l} + \sum_{l=1}^{2K} h_{l,k}\, m_{i,\,j+N(l-1)} \qquad (2.14)$$

where $m_{i,l}$ denotes $(M)_{i,l}$.

The $w_{\cdot,\cdot}$, $r_{\cdot,\cdot}$, and $h_{\cdot,\cdot}$ are model parameters to be estimated. It is assumed that the rows in M are exchangeable such that the weights $w_{l,\,j+(k-1)N}$ are independent of i, the weights $r_{l,\,i+(k-1)N}$ are independent of j, and the parameters $h_{l,j}$ are independent of both i and j. However, if type constraints on entities are considered in the model, the segmentation of parameter space can be changed. For example, the semantics of the relation type "like" is very different when the subject is a person than if the subject is a dog and the object is a bone. Technically this can be achieved by defining $w_{l,\,j+(k-1)N,\,type(i)}$, and correspondingly the model would have more parameters.

The three terms in the sum in Equation 2.14 exploits respectively the subject-specific information, the object-specific information, and the entity-pair information. The subject-specific information represents triples where i is the subject, when $l = 1, \ldots, K$

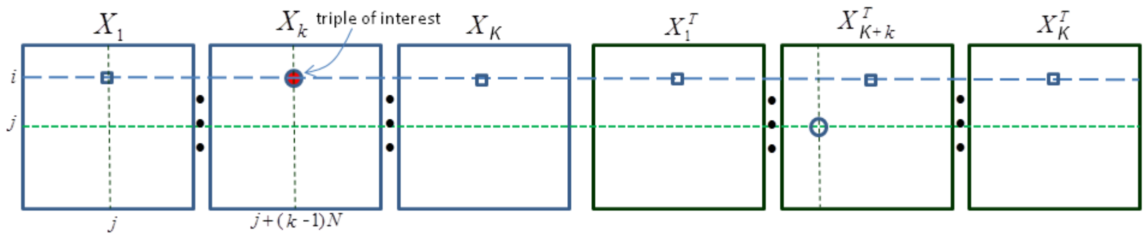


Figure 2.3: The figure shows the matrix M and illustrates the terms in Equation 2.14. The goal is to predict the matrix entry $\tilde{x}_{i,\ j,\ k}$ in matrix X_k indicated by the small red circle. The terms in the first sum (subject-specific information) are represented by the upper dashed blue line, the terms in the second sum (object-specific information) are represented by the lower green dotted line, and the terms in the third sum (entity-pair information) correspond to the small rectangles.

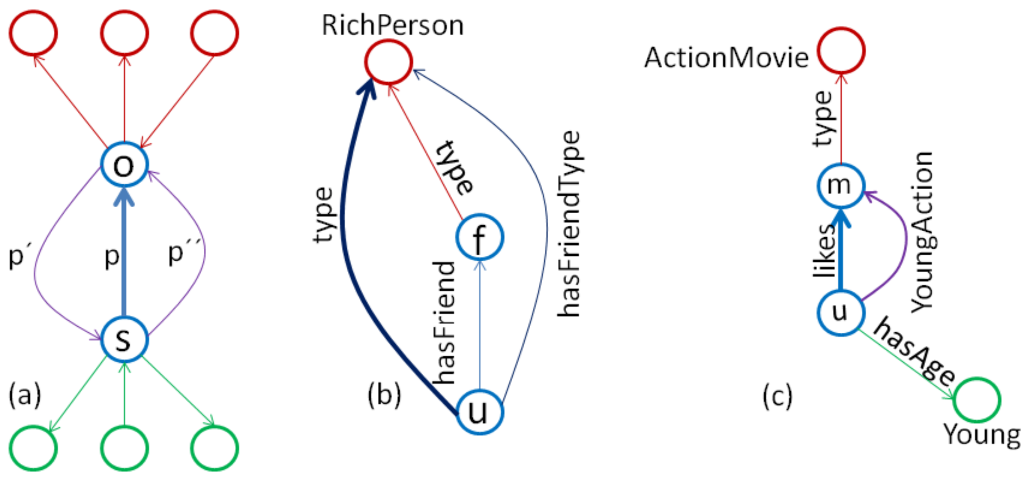


Figure 2.4: (a): The goal is to predict the likelihood of the triple (s, p, o). In Equation 2.14, the green triples attached to s correspond to the subject-specific information, the red triples attached to o correspond to the object-specific information, and the purple triples linking s and o correspond to the entity-pair information. (b): From the triple (u, hasFriend, f) and (f, type, richPerson) one can derive via aggregation (s, hasFriendType, RichPerson), which can be useful to predict (u, type, richPerson). (c): From (u, hasAge, Young) and (m, type, ActionMovie) one can derive (u, youngAction, m), which is useful for predicting (u, likes, m).

or where i is the object, when $l = K+1, \ldots, 2K$. Similarly, the object-specific information represents triples where j is the subject, when $l = 1, \ldots, K$ or where j is the object, when $l = K+1, \ldots, 2K$. The entity-pair information considers all triples that involve both i and j with any relation type (see Figure 2.4 (a)). Consider Equation 2.14 in terms of an if-then-rule where the right side of the equation describes the condition and the left side describes the conclusion. In this view the subject ?s and the object ?o would be variables[9] and the subject-specific information describes relations including the first variable, the object-specific information describes relations including the second variable, and the entity-pair information describes relations including both variables. All these variables are universally quantified, which means that the expression is valid for all subjects ?s and all objects ?o. It is possible to introduce additional variables in the condition part via exploiting further relational information in the relational graph, as described in Section 1.2.2, and these variables would be existentially quantified (as in Horn clauses).

2.3.1.1 Cost Function and Parameter Optimization

Equation 2.14 can be written efficiently in matrix form as

$$\tilde{X} = MW + [MR]_T + [M^{'} H]_{N \times KN} \tag{2.15}$$

where $M^{'} = ([X_1]_{\text{vec}}, \ldots, [X_K]_{\text{vec}}, [X_1^T]_{\text{vec}}, \ldots, [X_K^T]_{\text{vec}})$ is an $N^2 \times 2K$ matrix. The column vector $[.]_{\text{vec}}$ contains all elements of the corresponding relational adjacency matrix. $W \in \mathbb{R}^{2KN \times KN}$, $R \in \mathbb{R}^{2KN \times KN}$, and $H \in \mathbb{R}^{2K \times K}$ are parameter matrices. The operation $[.]_{N \times KN}$ transforms the result of the matrix product into a $N \times KN$ matrix.

A penalized least squares cost function for Equation 2.15 is defined as

$$\min_{W, R, H} \quad \|X - \tilde{X}\|_F^2 + \lambda_W \|W\|_F^2 + \lambda_R \|R\|_F^2 + \lambda_H \|H\|_F^2 \tag{2.16}$$

where $\|.\|_F$ is the Frobenius norm. The last three terms are used to regularize the solution to avoid overfitting.

[9]The common notation of indicating a variable by a question mark in front of a symbol is used. Here means variables as used in logics where they represent objects. In contrast, random variables in this thesis stand for the truth values of statements.

Considering each term in Equation 2.15 as a component to predict $\tilde{X}$, the proposed model M$_{\text{add}}$ can then be viewed as an additive model described in Section 2.2. To optimize the parameter matrices $\boldsymbol{W}$, $\boldsymbol{R}$, and $\boldsymbol{H}$ is the same as to optimize each term in Equation 2.15. Therefore, the solution 2.13 for the additive models can be used to solve the model M$_{\text{add}}$ (Equation 2.16). Particularly, predictions from components of $\tilde{X}^{(W)} = MW$, $\tilde{X}^{(R)} = [MR]_T$, and $\tilde{X}^{(H)} = [M^{'}\ H]_{N\times KN}$ need to be optimize.

Following the additive models in Section 2.2, SVD is performed firstly on each relational matrix to reduce computation and to further regularize the solution.

$$M = UDV^T \qquad M^{'} = U^{'}D^{'}(V^{'})^T \tag{2.17}$$

Only the leading singular values and corresponding singular vectors are used in the model. Another benefit of this low-rank approximation is that the model M$_{\text{add}}$ implicitly benefits from a sharing of statistical strengths leading to performance improvements, as it is well known from Latent Semantic Analysis (LSA).

The solution 2.13 for the additive models is similar to alternating least squares, where for each update all but one parameters are fixed. In the alternating least squares steps, three updates iterate until convergence.

$$MW \leftarrow U_r \operatorname{diag}\left\{\frac{d_i^2}{d_i^2 + \lambda_W}\right\}_{i=1}^{r} U_r^T \left(X - \tilde{X}^{(R)} - \tilde{X}^{(H)}\right) \tag{2.18}$$

$$MR \leftarrow U_r \operatorname{diag}\left\{\frac{d_i^2}{d_i^2 + \lambda_R}\right\}_{i=1}^{r} U_r^T \left[X - \tilde{X}^{(W)} - \tilde{X}^{(H)}\right]_T \tag{2.19}$$

$$M^{'}\ H \leftarrow U^{'}_{r^{'}} \operatorname{diag}\left\{\frac{(d^{'})_i^2}{(d^{'})_i^2 + \lambda_H}\right\}_{i=1}^{r^{'}} U^{'T}_{r^{'}} \left[X - \tilde{X}^{(W)} - \tilde{X}^{(R)}\right]_{N^2\times K} \tag{2.20}$$

where r and $r^{'}$ are respectively the predefined rank of M and $M^{'}$. Particularly, for the update in Equation 2.20 an solution in terms of the V-matrices might be more efficient (see Equation 2.12).

2.3.1.2 Computational Costs

Considering domains with several million entities, the computations seem to be expensive. Fortunately, relational adjacency matrices in many domains of interests are sparse. For example, nonzero elements are often restricted to one or a small number of blocks in the matrices due to type constraints; the user movie matrix is sparse since users have rated only a small percentage of all the movies. As the alternating least squares algorithm is based on SVD, it exploits sparse matrix algebra for calculating the decompositions and the computational benefits have been shown in Section 2.1.2.4. Furthermore, one could also apply the SVD to each relational adjacency matrix separately or to blocks of relational adjacency matrices, instead of $\boldsymbol{M}$. Also, if the entries of interests is in one particular relational adjacency matrix, only the parameters relevant to predicting the entries in that particular matrix need to be calculated. As the alternating least squares is employed, the convergence is quite fast, requiring fewer than 10 iterations.

2.3.2 Extensions

Relational information in different formats (structure and unstructured) can be included to the additive model $\mathrm{M_{add}}$ by means of aggregation, interaction teams and kernels.

Aggregation by Join Operations The triples represented in Equation 2.15 only consider the immediate neighborhood of the triples (s, p, o). It is easy to extend the formalism to also consider triples further away in the graph. As an example, consider the case that the likelihood that a person likes a movie is increased if at least one friend likes the movie. The latter information can be represented by a join operation on two adjacency matrices, i.e. $X_{\text{friendLikesMovie}} = \min(1, X_{\text{friendOf}} X_{\text{likes}})$ where min is applied component wise. Then $X_{\text{friendLikesMovie}}$ (and its transposed) is simply added as an additional relational adjacency matrix. Now one can model (via the subject-specific information in Equation 2.15) that a person might like "Action Hero 3" if at least one friend likes "Action Hero 3". With $M_i \in (X_1, \dots X_P, X_1^T, \dots X_K^T)$, the general form of an aggregated adjacency matrix is $X_a = \min(1, \prod_i M_i)$. Naturally, it is not feasible to consider an infinite set of matrix products. Possible approaches are that the user defines a small set of interesting candidates or that one applies structural search, e.g., by using approaches borrowed from the field of Inductive Logic Programming. Section 1.2.2.1 lists many more forms of aggregation. For example, instead of the ***min*** operation, one can count how many friends liked a movie, or what percentage of friends liked a movie. Here are two interesting examples involving join operations. First, with an assumption that a person tends to be rich if this person has a rich friend, the triple of interest is (?u, type, RichPerson). By joining (?u, hasFriend, ?f) and (?f, type, RichPerson), one can obtain a matrix that indicates if anybody of a person's friends is rich (see Figure 2.4 (b)). Second, with an assumption that a person often prefers restaurants of the nationality of that person, the triple of interest is (?u, likes, ?r). By joining (?u, hasNationality, ?c) and (?r, hasNationality, ?c), one can obtain a matrix that indicates if the user and the restaurant have the same nationality.

Contextual and Unstructured Data Sometimes there is contextual information available, often in textual form, that describe entities and relationships and can be exploited for link prediction. For example, one can use keywords in an entity's Wikipedia articles as attributes of that entity. The triples (s, itsWikiPageHasKeyword, Keyword) can simply be added as an additional relational adjacency matrix in the approach. If a keyword can be identified as an entity, then this information is even more valuable. Information extraction (IE) can also be used to extract triples from text and these triples can then be presented in matrix form as well. In the latter case, the entity-pair information in Equation 2.15 can be expected to be most valuable. For example, if the IE system extracts with high confidence that (Jack, knows, Jane), this could be information for predicting that (Jack, hasFriend, Jane).

Interaction Terms In Equation 2.15 a linear system is used, which is suitable in many high-dimensional domains. Of course, one can apply more general models such as neural networks as predictive models. Often this is unsuitable since the computational costs would explode. The proposed approach stays with a linear model but adds nonlinear interaction terms. As an example, assume that young users prefer action movies. A new triple (?u, YoungAction, ?m) is defined to be true if (?u, hasAge, Young) is true and (?m, type, ActionMovie) is true (see Figure 2.4 (c)). In general, the entity-pair information in Equation 2.14 can be expected to be most valuable here as well. To keep the number of these interaction terms small, a feature selection procedure evaluating the Pearson correlation between targets and interaction term is applied, as shown in Section 2.4 and 2.5.

Kernel Formulation So far the discussion focused on a representation in feature space. Here a representation in kernel space is discussed. A kernel formulation is appropriate for data in a multi-relational graph and suitable kernels are described in [7, 22, 62, 106]. Recall that kernel matrix $\mathrm{ker} = XX^T$ is calculated before performing the SVD for a matrix X. Naturally, one could start with a kernel matrix suitable for the multi-relational graph. In this view the alternating least squares solution in Section 2.3.1.1 is an efficient way of calculating a kernel solution with two kernels. The first one $\mathrm{ker}(.,.)$ involving two entities, either two subjects $\mathrm{ker}(s,s')$ or two objects $\mathrm{ker}(o,o')$. The second kernel $\mathrm{ker}'((s,o),(s',o'))$ involves two subject-object pairs. Given the corresponding kernel matrices ker and ker', one can decompose using SVD, i.e.

$$\mathrm{ker} = UDD^TU^T \quad \mathrm{ker}' = U'D'(D')^T(U')^T$$

and use the resulting terms in the update Equations 2.18 to 2.20.

2.3.3 Tuning of Hyperparameters

The proposed model $\mathrm{M}_{\mathrm{add}}$ contains several hyperparameters that need to be tuned (rank of approximations; regularization parameters). In [6] a random grid search is described for the best hyperparameters using cross-validation sets (i.e. tune the parameters on the evaluation set but not the test set).

2.3.4 Some Reference Models

For link prediction, both intrarelational correlations and contextual information can be helpful for the learning. The general framework $\mathrm{M}_{\mathrm{add}}$ exploits intrarelational correlations, contextual information respectively for the subjects, the objects and the subject-object pairs by learning from the matrix M and its extensions.

2.3.4.1 Exploiting Intrarelational Correlations

Intrarelational correlations exploits dependencies within the relation of interest and would correspond to the data dependencies exploited in typical collaborative learning systems. The leading approaches exploiting intrarelational correlations are based on a factorization of the query relational adjacency matrix $\boldsymbol{X}$, which is regularized SVD as described in Section 2.1.2 and is denoted as $\mathrm{M_{CF}}$.

2.3.4.2 Context Models

Contextual information consists of all other information sources except intrarelational correlations. As illustrations, three sources of contextual information are under consideration. The first one is multi-relational contextual information that is derived from the associated relational graph. The second one concerns information from unstructured sources, e.g., in form of textual documents describing the involved entities (e.g., from the entities' Wikipedia pages). The third one exploits the nonlinear interactions between the associated information sources. By using low-rank approximations in the context models, the models perform latent semantic analyses and can generalize across specific terms, i.e., the model might use similar latent representations for semantically related terms. Assume that contextual information is available from which one can derive an estimation of how likely a target relation is true, denoted by $f_{i,j}$. The corresponding matrix $\boldsymbol{F}$ with $(\boldsymbol{F})_{i,j} = f_{i,j}$ has the same dimensionality as $\boldsymbol{X}$ and entries typically assume values between zero and one, although this is not enforced.

Three context models that will be used in the applications are as following: Let $\boldsymbol{A}$ be a matrix with as many rows as $\boldsymbol{X}$, each row for a subject entity in $\boldsymbol{X}$. The columns of $\boldsymbol{A}$ represent features describing the subjects in $\boldsymbol{X}$. Similarly, $\boldsymbol{B}$ is a matrix with each row for an object entity in $\boldsymbol{X}$. The columns of $\boldsymbol{B}$ represent features describing the objects in $\boldsymbol{X}$. Finally, a matrix $\boldsymbol{C}$ is formed by the Kronecker product[10] $\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B}$, i.e., $\boldsymbol{C}$ contains all possible product terms of the elements of $\boldsymbol{A}$ and $\boldsymbol{B}$. The number of rows in $\boldsymbol{C}$ is the number of rows of $\boldsymbol{A}$ times the number of rows of $\boldsymbol{B}$, and the number of columns in $\boldsymbol{C}$ is the number of columns of $\boldsymbol{A}$ times the number of columns of $\boldsymbol{B}$. Following the example of Figure 2.4 (c), a column in $\boldsymbol{C}$ might indicate if a movie is an action movie and, at the same time, the user is young and the model might learn that young people like action movies.

To control overfitting, a regularized least squares cost function for the contextual

[10]see Definition 8 in Section 3.1

information F is formulated

$$\min_{W^{(A)}, R^{(B)}, H^{(C)}} \quad \|X - F\|_F^2 + \lambda_A \|W^{(A)}\|_F^2 + \lambda_B \|R^{(B)}\|_F^2 + \lambda_C \|H^{(C)}\|_F^2 \tag{2.21}$$

where

$$F = AW^{(A)} + [BR^{(B)}]_T + [CH^{(C)}]_{N \times KN} \quad .$$

Thus the entries in F are predicted as a linear combination of the subject features in A, the object features in B and the interaction features in C.

Let this model (Equation 2.21) be denoted as $\mathrm{F}_{\mathrm{all}}$. Including the intrarelational model M_{CF} as an additional fourth component to be optimized with F leads to the model $\mathrm{M}_{\mathrm{add}}$ introduced in Section 2.3.1. Additionally, one can consider more specific models from F that contains only one component, i.e. the model F_{A}, F_{B} and $\mathrm{F}_{\mathrm{INTER}}$ that contains only one component for contextual information respectively for the subject entities, the object entities and the interaction terms of subject-object pairs. Solutions for all these context models can be obtained using alternating least squares as model $\mathrm{M}_{\mathrm{add}}$.

2.3.4.3 Hierarchical Bayes

$\mathrm{M}_{\mathrm{add}}$ first smoothes each term in 2.16 via a regularized low-rank approximation and then globally optimizes each term. Alternatively, one can first add X and F and then smooth the resulting matrix. The later one is defined as a hierarchical Bayes model $\mathrm{M}_{\mathrm{HBS}}$ for the combination of contextual information with intrarelational correlation. The overall system $\mathrm{M}_{\mathrm{HBS}}$ is a low-rank approximation that minimizes

$$\min_{X_{\mathrm{HBS}}} \quad \|(X + F) - X_{\mathrm{HBS}}\|_F^2$$

Again, the solution can be based on the SUNS, in this case in the form of

$$X + F = U^{(X+F)} D^{(X+F)} \left(V^{(X+F)}\right)^T \tag{2.22}$$

and a regularized low-rank approach now leads to the model

$$X_{\text{HBS}} = U_r^{(X+F)} \operatorname{diag}\left\{ \frac{\left(d_i^{(X+F)}\right)^2}{\left(d_i^{(X+F)}\right)^2 + \lambda} \right\}_{i=1}^{r} U_r^{(X+F)^T} (X + F). \tag{2.23}$$

where r is the predefined rank of $X + F$, λ is a regularizer. The basic idea in this model is that contextual predictions should overwrite the zeros in X, prior to applying the factorization and reconstruction.

2.3.5 Experiments on Synthetic Data

Two synthetic data sets of different size have been generated according to the modeling assumptions. The target triples are a sum of four components: the first one is modeling the interlink correlation, the second one uses triples describing the subject entities, the third one uses triples describing the object entities, and the fourth one uses interaction terms. In addition, triples related to the subject are involved, i.e. describing subject attributes, and triples related to the object, i.e. describing object attributes. Moreover, interaction triples are generated by conjunctions on subject and object triples.

5-fold cross-validation was employed such that the subject entities are partitioned into training, validation and test sets. In the test and validation folds, one true relation was randomly selected to be treated as unknown (test statement) for each subject entity in the data set and all query entries were set to 0. In the test phase all unknown relations for the entity were then predicted, including the entry for the test statement. The test statement should obtain a high likelihood value, if compared to the other unknown entries. The normalized discounted cumulative gain (nDCG@all) [38] is a measure to evaluate a predicted ranking.

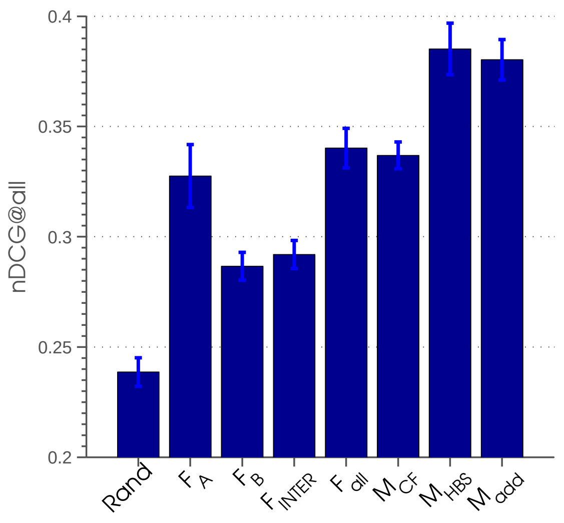
Figure 2.5 shows the results of using different relational adjacency matrices. In the first set of experiments, both the intrarelational correlation and the context models have significant predictive power and the two models (M_{add} and M_{HBS}) that uses all sources of information improve upon the subsystems. In the second set of experiments, the additive model M_{add} performs best. A model (M_{CF}) only using intrarelational information is quite strong. The reason is that, if sufficient amount of target triples are known to be true, the information on subject and object attributes is implicitly modeled in M_{CF} as well. This is a result also confirmed in the remaining experiments of this chapter: if M_{CF} is quite strong, adding subject and object information does not improve the model further, even when the latter might have predictive power. As a conclusion, the two combination schemes (M_{add} and M_{HBS}) seem to be more robust and perform well on both experiments.

2.4 Use Case: Relation Prediction in the Semantic Web

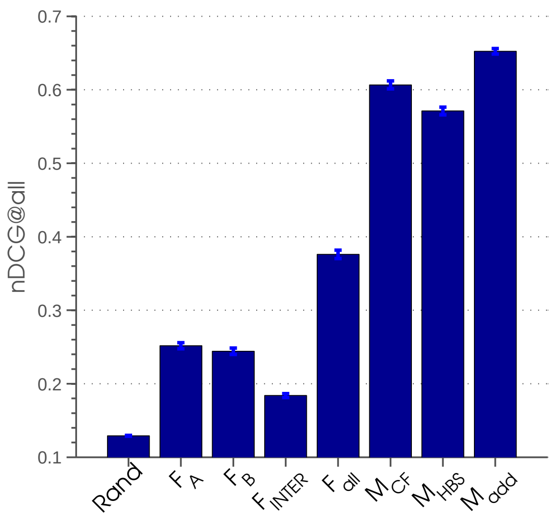
In the Semantic Web area, three common approaches for deriving or predicting instantiated relations are information extraction (IE), deductive reasoning (DR) and machine learning (ML). In IE, the relation of interest can be derived from unstructured data such as texts or images and the goal is to derive a mapping from unstructured input to statements. An IE system could be based on rules or on statistical classifiers. In addition to unstructured information, a knowledge base in form of a triple store of known facts forming an RDF graph might be available. Deductive reasoning is used to derive additional true statements when a set of facts and axioms is available. Relational machine learning also uses a set of true statements but estimates the truth values of novel statements by exploiting regularities in the data.

Powerful methods have been developed for all three approaches [16, 86] and all have their respective strengths and shortcomings. IE can only be employed if unstructured information is available that is relevant to a relation, deductive reasoning can only derive a small subset of all statements that are true in a domain and relational machine learning is only applicable if the data contains relevant statistical structure. Furthermore, multivariate prediction (such as SUNS) generalizes supervised learning to predict several variables jointly, conditioned on some inputs. The improved predictive performance in multivariate prediction, if compared to simple supervised learning, has been attributed to the sharing of statistical strength between the multiple tasks, i.e., data is used more efficiently (see [100] and citations therein for a review). Due to the large degree of sparsity of the relationship data in typical semantic graph domains, it is expected that multivariate prediction can aid the learning process in such domains.

The goal of this section is to combine all three methods, i.e. IE, DR and ML to



(a) Toydata



(b) Toydata1000

Figure 2.5: Test results on synthetic data. The first experiment (left) had 100 subjects and 80 objects, the subject attributes $\boldsymbol{A}$ had 80 columns, the object attributes $\boldsymbol{B}$ had 80 columns, and the interaction terms $\boldsymbol{C}$ had 8,000 rows and 4,000 columns. The three context models $\mathrm{F_A}$, $\mathrm{F_B}$, and $\mathrm{F_{INTER}}$ make valuable predictions significantly above random. $\mathrm{F_{all}}$ uses subject attributes, object attributes and interaction terms, but not intrarelational correlations in the target triples. The combination of all three context models, i.e. $\mathrm{F_{all}}$, is better than any of the individual context models. The context model and the intrarelation correlation are comparable strong in prediction: $\mathrm{M_{CF}}$ gives comparable results to $\mathrm{F_{all}}$. Both combination schemes ($\mathrm{M_{add}}$ and $\mathrm{M_{HBS}}$) are better than the intrarelational model or the context model on their own, so both combination schemes are sensible. There is no statistical significant difference of the performance between $\mathrm{M_{add}}$ and $\mathrm{M_{HBS}}$. The second experiment (right) had 1,000 subjects and 1,000 objects, the subject attributes $\boldsymbol{A}$ had 6 columns, the object attributes $\boldsymbol{B}$ had 7 columns, and the interaction terms $\boldsymbol{C}$ had 1,000,000 rows and 42 columns. It is shown that the additive method $\mathrm{M_{add}}$ is significantly better than the model that only relies on intrarelational correlations ($\mathrm{M_{CF}}$). The hierarchical Bayes model $\mathrm{M_{HBS}}$ does not significantly improve w.r.t. $\mathrm{M_{CF}}$. Predictions only based on subject attributes $\mathrm{F_A}$, only based on object attributes $\mathrm{F_B}$, and only based on interaction terms $\mathrm{F_{INTER}}$ are much better than random.

exploit all sources of available information by using the combination models M_{add} and M_{HBS} from Section 2.3. Technically, all triples that can either be inferred explicitly by calculating the deductive closure or on demand are added to the models. IE supplies evidence for the statements under consideration, such as the text describing the subject s (or object o) , the predicate[11] p or text that describes the (s, p, o) statement. And finally relational machine learning models the dependencies between statements.

The following two sets of experiments validate the two combination models using data from the YAGO ontology, and from Linked Life Data and Bio2RDF, all of which are part of the Linked Open Data (LOD) cloud. It shows that the combination models (M_{add} and M_{HBS}) are most effective when the information supplied via IE is complementary to the information supplied by statistical patterns in the structured data, and if reasoning can add relevant covariate information.

2.4.1 Predicting Writer's Nationality in YAGO

The first set of experiments were done on the YAGO 2 semantic knowledge base to evaluate the model M_{HBS}. YAGO is derived from Wikipedia and also incorporates WordNet and GeoNames. There are two available versions of YAGO 2: core and full. The chosen one for the experiments is the first one, which currently contains 2.6 million entities, and describes 33 million facts about these entities. The experiment was designed to predict the nationalities of writers. Four different types of writers were chosen: American, French, German and Japanese. E.g., the triples for American writers were obtained with the SPARQL query:

```
SELECT ?writer ?birthPlace ?location WHERE {
    ?writer rdf:type ?nationality .
    ?writer yago:wasBornIn ?birthPlace .
    ?birthPlace yago:isLocatedIn ?location .
    FILTER regex(str( ?nationality ), "American_writers", "i")
}
```

[11]In Semantic Web area, a predicate corresponds to a relation type in relational domains

440 entities representing the selected writers were obtained, of which 354 entities (i.e. writers) were selected with valid yago:hasWikipediaUrl statements. The following five models were under consideration:

- ML: A ML model using the relationship of the writers' nationality (in total 4) and information on the city where a writer was born. In total, there were 233 variables which stand for the truth values of statements of the writers' nationality. No unstructured information is used in this ML model.
- IE: As textual source, the Wikipage of the writers was used. The terms "German, French, American, Japanese" were removed and it ended up with 36,943 keywords. An IE system using SUNS was built.
- ML+IE: The knowledge base with IE were combined in a way that the predicted triples derived from unstructured information are added to the adjacency matrices formed by the knowledge base. Then Equation 2.22 and Equation 2.23 were applied.
- ML+AGG: Geo-reasoning was performed to derive the country where a writer is born (from the city that a writer was born). This aggregate information was added as a statement to the writer. Naturally, a high correlation between country of birth and the writer's nationality is expected (but there is no 100% agreement!). Thus prior to machine learning, triples derived from the knowledge base were added.
- ML+AGG+IE: As ML+AGG but the predicted triples from IE were added.

10-fold cross validation was performed for each model such that the entries of the query relation were partitioned into training, validation and test sets. In the test and validation folds all entries in the query relation were set to 0. The models were evaluated with the area under precision and recall curve.

Figure 2.6 shows the results. As there are only 4 nationalities, which are almost always mutual exclusive (there is a small number of writers with more than one nationality), the intrarelational correlation (ML contribution) is quite weak and the country attributes were not used. But one can see that ML improved significantly by adding information on the country of birth (ML+AGG). The IE component gives excellent performance but ML improves the results by approximately 3 percentage points. Finally, by including geo-reasoning, the performance can be improved by another percentage point. This is a good example where all three components, geo-reasoning, IE and machine learning could be combined together for the overall best predictions.

2.4.2 Predicting Relationships between Diseases and Genes

In this set of experiments, information on known gene-disease patterns was extracted from Linked Life Data and Bio2RDF, which is from the LOD cloud and is in form of the triples (Gene, related_to, Disease). In total, $2,462$ genes and 331 diseases were considered. For genes $11,332$ features and for the diseases $1,283$ features were extracted from the LOD cloud. In addition, $8,000$ textual features describing genes and $3,800$ textual features describing diseases from corresponding text fields were retrieved in Linked Life Data and Bio2RDF. The products of any gene feature and any disease feature generate $49,801,621$ potential gene-disease interaction terms (Section 2.3.2) which were reduced to $1,132$ by using a fast feature selection procedure evaluating the Pearson correlation between targets and interaction term. The two tasks here are to predict diseases that are likely associated with a gene, and to rank genes based on their predicted relevance for a given disease.

5-fold cross validation was performed for each model, following the same experimental protocol as described in Section 2.3.5.

Figure 2.7 shows the results. One can see that combining IE with machine learning is effective in applications where a large number of relationships need to be predicted. When predicting diseases for genes, the contextual information (reflected in $\mathrm{F_{all}}$, from IE but no DR available) and the ML part (reflected in $\mathrm{M_{CF}}$) are both equally strong. In most data sets, one of the two is dominating. Both combination schemes ($\mathrm{M_{add}}$ and $\mathrm{M_{HBS}}$) are the leading approaches and provide results significantly better than $\mathrm{F_{all}}$ or $\mathrm{M_{CF}}$ on their own. Predicting genes for diseases generally gives a weaker nDCG score.

2.5 Use Case: Recommendation Systems

Section 2.3 has considered the problem of predicting instantiated binary relations in a multi-relational setting and exploits both intrarelational correlations and contextual information. A general framework for predicting links in multi-relational graphs has

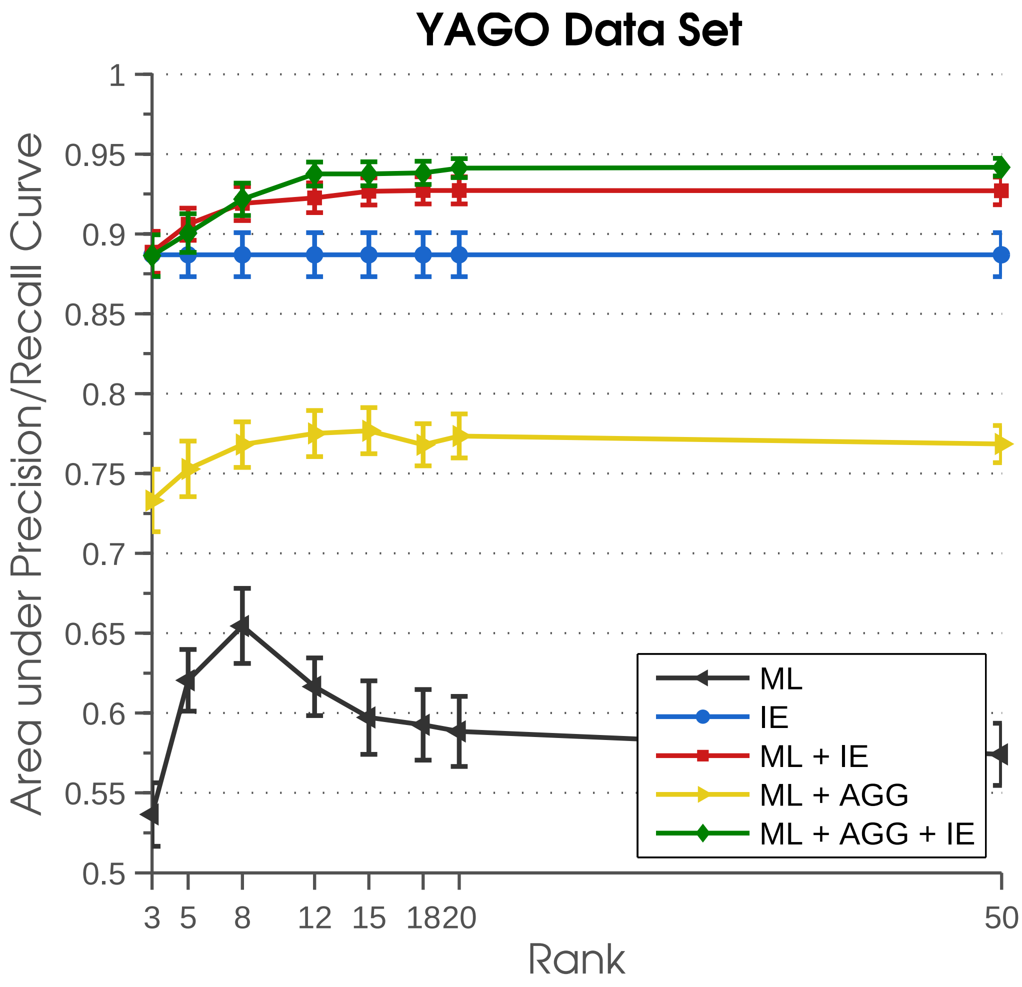
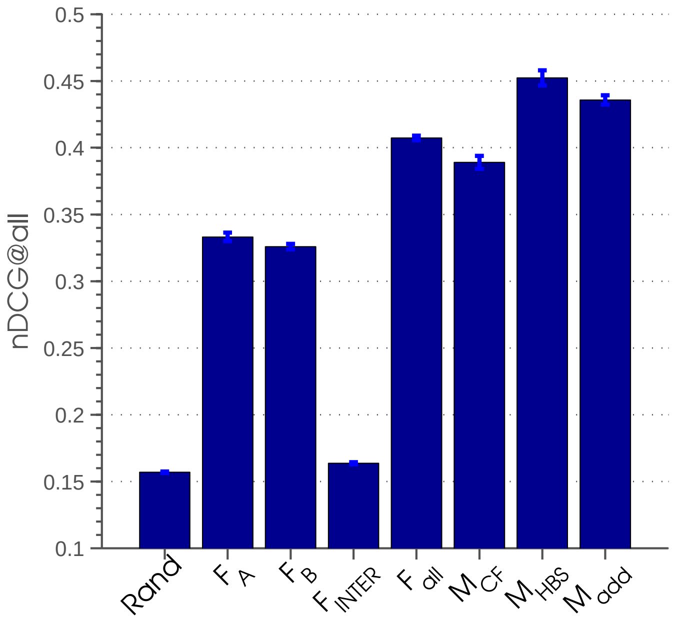
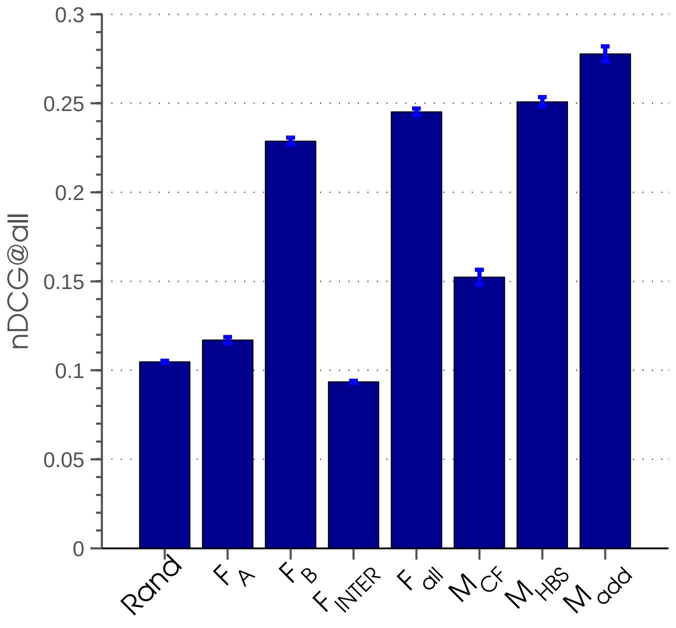


Figure 2.6: The area under curve for the YAGO 2 Core experiment as a function of the predefined rank r of the approximation.



(a) Gene-Disease



(b) Disease-Gene

Figure 2.7: The goal is to predict the relationship between genes and diseases. The first set of experiments ranked recommended diseases for genes and the second set of experiments ranked recommended genes for diseases. Contextual features from disease attributes F_A and from gene attributes F_B, and contribution from the interaction term F_{INTER} were considered. In both tasks, the combination of all context models in F_{all} is better than the individual context models where F_{INTER} is quite weak. For the first task, both combination schemes are better than the intrarelational model (M_{CF}) or the context model (F_{all}) on their own, so both combination schemes are sensible. The hierarchical Bayes models, i.e. M_{HBS}, gives the best result. The results are generally better than the results reported in [42] since, there, only contextual features from text documents were used. The second task, predicting genes for diseases, is more difficult due to the great number of potential genes. Intrarelational correlation (M_{CF}) on its own is relatively weak. Again, both combination schemes give good results.

been presented. It is shown that efficient learning can be achieved using an alternating least squares approach. A number of sensible algorithms were presented, which are all modular and have unique solutions. Particularly, two combination schemes ($\mathrm{M}_{\mathrm{add}}$ and $\mathrm{M}_{\mathrm{HBS}}$) have been shown to be effective and there is no clear best approach, although there seems to be a general advantage for combing the strengths of exploiting intrarelational correlation and contextual information.

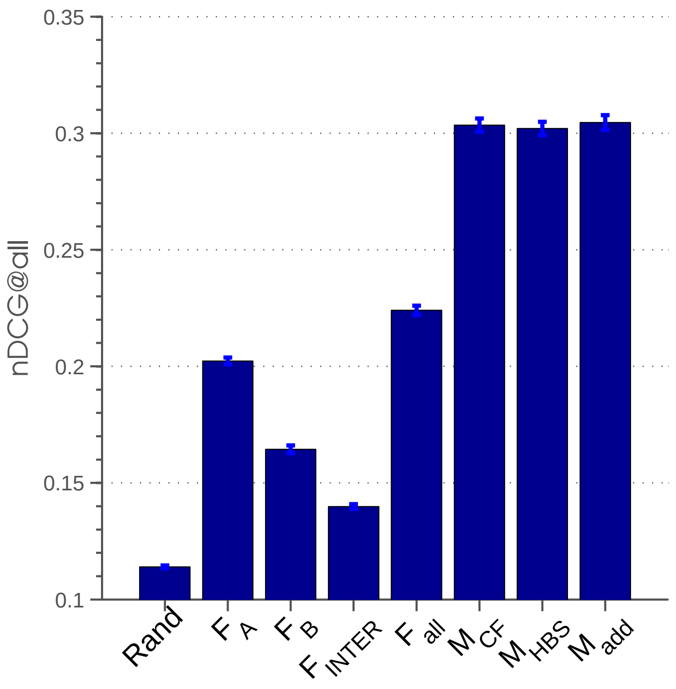
In the applications of recommendation systems, contextual information could be extracted from the database and from textual data. One can simply treat the keywords in the textual descriptions as additional features describing subjects or objects. In some applications, it is useful to add aggregated information. This can be represented as additional features as well.

3-fold cross validation was performed for each model in two sets of experiments, following the same experimental protocol as described in Section 2.3.5. The goal is to show the improvement on predictive performance by including prior knowledge to exploit both intrarelational correlations and contextual information for recommending movies and books to users. An interesting observation is that sometimes there are significant contributions from the contextual information, but that this information also seems to be reflected in the intrarelational correlations, so adding contextual information to a model that uses intrarelational correlations does not further improve performance. In general, correlated information or information that is represented several times does not hurt performance, since a regularization is employed.

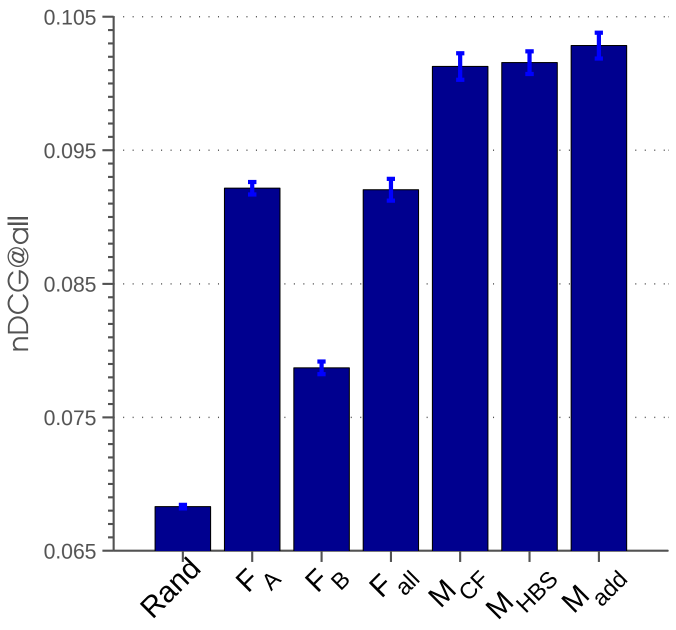
2.5.1 Modeling MovieLens Data

943 users and 1,600 movies from the MovieLens data set[12] were used to evaluate if a user has seen a movie or not. 99 user attributes were derived from age (5 classes), gender (2 classes) and occupation (21 classes). The 89 movie attributes were derived from genre, release month and keywords extracted from the Wikipedia pages. Figure 2.8 (a) shows the results. Although the attribute information on the movies and the users have predictive power (significantly above random), a model exploiting intrarelational correlations (M_{CF}) gives very good performance and the two combination models ($\mathrm{M}_{\mathrm{add}}$ and $\mathrm{M}_{\mathrm{HBS}}$) cannot improve beyond the performance of M_{CF}.

[12]http://www.grouplens.org/node/73



(a) MovieLens



(b) Book-Crossing

Figure 2.8: (a) Experiments on movielens data. F_A describes the modeling performance using only the attribute information on the movies and F_B describes the modeling performance using only the attribute information on the users. Although the attribute information on the movies and the users have predictive power (significantly above random), a model exploiting intrarelational correlations (M_{CF}) gives very good performance and the two combination models (M_{add} and M_{HBS}) can not significantly improve beyond the performance of M_{CF}. As in the experiment on the synthetic data, there is information in the attribute data but this information is also represented in M_{CF}. (b) Experiments on the book-crossing data set. The same trends can be seen as in the movielens experiments although the additive model M_{add} seems to improve on the model M_{CF}, which only exploits intrarelational correlations.

As in the experiment on one synthetic data in Section 2.3.5, there is information in the contextual data but this information is also represented in M_{CF}. Additionally, $8,811$ potential interaction terms were reduced to 200 by using a fast feature selection procedure evaluating the Pearson correlation between targets and interaction term.

2.5.2 Modeling Book Preferences

The BookCrossing data set[13] was used to predict if a user rated a book. The data set consisted of $105,283$ users and $340,554$ books. A user is described by $5,849$ attributes (derived from age and city, province and country) and a book is described by $24,508$ attributes (authors, publication year, publisher). The goal is to predict if a user would rate (i.e. read) a book. The results in Figure 2.8 (b) show that the additive model $\mathrm{M}_{\mathrm{add}}$ gives best results.

[13]http://www.bookcrossing.com

Chapter 3

Learning from Latent and Observable Patterns on Multi-relational Data via Tensors

Representing the multi-relational data by a 3-way tensor is a more natural way compared to using a matrix, as a third dimension models the interactions of entities under different relation types. This chapter starts with an introduction to tensors and tensor decompositions. What follows is an analysis on the computational complexity of tensor models, to show that a lower tensor rank guarantees fast tensor decompositions. As the rank of a tensor also determines its generalization ability, a too low rank may not be sufficient to represent the data. Thus there is a theoretical consideration about under which conditions factorizations are efficient for the learning. Based on these discussions, a novel tensor decomposition approach ARE is proposed which stands for Additive Relational Effects. ARE combines the strengths of tensor factorizations (particularly RESCAL) and prior knowledge in an additive way to discover different relational effects from the networked data.

The rest of this chapter is organized as following: Section 3.1 will provide an overview of high-order tensor decompositions, including the basic operations and some well-established tensor models. Section 3.2 will discuss under which conditions a tensor decomposition requires a high or a low rank. Section 3.3 will present the novel Additive Relational Effects model, including a scalable algorithm for computation and evaluations on several benchmark data sets to show that ARE outperforms many state-of-the-art relational learning algorithms in both runtime and predictive performance.

3.1 An Introduction to Tensors and Tensor Decompositions

High-dimensional modeling is becoming ubiquitous across the sciences and industry due to its descriptive flexibility for multi-relational data. The advances in parallel computational technology increase the feasibility of high-dimensional modeling for large scale learning problems. A tensor is a multi-dimensional array that can be used for high-dimensional modeling. Tensor decompositions were popular early in the field of chemometrics and psychometrics [104], and in recent years they received more attention also from machine learning and data mining community. This section will briefly review definitions and properties of tensors, and some well-established tensor models. For a more detailed and complete introduction to tensor decompositions, one can refer to the survey paper from Kolda and Bader [52].

3.1.1 Notation and Preliminaries

An N-way or Nth-order tensor can be represented as a multi-dimensional array of numerical values. The order (also known as ways or modes) of a tensor is the number of dimensions. Particularly, a matrix is a second-order tensor, a vector is a first-order tensor, and a scalar is a zero-order tensor. Tensor elements in form of a scalar, a vector and a matrix can be extracted from a tensor by different ways of indexing.

Definition 4 (Entries)**.** *Fixing all indices of elements in a tensor generates a scalar. Each entry in a N-order tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is indexed by a N-tuple of subscripts, $(i_1, i_2, \cdots, i_N)$. Element $(i_1, i_2, \cdots, i_N)$ of the tensor $\mathbf{X}$ is denoted as $x_{i_1 i_2 \cdots i_N}$. The i-th entry of a vector $\boldsymbol{x} \in \mathbb{R}^I$ is donated as x_i. Element (i, j) of a matrix $X \in \mathbb{R}^{I \times J}$ is denoted as x_{ij}. Element (i, j, k) of a third-order tensor $\mathbf{X} \in \mathbb{R}^{I \times J \times K}$ is denoted as x_{ijk}.*

Definition 5 (Fibers [52])**.** *Fixing all but one indices of elements in a tensor generates a vector, which is also known as a fiber. A matrix column $\boldsymbol{x}_{:j}$ (also denoted as*

$\boldsymbol{x}_j$) of matrix $X \in \mathbb{R}^{I \times J}$ is a mode-1 fiber and the matrix row $\boldsymbol{x}_{i:}$ is a mode-2 fiber. A third order tensor $\mathbf{X} \in \mathbb{R}^{I \times J \times K}$ has column fibers $\boldsymbol{x}_{:jk}$, row fibers $\boldsymbol{x}_{i:k}$ and tube fibers $\boldsymbol{x}_{ij:}$.

Definition 6 (Slices [52]). *Fixing all but two indices of elements in a tensor generates a matrix, which is also known as a slice. The horizontal, lateral, and frontal slices of a third-order tensor $\mathbf{X} \in \mathbb{R}^{I \times J \times K}$ are respectively denoted as $X_{i::}$, $X_{:j:}$, and $X_{::k}$. Alternatively, the k-th frontal slice $X_{::k}$ will be denoted as X_k for convenience.*

As consequences, some basic operations defined over scalars, vectors or matrices can be carried on with tensors.

Definition 7 (Rank-one Tensor [52]). *An N-order tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is called a rank-one tensor if it can be expressed via the outer product of N vectors.*

$$\mathbf{X} = \boldsymbol{a}^{(1)} \circ \boldsymbol{a}^{(2)} \circ \cdots \circ \boldsymbol{a}^{(N)}$$

where $\circ$ is the vector outer product. Equivalently, each element of $\mathbf{X}$ is the product of the corresponding elements from the vectors.

$$x_{i_1 i_2 \cdots i_N} = a^{(1)}_{i_1} \circ a^{(2)}_{i_2} \circ \cdots \circ a^{(N)}_{i_N}$$

Figure 3.1 shows an example of a third-order rank-one tensor $\mathbf{X} \in \mathbb{R}^{I \times J \times K}$. Particularly, SVD decomposition (Equation 2.1) of a matrix $X \in \mathbb{R}^{I \times J}$ can be viewed as a sum of rank-one tensors, i.e. $X = \sum_{p=1}^{r} d_p\, u_p \circ v_p$.

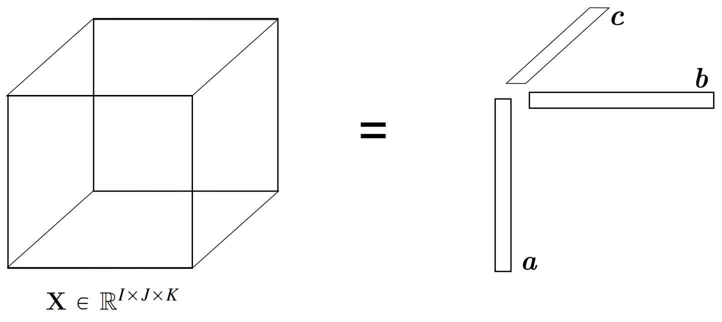


Figure 3.1: This picture is adopted from [52]. A third-order rank-one tensor $\mathbf{X} = \boldsymbol{a} \circ \boldsymbol{b} \circ \boldsymbol{c}$, where each element can be computed via $x_{ijk} = a_i b_j c_k$, for $i = 1, \ldots, I$, $j = 1, \ldots, J$, $k = 1, \ldots, K$.

Definition 8 (Kronecker Product [52]). *The Kronecker product of matrices $A \in \mathbb{R}^{I \times J}$*

and $B \in \mathbb{R}^{K \times L}$ is denoted as $A \otimes B$, and can be computed via

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1J}B & \\ a_{21}B & a_{22}B & \cdots & a_{2J}B & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \cdots & a_{IJ}B & \end{bmatrix}$$

which is a matrix of size $IK \times JL$. Equivalently,

$$A \otimes B = [a_1 \otimes b_1 \quad a_1 \otimes b_2 \quad a_1 \otimes b_3 \quad \cdots \quad a_J \otimes b_{L-1} \quad a_j \otimes b_L]$$

where for vectors, the operation of Kronecker product equals to the outer product $\circ$.

Definition 9 (Khatri-Rao Product [52])**.** *The Khatri-Rao product of matrices $A \in \mathbb{R}^{I \times K}$ and $B \in \mathbb{R}^{J \times K}$ is denoted as $A \odot B$, and can be computed via*

$$A \odot B = [\boldsymbol{a}_1 \otimes \boldsymbol{b}_1 \quad \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 \quad \cdots \quad \boldsymbol{a}_K \otimes \boldsymbol{b}_K]$$

which is a matrix of size $IJ \times K$. The operation of Khatri-Rao Product can be viewed as the "matching columnwise" Kronecker product.

Definition 10 (n-mode unfolding [52])**.** *Unfolding of a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, also known as matricization or flattening, is to reorder the elements of $\mathbf{X}$ into a matrix. Particularly, the n-mode unfolding of the tensor $\mathbf{X}$ is denoted as $X_{(n)}$ and the columns of $X_{(n)}$ are the mode-n fibers of $\mathbf{X}$.*

Definition 11 (n-mode product [52])**.** *The n-mode product of a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $U \in \mathbb{R}^{J \times I_n}$ is denoted as $\mathbf{X} \times_n U$ and can be expressed in terms of n-mode unfolding of $\mathbf{X}$.*

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

The n-mode product of a tensor and a matrix is then to multiply each mode-n fiber of $\mathbf{X}$ by a matrix U. This operation is related to a change of basis via U when the tensor $\mathbf{X}$ defines a multi-linear operation.

As an example[1] to illustrate most of the aforementioned concepts, let the frontal slices

[1]The example is adopted from [52].

of $\mathbf{X} \in \mathbb{R}^{3\times4\times2}$ be

$$X_1 = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \qquad X_2 = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}.$$

Thus the mode-1 fibers of $\mathbf{X}$ are vectors of

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}, \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix}, \begin{bmatrix} 10 \\ 11 \\ 12 \end{bmatrix}, \begin{bmatrix} 13 \\ 14 \\ 15 \end{bmatrix}, \begin{bmatrix} 16 \\ 17 \\ 18 \end{bmatrix}, \begin{bmatrix} 19 \\ 20 \\ 21 \end{bmatrix}, \begin{bmatrix} 22 \\ 23 \\ 24 \end{bmatrix}.$$

The mode-2 fibers of $\mathbf{X}$ are vectors of

$$\begin{bmatrix} 1 \\ 4 \\ 7 \\ 10 \end{bmatrix}, \begin{bmatrix} 2 \\ 5 \\ 8 \\ 11 \end{bmatrix}, \begin{bmatrix} 3 \\ 6 \\ 9 \\ 12 \end{bmatrix}, \begin{bmatrix} 13 \\ 16 \\ 19 \\ 22 \end{bmatrix}, \begin{bmatrix} 14 \\ 17 \\ 20 \\ 23 \end{bmatrix}, \begin{bmatrix} 15 \\ 18 \\ 21 \\ 24 \end{bmatrix}.$$

The mode-3 fibers of $\mathbf{X}$ are vectors of

$$\begin{bmatrix} 1 \\ 13 \end{bmatrix}, \begin{bmatrix} 2 \\ 14 \end{bmatrix}, \begin{bmatrix} 3 \\ 15 \end{bmatrix}, \begin{bmatrix} 4 \\ 16 \end{bmatrix}, \begin{bmatrix} 5 \\ 17 \end{bmatrix}, \quad \vdots \quad, \begin{bmatrix} 9 \\ 21 \end{bmatrix}, \begin{bmatrix} 10 \\ 22 \end{bmatrix}, \begin{bmatrix} 11 \\ 23 \end{bmatrix}, \begin{bmatrix} 12 \\ 24 \end{bmatrix}.$$

The 1-mode, 2-mode and 3-mode unfoldings of $\mathbf{X}$ are respectively

$$X_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}$$

,

$$X_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix}$$

,

$$X_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5\cdots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17\cdots & 21 & 22 & 23 & 24 \end{bmatrix}.$$

Given a matrix $U = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$, the 1-mode product of $\mathbf{X}$ and U is a tensor $\mathbf{Y} \in \mathbb{R}^{2\times 4\times 2} = \mathbf{X} \times_1 U$, with frontal slices

$$\mathbf{Y}_1 = \begin{bmatrix} 22 & 49 & 76 & 103 \\ 28 & 64 & 100 & 136 \end{bmatrix}, \qquad \mathbf{Y}_2 = \begin{bmatrix} 130 & 157 & 184 & 211 \\ 172 & 208 & 244 & 280 \end{bmatrix}$$

Given a vector $\boldsymbol{v} = [1 \quad 2 \quad 3 \quad 4]^T$, the 2-mode product of $\mathbf{X}$ and $\boldsymbol{v}$ is a matrix $Y \in \mathbb{R}^{3\times 2}$.

$$Y = \begin{bmatrix} 70 & 190 \\ 80 & 200 \\ 90 & 210 \end{bmatrix}$$

Finally, three definitions are important for defining an optimization problem for a tensor.

Definition 12 (Tensor Norm [52])**.** *The norm of a tensor* $\mathbf{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ *is denoted as* $\|\mathbf{X}\|$ *and can be calculated via the square root of the sum of the squares of all its elements.*

$$\|\mathbf{X}\| = \sqrt{\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_N=1}^{I_N} x^2_{i_1 i_2\cdots i_N}}$$

which corresponds to the Frobenius norm $\|X\|$ *for a matrix* $X \in \mathbb{R}^{I\times J}$.

Definition 13 (Tensor Rank [52])**.** *The rank of a tensor* $\mathbf{X} \in \mathbb{R}^{I_1\times I_2\times\cdots\times I_N}$ *is denoted as* $\operatorname{rank}(\mathbf{X})$ *and is defined as the minimal number of rank one tensors required to generate* $\mathbf{X}$ *as their sum.*

$$\operatorname{rank}(\mathbf{X}) = \min\left\{ r \;\middle|\; \mathbf{X} = \sum_{p=1}^{r} \boldsymbol{a}_p^{(1)} \circ \boldsymbol{a}_p^{(2)} \cdots \circ \boldsymbol{a}_p^{(N)} \right\}$$

where $\boldsymbol{a}_p^{(n)} \in \mathbb{R}^{I_n}$, $n = 1,\ldots,N$. *Particularly, the tensor rank of a third-order tensor* $\mathbf{X} \in \mathbb{R}^{I\times J\times K}$ *is defined as*

$$\operatorname{rank}(\mathbf{X}) = \min\left\{ r \;\middle|\; \mathbf{X} = \sum_{p=1}^{r} \boldsymbol{a}_p \circ \boldsymbol{b}_p \circ \boldsymbol{c}_p \right\}$$

where $\boldsymbol{a}_p \in \mathbb{R}^I$, $\boldsymbol{b}_p \in \mathbb{R}^J$ *and* $\boldsymbol{c}_p \in \mathbb{R}^K$.

Definition 14 (n-Rank [52])**.** *The* n-rank *(also called multi-linear rank) of a tensor* $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ *is denoted as* n-rank($\mathbf{X}$) *and is defined by a tuple* $(r_1, r_2, \ldots, r_N)$*, where* r_n *is the rank of the matrix* $X_{(n)}$*,* $n = 1, \ldots, N$*.*

3.1.2 Tensor Decompositions

3.1.2.1 The CANDECOMP/PARAFAC Decomposition

The CANDECOMP/PARAFAC (CP) decomposition factorizes a tensor into a finite sum of rank-one tensors. The idea of CP was earliest proposed by Hitchcock in 1927 [34, 35], and became popular after the introduction by Carroll and Chang [11], and Harshmann [28] in 1970. CP has been applied in the fields of psychometrics, chemometrics, sensor array processing, telecommunications, neuroscience, image compression and classification, and many more applications that can be seen in [52].

Figure 3.2 shows the CP decomposition of a third-order tensor $\mathbf{X} \in \mathbb{R}^{I \times J \times K}$, which can be formulated as

$$\mathbf{X} \approx \sum_{p=1}^{r} \boldsymbol{a}_p \circ \boldsymbol{b}_p \circ \boldsymbol{c}_p$$

where rank-one components $\boldsymbol{a}_p \in \mathbb{R}^I$, $\boldsymbol{b}_p \in \mathbb{R}^J$, $\boldsymbol{c}_p \in \mathbb{R}^K$, and $r \in \mathbb{Z}^+$ is the predefined rank for the decomposition. Latent factor matrices of CP refer to matrices where the columns consist of rank-one components, i.e. $A = [\boldsymbol{a}_1 \quad \boldsymbol{a}_2 \quad \ldots \quad \boldsymbol{a}_r]$ and a similar way for $\boldsymbol{B}$ and $\boldsymbol{C}$. Correspondingly, each element of $\mathbf{X}$ can be computed via

$$x_{ijk} \approx \sum_{p=1}^{r} a_{ip} b_{jp} c_{kp}$$

where $i = 1, \ldots, I$, $j = 1, \ldots, J$, $k = 1, \ldots, K$.

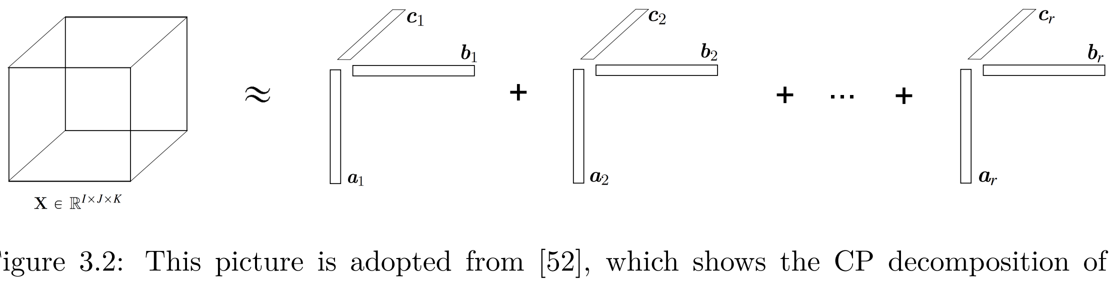


Figure 3.2: This picture is adopted from [52], which shows the CP decomposition of a third-order tensor $\mathbf{X} \approx \sum_{p=1}^{r} \boldsymbol{a}_p \circ \boldsymbol{b}_p \circ \boldsymbol{c}_p$.

The CP decomposition is basically unique, which means that given a predefined rank r the rank-one components are unique except the scaling and permutation indeterminacies. This uniqueness property is interesting and helpful for interpreting the latent components.

The workhorse algorithm for CP is the alternating least squares (ALS) method [11, 28]. Algorithm 1 illustrates the CP-ALS of a third-order tensor.

Algorithm 1 CP-ALS for Third-order Tensors [11, 28]

```
Require: $\mathbf{X} \in \mathbb{R}^{I \times J \times K}$, $r$
1: Initialize $A \in \mathbb{R}^{I \times r}$, $B \in \mathbb{R}^{J \times r}$, $C \in \mathbb{R}^{K \times r}$
2: repeat
3:     $A \leftarrow X_{(1)}(C \odot B)(C^T C * B^T B)^{\dagger}$, normalize columns of $A$
4:     $B \leftarrow X_{(2)}(C \odot A)(C^T C * A^T A)^{\dagger}$, normalize columns of $B$
5:     $C \leftarrow X_{(3)}(B \odot A)(B^T B * A^T A)^{\dagger}$, normalize columns of $C$
6: until fitness ceases to improve or maximum iterations achieved
7: return $A$, $B$, $C$, and their norms $\lambda$
```

3.1.2.2 The Tucker Decomposition

The Tucker decomposition factorizes a tensor into a core tensor transformed by factor matrices, each of which is for one mode of the tensor. The idea of Tucker decomposition was first brought by Tucker in 1963 [105], and the model was refined in subsequent works from Levin [60] and Tucker [102, 103]. Tucker decomposition has been applied in the fields of chemical analysis, psychometrics, signal processing, computer vision, and many more applications that can also be found in [52].

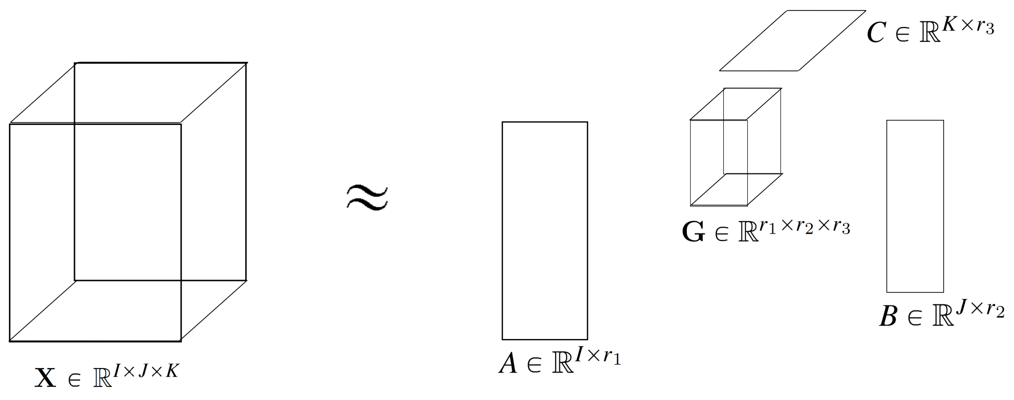
Figure 3.3 shows the Tucker decomposition of a third-order tensor $\mathbf{X} \in \mathbb{R}^{I \times J \times K}$, which can be formulated as

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

where $A \in \mathbb{R}^{I \times r_1}$, $B \in \mathbb{R}^{J \times r_2}$, $C \in \mathbb{R}^{K \times r_3}$ are the latent factor matrices respectively for the 1-mode, 2-mode, and 3-mode unfoldings of $\mathbf{X}$. $\mathbf{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ is the core tensor showing the interaction between the latent components from the three factor matrices. For example, the element $g_{pgr} \in \mathbf{G}$ shows the interaction between latent component $\boldsymbol{a}_p$[2], $\boldsymbol{b}_q$ and $\boldsymbol{c}_r$. Correspondingly, each element of $\mathbf{X}$ can be computed via

$$x_{ijk} \approx \sum_{p=1}^{r_1} \sum_{q=1}^{r_2} \sum_{r=1}^{r_3} g_{pqr} a_{ip} b_{jq} c_{kr}$$

where $i = 1, \ldots, I$, $j = 1, \ldots, J$, $k = 1, \ldots, K$.



$\mathbf{X} \in \mathbb{R}^{I \times J \times K}$
$\approx$
$A \in \mathbb{R}^{I \times r_1}$
$C \in \mathbb{R}^{K \times r_3}$
$\mathbf{G} \in \mathbb{R}^{r_1 \times r_2 \times r_3}$
$B \in \mathbb{R}^{J \times r_2}$

Figure 3.3: This picture is adopted from [52], which shows the Tucker decomposition of a third-order tensor $\mathbf{X} \approx \mathbf{G} \times_1 A \times_2 B \times_3 C$.

CP can be viewed as a special case of Tucker where the core tensor $\mathbf{G}$ is diagonal. The CP decomposition facilitates analysis on the latent components, such as applying CP to RDF graphs for faceted browsing [18]. Another two special cases of Tucker that will be used in this thesis are Tucker 1 and Tucker 2 models. A Tucker 1 decomposition of the third-order tensor $\mathbf{X}$ can be formulated as

$$\mathbf{X} \approx \mathbf{G} \times_3 C$$

where C acts as a weighting matrix on the tensor $\mathbf{G}$ such that the k-th frontal slice of $\mathbf{X}$ is a linear combination of the frontal slices of $\mathbf{G}$ with weights $\boldsymbol{c}_{k:}$. A Tucker 2 decomposition of the third-order tensor $\mathbf{X}$ can be formulated as

$$\mathbf{X} \approx \mathbf{G} \times_1 A \times_2 B \ .$$

An example of Tucker 2 decomposition will be shown in Section 3.1.2.3.

In contrast to CP, the Tucker decomposition is usually not unique due to the fact that the core tensor $\mathbf{G}$ can be modified without impact on the reconstruction as soon as

[2] $\boldsymbol{a}_p$ is the pth column of matrix A, likewise for $\boldsymbol{b}_q$ and $\boldsymbol{c}_q$

the corresponding inverse are applied to the factor matrices. However, different efforts have been made to improve the uniqueness by adding constrains on **G** to simplify the interactions between latent components.

To compute the Tucker decomposition, a High-Order Orthogonal Iteration (HOOI) algorithm [58] is used. Algorithm 2 illustrates the HOOI for a Tucker decomposition of a third-order tensor.

Algorithm 2 Tucker HOOI for Third-order Tensors [58]

```
Require: $\mathbf{X} \in \mathbb{R}^{I \times J \times K}$, $r_1$, $r_2$, $r_3$
 1: Initialize $A \in \mathbb{R}^{I \times r}$, $B \in \mathbb{R}^{J \times r}$, $C \in \mathbb{R}^{K \times r}$ respectively with the leading left singular
    vectors of the corresponding unfolding of $\mathbf{X}$
 2: repeat
 3:     $\mathbf{Y} \leftarrow \mathbf{X} \times_2 B^T \times_3 C^T$
 4:     $A \leftarrow r_1$ leading left singular vectors of $Y_{(1)}$
 5:     $\mathbf{Y} \leftarrow \mathbf{X} \times_1 A^T \times_3 C^T$
 6:     $B \leftarrow r_2$ leading left singular vectors of $Y_{(2)}$
 7:     $\mathbf{Y} \leftarrow \mathbf{X} \times_1 A^T \times_2 B^T$
 8:     $C \leftarrow r_3$ leading left singular vectors of $Y_{(3)}$
 9: until fitness ceases to improve or maximum iterations achieved
10: $\mathbf{G} \leftarrow \mathbf{X} \times_1 A^T \times_2 B^T \times_3 C^T$
11: return $A$, $B$, $C$, $\mathbf{G}$
```

3.1.2.3 The Relational Model RESCAL

RESCAL is a third-order tensor decomposition that can be seen as a special case of Tucker 2 decomposition. In RESCAL, entities including subjects and objects are modeled in the first two dimensions and a third dimension models different relation types. Instead of using three latent factor matrices in Tucker, RESCAL models all entities in one latent matrix. The core tensor of RESCAL shows interactions of latent components under different relation types. RESCAL was introduced by Nickel et al. in recent years [71, 72] and has been applied to the tasks of link prediction, entity resolution and link-based clustering. It has been proved that RESCAL has strong relational learning ability and scales well with large data sets such as the YAGO knowledge base.

Figure 3.4 shows the RESCAL decomposition of a third-order tensor $\mathbf{X} \in \mathbb{R}^{N \times N \times K}$, which can be formulated as

$$\mathbf{X}_k \approx AR_kA^T, \quad \text{for} \quad k = 1, \ldots, K$$

where N is the number of entities including both subjects and objects, A is a matrix containing the latent-component representation of the entities in the domain and R_k is an $r \times r$ matrix modelling the interactions of the latent components in the k-th relation.

An important aspect of RESCAL which distinguishes it from other tensor factorizations models is that entities have a unique latent representation via the matrix A. This enables a relational learning effect via the propagation of information over different relations and regardless of whether an entity occurs as a subject or object in a relationship.

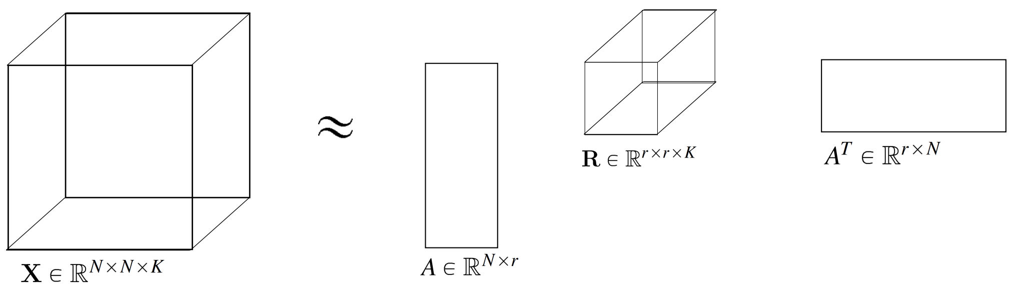

$\approx$
$\mathbf{X} \in \mathbb{R}^{N \times N \times K}$
$A \in \mathbb{R}^{N \times r}$
$\mathbf{R} \in \mathbb{R}^{r \times r \times K}$
$A^T \in \mathbb{R}^{r \times N}$

Figure 3.4: The RESCAL decomposition on a third-order tensor $\mathbf{X} \approx \mathbf{R} \times_1 A \times_2 A$.

RESCAL decomposition addresses to a regularized optimization problem.

$$\min_{A,\mathbf{R}} \quad \sum_{k=1}^{K} \|X_k - AR_kA^T\|_F^2 + \lambda_A\|A\|_F^2 + \lambda_R \sum_{k=1}^{K} \|R_k\|_F^2$$

To compute the RESCAL decomposition, an efficient and scalable RESCAL alternating least squares (RESCAL-ALS) algorithm [71, 72] was proposed which is illustrated in Algorithm 3. Note that $.*$ denotes elementwise product of matrices such that $(A. * B)_{ij} = a_{ij}b_{ij}$.

Algorithm 3 RESCAL-ALS [71, 72]

```
Require: $\mathbf{X} \in \mathbb{R}^{N\times N\times K}$, $r$, $\lambda_A$, $\lambda_R$
 1: Initialize $A \in \mathbb{R}^{N\times r}$ with the $r$ largest eigenvectors of $\sum_{k=1}^{K}(X_k + X_k^T)$
 2: repeat
 3:     $A \leftarrow \left[\sum_{k=1}^{K}\left(X_k A R_k^T + X_k^T A R_k\right)\right]\left[\sum_{k=1}^{K}\left(R_k A^T A R_k^T + R_k^T A^T A R_k + \lambda_A I\right)\right]^{-1}$
 4:     for $k = 1,\ldots,K$ do
 5:         $U, S, V \leftarrow \mathrm{SVD}(A)$
 6:         $\boldsymbol{s} \leftarrow \mathrm{diag}(S)$
 7:         $\boldsymbol{p} \leftarrow \boldsymbol{s} \otimes \boldsymbol{s}$
 8:         for $i = 1,\ldots,r^2$ do
 9:             $p_i \leftarrow \frac{p_i}{(p_i^2+\lambda_R)}$
10:         end for
11:         Reshape $\boldsymbol{p} \in \mathbb{R}^{r^2}$ to $\boldsymbol{P} \in \mathbb{R}^{r\times r}$
12:         $R_k \leftarrow V\left(P.*(U^T X_k U)\right)V^T$
13:     end for
14: until fitness ceases to improve or maximum iterations achieved
15: return $A$, $\mathbf{R}$
```

3.2 On the Rank of Tensors

3.2.1 The Computational Complexity of Tensor Decompositions

The computational complexity of a tensor decomposition is estimated by counting the number of elementary operations performed by the algorithm. As an elementary operation takes a fixed amount of time to perform, the total computational time taken is proportional to the number of elementary operations performed by the algorithm. For example, the computational complexity of a matrix multiplication AB (with $A \in \mathbb{R}^{I\times J}$, $B \in \mathbb{R}^{J\times K}$) is $O(IJK)$, since the elementary operation $a_{ij}b_{jk}$ is performed IJK times. Computing a tensor decomposition usually requires some basic matrix operations. Table 3.1 lists the runtime complexity of these basic matrix operations.

Based on Table 3.1, it can be easy to derive the computational complexity of CP, Tucker, and RESCAL decompositions of a third-order tensor $\mathbf{X} \in \mathbb{R}^{N\times N\times K}$ (usually $K << N$) with a predefined rank r.

For each iteration of the CP-ALS algorithm (Algorithm 1), updates to the three latent

Table 3.1: Computational Complexity of Basic Matrix Operations

Operation	Complexity
Matrix Multiplication AB, $A \in \mathbb{R}^{I\times J}$, $B \in \mathbb{R}^{J\times K}$	$O(IJK)$, or $O(nnz(A)K)^*$
Matrix Inversion A^{-1}, $A \in \mathbb{R}^{N\times N}$	$O(N^3)$ [65]
The Largest r Eigenvectors of $A \in \mathbb{R}^{N\times N}$	$O(Nr^2)$ [59]
Kronecker Product $A \otimes B$, $A \in \mathbb{R}^{I\times J}$, $B \in \mathbb{R}^{K\times L}$	$O(IJKL)$
Khatri-Rao Product $A \odot B$, $A \in \mathbb{R}^{I\times K}$, $B \in \mathbb{R}^{J\times K}$	$O(IJK)$

$*$ In case A is sparse. nnz stands for the number of nonzeros.

factor matrices are required. Take the update of latent matrix C as an example,

$$C \leftarrow X_{(3)}(B \odot A)(B^T B * A^T A)^\dagger$$

where A, $B \in \mathbb{R}^{N\times r}$ and $C \in \mathbb{R}^{K\times r}$. The computational complexity of the Khatri-Rao Product $B \odot A$ is $O(N^2 r)$. The matrix product $B^T B$ and $A^T A$ both require computational complexity of $O(Nr^2)$. Furthermore, since $B^T B * A^T A$ is a $r \times r$ matrix, the computational complexity of the product between $B^T B$ and $A^T A$ is $O(r^3)$, and the pseudo-inversion on top of the product requires another time complexity of $O(r^3)$. As the adjacency tensor for large scale data is usually sparse, the 1-mode unfolding $X_{(3)}$ is therefore sparse. The product bwtween $X_{(3)}$, $(B \odot A)$ and $(B^T B * A^T A)^\dagger$ requires a runtime of $O\left(nnz(\mathbf{X})\, r + Kr^2\right)$. So the overall time required to update C in one iteration is $O\left(N^2 r + 2Nr^2 + 2r^3 + nnz(\mathbf{X})\, r + Kr^2\right)$, which has a asymptotic time complexity of $O\left(nnz(\mathbf{X})\, r + N^2 r + Nr^2 + r^3\right)$. Similar analysis can be done on the updates of factor matrices B and C for CP-ALS. The conclusion is CP scales linearly with the number of nonzeros in the adjacency tensor, quadratic to the number of entities, cubic to the rank of tensor.

For using HOOI (Algorithm 2) to compute the Tucker decomposition, it is required to compute the r leading left singular vectors of $Y_{(1)}$ in line 4 of Algorithm 2, i.e. the r largest eigenvectors of $Y_{(1)}Y_{(1)}^T$ where $Y_{(1)} = X_{(1)}(C \otimes B)^T$. The matrix product $Y_{(1)}Y_{(1)}^T$ requires $O(N^2 r_2 r_3)$ elementary operations since $Y_{(1)}$ is a $N \times r_2 r_3$ matrix. To compute the r largest eigenvectors of a $N \times N$ matrix requires a computational complexity of $O(Nr^2)$ according to Table 3.1. Nevertheless, a memory problem immediately comes up as the line 3 of Algorithm 2 generates a dense tensor $\mathbf{Y}$ and $Y_{(1)}Y_{(1)}^T$ ends up with a dense matrix requiring memory complexity quadratic to the number of entities N.

Regarding to the runtime complexity of RESCAL-ALS (Algorithm 3), a detailed analysis is given in [69]. It is shown that RESCAL scales linearly with the data size, i.e. linearly with the number of entities, the number of relations, and the number of known facts. The computational complexity with regard to the number of latent components r is $O(r^3)$.

Considering the runtime complexity, RESCAL is more efficient than CP when the number of entities involved is large due to the fact that CP scales quadratic to N. Furthermore, the intermediate high demand on memory for Tucker limits its application on large scale data. RESCAL has rather low memory demand as the storage required by all the computations is linear to the data size and the size of latent factors.

3.2.2 The Conditions on the Rank of a Tensor

The analysis on the computational complexity of CP, Tucker, and RESCAL decompositions for third-order tensors shows, that the rank of a tensor (either the tensor rank or n-rank) is key for the runtime performance of tensor decompositions. The rank of a tensor is on the other hand controlling the complexity and the generalization ability of the model. A too low rank may not be sufficient to model the data, while a too high rank may cause overfitting problem.

Recall that when representing a relational graph $\mathcal{G}$ by an adjacency matrix X (Definition 2 in Section 1.2.3.2), it simultaneously records the information from $\mathcal{G}$ to a bipartite graph Γ. Section 2.1.2.1 has shown that the process of matrix factorizations (particularly SVD decomposition) is an action of grouping respectively the subject entities and relation type - object pairs based on their connectivity. After the factorization process, latent components are used to represent these subject groups or relation type - object groups (in case of only one relation type in the adjacency matrix, object groups).

Tensor factorizations can be analysed in a similar way where the relational graph $\mathcal{G}$ is represented by an adjacency tensor (Definition 3 in Section 1.2.3.2), accordingly represented by a bipartite graph Γ. The process of tensor decompositions compared to matrix factorizations differs in that tensor decompositions consider correlations in different unfolding matrices of a tensor. From Algorithm 1, 2 and 3, respectively for CP, Tucker and RESCAL decompositions, one can see that the latent representations are learnt based on the column correlations of the corresponding unfolding matrix. The 1-mode unfolding of a tensor is a matrix with each row for a subject entity, each column for a relation type - object pair. The 2-mode unfolding of a tensor has each row representing an object entity, each column representing a subject - relation type pair. The 3-mode unfolding of a tensor has rows for relation types, columns for subject - object pairs. After a tensor decomposition process, the latent representations for subjects, objects and relation types are respectively learnt from the correlations of relation type - object pairs, the correlations of subject - relation type pairs, and subject - object pairs. Entity-specific information as described in Section 1.2.2.1 can be used to exploit the first two kinds of correlations. However, the entity-pair information that is described in Section 1.2.2.2 and demonstrated in Figure 2.4 (c) is not yet used in the previous tensor models.

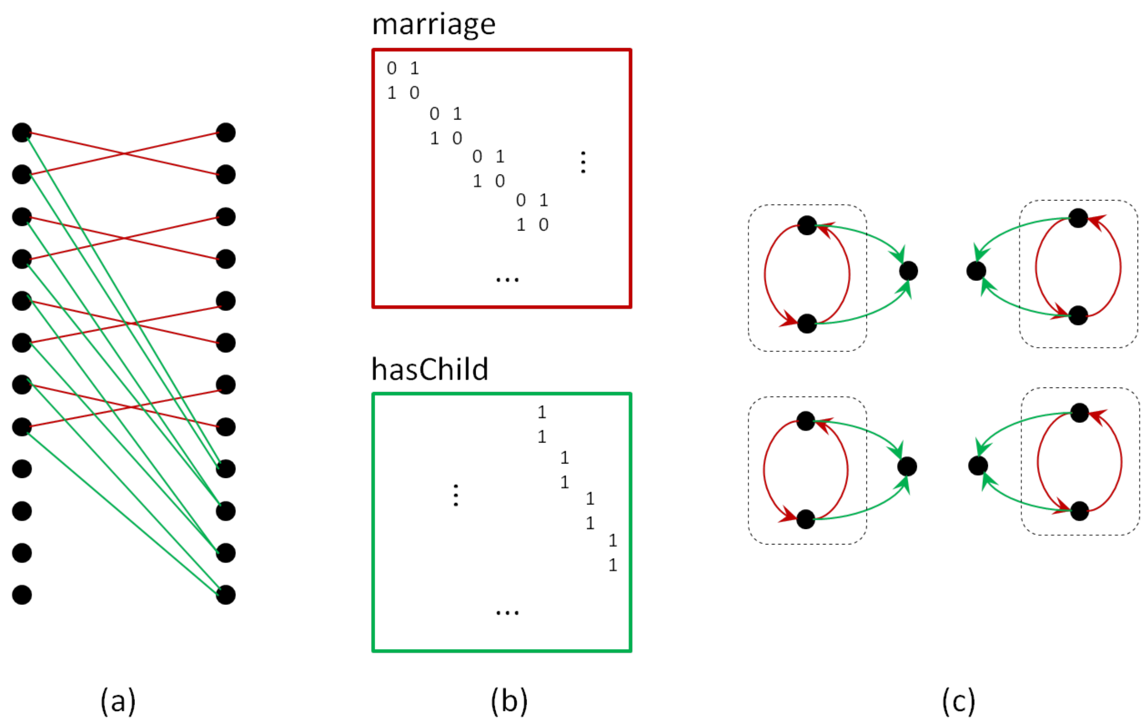


Figure 3.5: An example of 4 families, each has one child. In the three figures, the red color relates to the marriage relation; The green color relates to the hasChild relation. Figure (c) shows the relational graph with each dotted frame for a strongly connected component. A strongly connected component is a maximal subgraph of a relational graph where each vertex is reachable form any other vertex. Figure (b) shows the adjacency matrices for the marriage relation and the hasChild relation. Figure (a) shows the bipartite graph which corresponds to the relational graph (c).

To give a concrete example, Figure 3.5 (c) shows a relational graph of 4 families, each consisting of a pair of parents and their children. Two frontal slices representing the marriage relation and the hasChild relation are shown in Figure 3.5 (b). Figure 3.5 (a) gives the corresponding bipartite graph. It can be seen that the 4 pairs of marriages are isolated in the graph as one person is usually married to exactly one another person. It is hard to find any correlated columns or correlated rows in the unfolding matrices of the tensor that can be used to predict a marriage. Therefore it requires at least 4 latent components to recover the marriage matrix exactly. Technically, the marriage matrix can be decomposed as:

$$X = I \otimes \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

where I is an identity matrix of size 4×4.

However, the marriage relation can be easily predicted from a pattern expressed as:

$$\text{marriage}(var1, var2) \leftarrow \text{hasChild}(var1, var3) \wedge \text{hasChild}(var2, var3)$$

which is essentially the Common Neighbors (Section 1.2.2.2).

Another high rank requirement comes from the relation of myself (not plotted in the figure), which has an identity matrix as its adjacency matrix. The myself relation could not be learnt from the other known facts but it requires a full rank to recover any unfolding matrix that contains it, therefore, requires a full rank for the tensor decomposition.

From the theoretical considerations, a bound was proved to tell under which conditions the tensor decompositions require a high rank or a low rank[3].

Theorem 1. *The tensor rank* $\text{rank}(\mathbf{X})$ *and multi-linear rank* $\text{n-rank}(\mathbf{X}) = (r_1, r_2, r_3)$ *of any adjacency tensor* $\mathbf{X} \in \{0,1\}^{N\times N\times K}$ *over* K *relations with adjacency matrices* $\{X_k\}_{k=1}^{K}$ *is bounded by*

$$\sum_{k=1}^{K} \text{dp}(\Gamma_k) \geqslant \theta \geqslant \max_k \text{rank}(X_k) \geqslant \max_k \text{scc}_+(X_k)$$

where θ *is any of the quantities* $\text{rank}(X)$, r_1, *or* r_2. Γ_k *is the bipartite graph of*

[3]The theoretical prove of Theorem 1, Lemma 2 and Lemma 3 in [70] was done by the coauthor Maximilian Nickel. It is straightforward to derive Theorem 1 in this thesis by combing the three theoretical results.

relation X_k. $\mathrm{dp}(\Gamma_k)$ *is the biclique cover number of* Γ_k*, where biclique cover number is the minimum number of complete bipartite subgraphs needed to cover all edges in the bipartite graph.* $\mathrm{scc}_+(X_k)$ *is the number of strongly connected components in the bipartite graph* Γ_k *with a constraint that each component contains more than one element.*

As shown in Figure 3.5 (c), the number of strongly connected components in the marriage relation is 4, correspondingly $scc_+ = 0$ for the hasChild relation, and $scc_+ = 0$ for the myself relation. Thus the maximum number of strongly connected components in all the relations occurs in the marriage relation, i.e. $\max_k \mathrm{scc}_+(X_k) = 4$. Theorem 1 tells that the rank required to recover the adjacency tensor $\mathbf{X}$ for this example is at least 4. Equivalently, for m marriages, a factorization model would at least require m latent components to recover the adjacency tensor. As the computational complexity of tensor decompositions is in cubic of the rank, large m would have rather high computation demand which render the application of factorization approaches on this kind of relations.

To summarize, tensor decompositions are very efficient if the related bipartite graph can be partitioned into a few fully connected groups. But they are inefficient to handle relations that are locally connected, such as the marriage relation. Some observable patterns that are path-based or neighborhood-based (e.g. Common Neighbors) can easily predict relations via exploiting the entity-pair information.

3.3 Reducing the Factorization Rank by including Observable Patterns

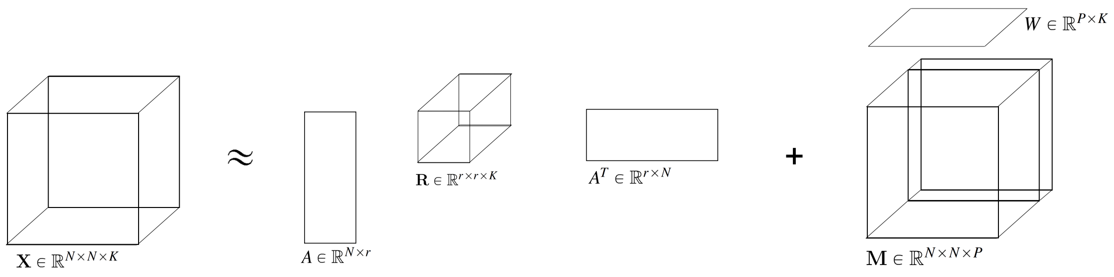
Motivated by the previous discussion, an Additive Relational Effects (ARE) tensor model will be presented in this section, which derives the strong relational learning effects from the highly scalable tensor decomposition approach RESCAL, and a Tucker 1 tensor which includes observable patterns as instantiated relations. An efficient least squares approach will also be proposed to compute the combined model ARE. Experiments on several benchmark data sets show that ARE starts to have good predictive performance at a very low rank, and it outperforms many state-of-the-art relational learning algorithms.

3.3.1 An Additive Relational Effects Model

To include the knowledge of observable patterns, an additive term that consists of the predictions of observable pattern methods is added to the RESCAL model. Let $\mathbf{X} \in \{0,1\}^{N\times N\times K}$ be a third-order adjacency tensor and $\mathbf{M} \in \mathbb{R}^{N\times N\times P}$ be a constant third-order tensor that contains the predictions of an arbitrary number of relational learning methods. Figure 3.6 shows the proposed Additive Relational Effects model (ARE), which can be formulated as

$$\mathbf{X} \approx \mathbf{R} \times_1 A \times_2 A + \mathbf{M} \times_3 W \tag{3.1}$$

where $A \in \mathbb{R}^{N\times r}$, $R \in \mathbb{R}^{r\times r\times K}$ and $W \in \mathbb{R}^{K\times P}$.



$W \in \mathbb{R}^{P\times K}$
$\approx$
$+$
$\mathbf{R} \in \mathbb{R}^{r\times r\times K}$
$A^T \in \mathbb{R}^{r\times N}$
$\mathbf{X} \in \mathbb{R}^{N\times N\times K}$
$A \in \mathbb{R}^{N\times r}$
$\mathbf{M} \in \mathbb{R}^{N\times N\times P}$

Figure 3.6: The ARE decomposition for a third-order tensor $\mathbf{X} \approx \mathbf{R} \times_1 A \times_2 A + \mathbf{M} \times_3 W$.

3.3.1.1 The RESCAL Part of ARE

The first term of equation (3.1) corresponds to the RESCAL model which has the following interpretation: The matrix $A \in \mathbb{R}^{N\times r}$ holds the latent variable representations of the entities, while each frontal slice R_k of $\mathbf{R}$ is an $r \times r$ matrix that models the interactions of the latent components for the k-th relation. The variable r denotes the number of latent components of the factorization.

As A encodes the information of each entity acting as a subject or as an object in different relationships, each column of A is a latent component representing a group of entities that share similar subject entities or similar object entities. Row n of A represents entity n, denoted as $\boldsymbol{a}_n$, $n = 1,\ldots,N$. This unique representation for entities in the domain enables information propagation in the relational graph. Take entities of employee i, employee i_0, department j, team t, team t_0 and relation type k (department membership) as an example. The confidence value of a team t member employee i belongs to department j can be calculated via $x_{ijk} = \boldsymbol{a}_i^T R_k \boldsymbol{a}_j$. Assume it is known that employee i_0 from team t_0 is in department j, i.e. $\boldsymbol{a}_{i_0}^T R_k \boldsymbol{a}_j = 1$ for $i_0 \neq i$. As entity t shares a common object entity j with t_0, the latent representation for entity t is similar to the one for entity t_0. Consequently, entity i will be similar to entity i_0 due to sharing the similar object entities (t and t_0), namely $\boldsymbol{a}_i \approx \boldsymbol{a}_{i_0}$. Therefore, it can be concluded that $x_{ijk} = \boldsymbol{a}_i^T R_k \boldsymbol{a}_j \approx \boldsymbol{a}_{i_0} R_k \boldsymbol{a}_j = 1$, i.e., employee i is most likely also in department j. Explicitly, this information propagation can be formulated as

$$\boldsymbol{a}_i^T R_k \boldsymbol{a}_j \quad \leftarrow \quad \boldsymbol{a}_i^T R_{k_1} \boldsymbol{a}_t \wedge \boldsymbol{a}_t^T R_{k_2} \boldsymbol{a}_j \wedge \boldsymbol{a}_{i_0}^T R_{k_1} \boldsymbol{a}_{t_0} \wedge \boldsymbol{a}_{i_0}^T R_k \boldsymbol{a}_j \quad \forall\, i,\, j,\, t\,. \tag{3.2}$$

where k_1 represents the relation type of team membership, k_2 represents the relation type teamOf.

3.3.1.2 The Observable Patterns in ARE

The second term of equation (3.1) is a Tucker 1 model. The model $\mathbf{M} \times_3 W$ shows how informative the observable patterns are. The construction of the tensor $\mathbf{M}$ is as following: Let $\mathcal{F} = \{f_p\}_{p=1}^P$ be a set of given real-valued functions $f_p : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ which assign scores to each pair of entities in the entities set $\mathcal{V}$. Examples of such score functions include link prediction heuristics such as Common Neighbours, Katz Centrality, the Jaccard coefficient, Horn clauses, and more can be found in Section 1.2.2.2. These scores can be interpreted as confidence values of a relationship being true between two entities. These real-valued predictions of P score functions are collected in the tensor $\mathbf{M} \in \mathbb{R}^{N \times N \times P}$ by setting $m_{ijp} = f_p(v_i, v_j)$. In the factorization $\mathbf{M}$ acts as prior knowledge that predicts the existence of relationships and the term $\mathbf{M} \times_3 W$ can be interpreted as learning a set of weights w_{kp}, which indicate how much the p-th piece of information in $\mathbf{M}$ correlates with the k-th relation in $\mathbf{X}$.

In the RESCAL part, information propagation is based on the connectivity of the associated relational graph. Equation 3.2 tells RESCAL can easily learn that all employees from a team of a department also belong to that department. It requires m connected sub relational graphs and consequently m latent components to model m different departments, with each latent component for one department. However, prior knowledge formulated as a transitive relation

$$\boldsymbol{a}_i^T R_k \boldsymbol{a}_j \quad \leftarrow \quad \boldsymbol{a}_i^T R_{k_1} \boldsymbol{a}_t \wedge \boldsymbol{a}_t^T R_{k_2} \boldsymbol{a}_j \quad \forall\, i,\; j,\; t$$

can be used to cover all the departments. This observable pattern can be integrated into the tensor $\mathbf{M}$ via a simple matrix multiplication $X_{k_1} X_{k_2}^T$. The weighting matrix W then weighs the importance of the product $X_{k_1} X_{k_2}^T$ to the relation k.

3.3.1.3 The Additive Effects from RESCAL and the Observable Pattens

The main idea of ARE is as following: The term $\mathbf{R} \times_1 A \times_2 A$ is equivalent to the RESCAL model and provides a very efficient approach to learn from latent patterns on relational data. The tensor $\mathbf{M}$ on the other hand is not factorized, such that it can hold information that is not efficient to predict via latent variable methods. As it is not clear a priori which score functions are good predictors for which relations, the term $\mathbf{M} \times_3 W$ learns how informative each score function is for each known relation. By integrating both terms in an additive model, the term $\mathbf{M} \times_3 W$ can potentially reduce the required rank for the RESCAL term by explaining links that spread in many latent groups but can be expressed by observable patterns. For instance, by including a copy of the observed adjacency tensor $\mathbf{X}$ in $\mathbf{M}$ (or some selected frontal slices X_k), the term $\mathbf{M} \times_3 W$ can easily model common multirelational patterns where two different relations are correlated via $x_{ijk} = \sum_{p \neq k} w_{kp} x_{ijp}$. Since w_{kp} is allowed to be negative, anticorrelations can also be modeled efficiently. ARE can be regarded to be similar in spirit to the model of [55], which extends SVD with additive terms to include nearest neighbor information in an uni-relational recommendation setting.

After learning the latent representation of the RESCAL part and the weights for the tensor $\mathbf{M}$, the confidence value of a relationship (i, j, k) can then be computed via a linear combination of the interactions between latent component i and latent component j under relation k, and the interactions of the entity i and entity j in all the observable patterns.

$$x_{ijk} \approx \boldsymbol{a}_i^T R_k \boldsymbol{a}_j + \sum_{p=1}^{P} w_{kp} m_{ijp}$$

To summarize, the ARE decomposition addresses to a regularized optimization problem.

$$\min_{A,\mathbf{R},W} \sum_{k=1}^{K} \left\| X_k - \left(A R_k A^T + \sum_{p=1}^{P} w_{kp} M_p \right) \right\|_F^2 + \lambda_A \|A\|_F^2 + \lambda_R \sum_{k=1}^{K} \|R_k\|_F^2 + \lambda_W \|W\|_F^2.$$

3.3.2 Updates for W

The Equation 3.1 can be rewritten in the 3-mode unfolding

$$E_{(3)} \approx W M_{(3)}$$

where $\mathbf{E} = (\mathbf{X} - \mathbf{R} \times_1 A \times_2 A)$. It is straightforward to derive the updates for W via regularized least squares

$$W \leftarrow (M_{(3)} M_{(3)}^T + \lambda_W I)^{-1} M_{(3)} E_{(3)}^T$$

where I denotes the identity matrix. However, this computation contains an intermediate result of a dense $N \times N \times K$ tensor $\mathbf{R} \times_1 A \times_2 A$. It requires a runtime complexity of $O(N^2 K r + \mathrm{nnz}(M) N)$ for the computation which is computational costly. Furthermore, the dense tensor has a memory complexity of $N \times N \times K$ which is not feasible for large data sets as N can be large. This scalability problem can be overcome by considering the special structure of the problem. Consider

$$\begin{aligned}
(R \times_1 A \times_2 A)_{(3)} M_{(3)}^T &= R_{(3)} (A \otimes A)^T M_{(3)}^T \\
&= R_{(3)} \left(M_{(3)} (A \otimes A) \right)^T \\
&= R_{(3)} \left(\mathbf{M} \times_1 A^T \times_2 A^T \right)_{(3)}^T
\end{aligned}$$

updates for W can be alternatively computed via

$$W^T \leftarrow \left[X_{(3)} M_{(3)}^T - R_{(3)} \left(\mathbf{M} \times_1 A^T \times_2 A^T \right)_{(3)}^T \right] Z \tag{3.3}$$

where $Z = (M_{(3)} M_{(3)}^T + \lambda_W I)^{-1}$. In Equation 3.3, the dense tensor $\mathbf{R} \times_1 A \times_2 A$ has never to be computed explicitly. The runtime complexity of $R_{(3)} (\mathbf{M} \times_1 A^T \times_2 A^T)_{(3)}$ is only $O(NKr^3 + \mathrm{nnz}(M) r)$. Therefore the computational complexity of Equation 3.3 with regard to the parameters N, K, and r is reduced to $O(NKr^3)$. Moreover, the terms $X_{(3)} M_{(3)}^T$ and Z are constant and have only to be computed once at the beginning of the algorithm. $X_{(3)} M_{(3)}^T$ and $M_{(3)} M_{(3)}^T$ are the products of sparse matrices and can be computed efficiently.

3.3.3 Updates for A and R

Following the high-scalability of the alternating least squares algorithm to compute the RESCAL factorization (Algorithm 3), a similar optimization scheme is used to compute the RESCAL part of ARE.

The update rules for A and $\mathbf{R}$ can be directly derived from RESCAL-ALS by setting

$$\mathbf{E} = \mathbf{X} - \mathbf{M} \times_3 W$$

and computing a RESCAL factorization of $\mathbf{E}$.

The updates for A can therefore be computed by:

$$A \leftarrow \left(\sum_{k=1}^{K} E_k A R_k^T + E_k^T A R_k\right)\left(\sum_{k=1}^{K} R_k A^T A R_k^T + R_k^T A^T A R_k + \lambda_A I\right)^{-1}$$

where $E_k = X_k - \mathbf{M} \times_3 \boldsymbol{w}_{k:}$ and $\boldsymbol{w}_{k:}$ denotes the k-th row of W.

The updates of $\mathbf{R}$ can be computed in the following way: Let $A = USV^T$ be the SVD of A, where the diagonal vector $\boldsymbol{s}$ contains the singular values of A. Furthermore, let $\boldsymbol{p} = \boldsymbol{s} \otimes \boldsymbol{s}$ and regularize $p_i \in \boldsymbol{p}$ via $p_i = \frac{p_i}{p_i^2 + \lambda_R}$. Reshape $\boldsymbol{p}$ to a matrix $P \in \mathbb{R}^{r \times r}$. An update of R_k can then be computed via

$$R_k \leftarrow V\left(P.* U^T B_k U\right) V^T$$

where $B_k = X_k - \mathbf{M} \times_3 \boldsymbol{w}_{k:}$.

A more detailed description of ARE-ALS is given in Algorithm 4. It can be seen that ARE-ALS retains the good scalability of RESCAL-ALS.

3.3.4 Experiments

Various multirelational data sets are used to evaluate the ARE model. A focus is on the generalization ability of ARE decomposition with respect to its rank. To evaluate

Algorithm 4 ARE-ALS

```
Require: $\mathbf{X} \in \mathbb{R}^{N \times N \times K}$, $\mathbf{M} \in \mathbb{R}^{N \times N \times P}$, $r$, $\lambda_A$, $\lambda_R$, $\lambda_W$
 1: $D \in \mathbb{R}^{K \times P} \leftarrow X_{(3)} M_{(3)}^T$
 2: $Z \in \mathbb{R}^{P \times P} \leftarrow (M_{(3)} M_{(3)}^T + \lambda_W I)$
 3: Initialize $A \in \mathbb{R}^{N \times r}$ with the eigenvectors of $\sum_{k=1}^{K} (X_k + X_k^T)$
 4: repeat
 5:   $W^T \leftarrow \left[ D - R_{(3)} \left( \mathbf{M} \times_1 A^T \times_2 A^T \right)_{(3)}^T \right] Z$
 6:   $A \leftarrow \left( \sum_{k=1}^{K} E_k A R_k^T + E_k^T A R_k \right) \left( \sum_{k=1}^{K} R_k A^T A R_k^T + R_k^T A^T A R_k + \lambda_A I \right)^{-1}$ ,
      where $E_k = X_k - M \times_3 \boldsymbol{w}_{k:}$
 7:   for $k = 1, \ldots, K$ do
 8:     $U, S, V \leftarrow \mathrm{SVD}(A)$
 9:     $\boldsymbol{s} \leftarrow \mathrm{diag}(S)$
10:     $\boldsymbol{p} \leftarrow \boldsymbol{s} \otimes \boldsymbol{s}$
11:     for $i = 1, \ldots, r^2$ do
12:       $p_i \leftarrow \frac{p_i}{(p_i^2 + \lambda_R)}$
13:     end for
14:     Reshape $\boldsymbol{p} \in \mathbb{R}^{r^2}$ to $P \in \mathbb{R}^{r \times r}$
15:     $R_k \leftarrow V \left( P. * \left( U^T B_k U \right) \right) V^T$,    where $B_k = X_k - \mathbf{M} \times_3 \boldsymbol{w}_{k:}$
16:   end for
17: until fitness ceases to improve or maximum iterations achieved
18: return $A$, $\mathbf{R}$, $W$
```

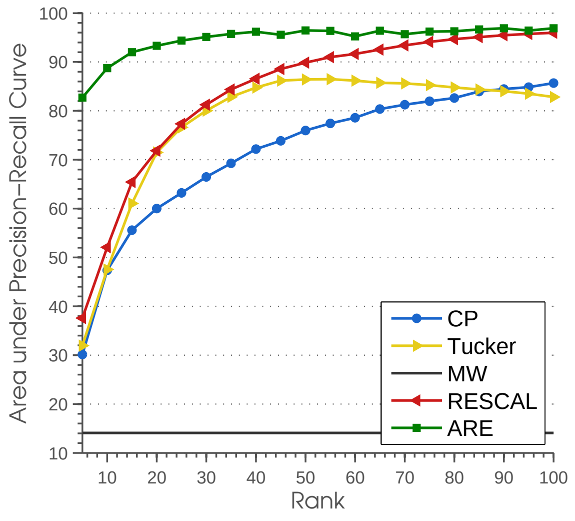
the performance of the additive approach of Equation 3.1, the well-established tensor factorizations CP and Tucker are included in the evaluation as well as the RESCAL factorization and the non-latent model $\mathbf{X} \approx \mathbf{M} \times_3 W$ (labelled as $\mathbf{M}W$).

For all Experiments evaluated by the area under the precision-recall curve (AUC-PR), k-fold cross validation was performed for each model, following the same experimental protocol as in [48, 50, 94, 71] and as described in Section 2.3.5. For experiments evaluated by nDCG@all, k-fold cross validation was performed for each model, following the same experimental protocol as described in Section 2.4.1. To make a fair comparison, experiments conducted on the same data set used the same data splits for different models.

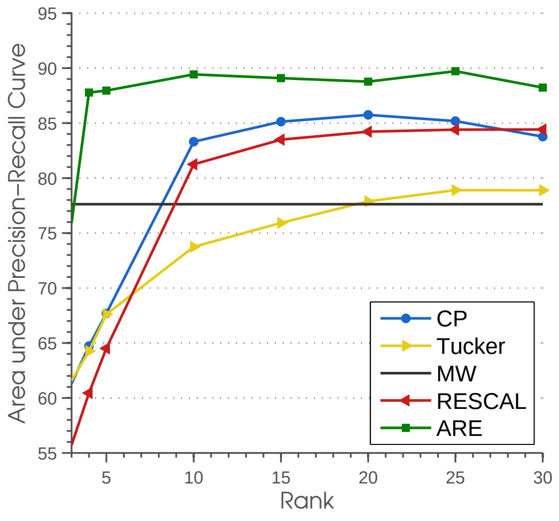
3.3.4.1 Social Evolution

The first set of experiments were performed on the Social Evolution data set[4] [64] which consists of multiple relations of persons living in a undergraduate dormitory. From the relational data, a $84 \times 84 \times 5$ adjacency tensor was constructed where the first two dimensions correspond to persons and the third mode represents the relationships between these persons such as friendship (CloseFriend), social media interaction (BlogLivejournalTwitter and FacebookAllTaggedPhotos), political discussion (PoliticalDiscussant), and social interaction (SocializeTwicePerWeek). Link prediction for each relation was performed via 5-fold cross validation. The tensor $\mathbf{M} \times_3 W$ consisted only of a copy of the observed tensor $\mathbf{X}$. Including $\mathbf{X}$ in $\mathbf{M}$ allows ARE to efficiently exploit that if a social relation holds for a pair of persons, it is likely that they also share other social activities. It can be seen from the results in Figure 3.7 $(b - f)$ that ARE achieves better performance than all competing approaches and achieves excellent performance already at a very low rank. On this data set, the predictive power of the observable pattern model MW is quite strong and it is not constrained by any rank requirement. As the observable patterns MW can explain most of the known facts in the tensor $\mathbf{X}$ via exploiting the entity-pair correlations, the RESCAL part of ARE therefore only needs a very low rank to model the facts that have not been covered by the heuristics.

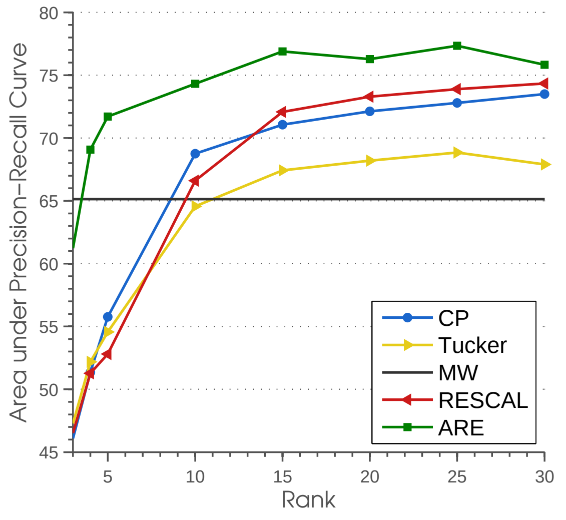
[4] `http://realitycommons.media.mit.edu/socialevolution.html`



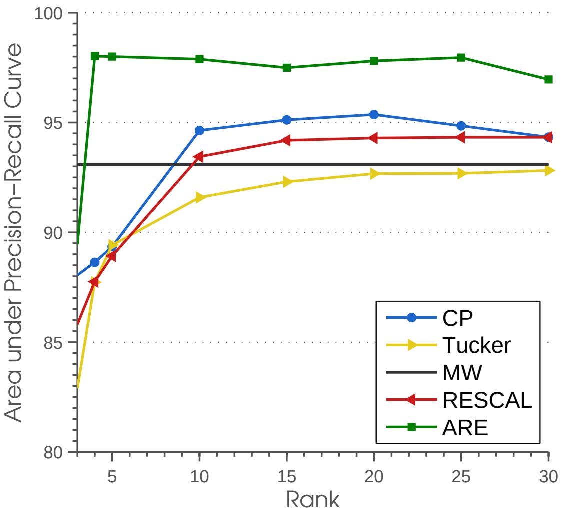
(a) Kinship



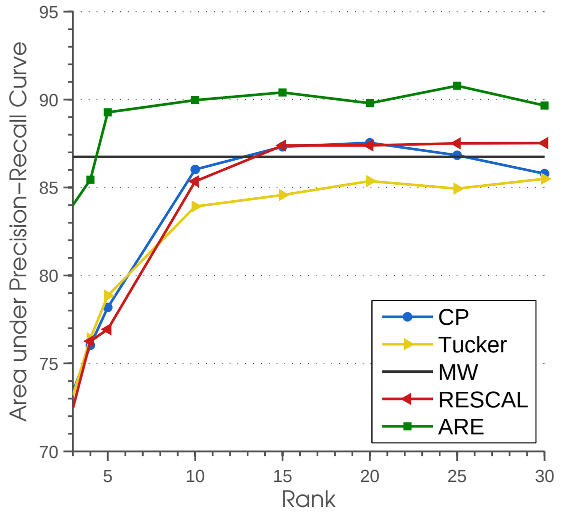
(b) PoliticalDiscussant



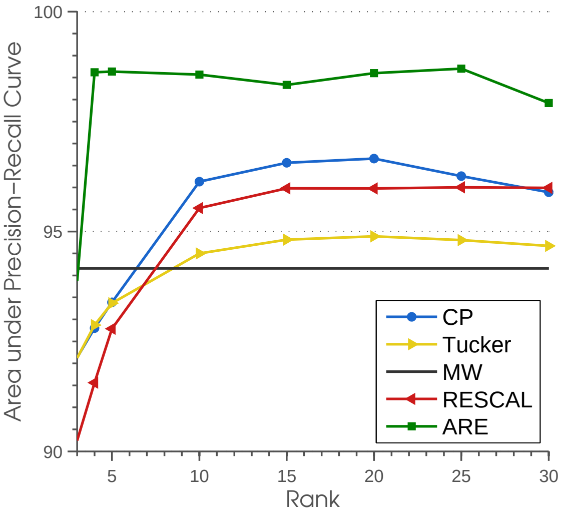
(c) CloseFriend



(d) BlogLiveJournalTwitter



(e) SocializeTwicePerWeek



(f) FacebookAllTaggedPhotos

Figure 3.7: Evaluation results for AUC-PR on the Kinship (3.7a) and Social Evolution data sets (3.7b-3.7f)

3.3.4.2 Kinship

The Kinship data set[5] is a popular data set for evaluating relational learning algorithms. It describes the complex kinship relations in the Australian Alyawarra tribe, originally collected by [13] in terms of 26 kinship relations between 104 persons. The task in the experiment was to predict unknown kinship relations. A 10-fold cross validation was conducted. Beside CP, RESCAL and DEDICOM [29], the experimental results of ARE was compared to the published results of IRM [50], MRC [50], BCTF [94], SME [26], and the latent factor model (LFM) of [40]. Table 3.2 shows the improvement of ARE over these methods. Figure 3.7a shows the predictive performance compared to the rank of multiple factorization methods. As the Kinship data set contains many high rank relations such as relation type myself with full rank 104, all the factorization approaches require almost full rank to model the data. Nevertheless, it can be seen that ARE outperforms all other methods at lower ranks significantly. Moreover, starting from rank 40 ARE already gives comparable results to the best results in Table 3.2. As in the previous experiments, $\mathbf{M}$ consists only of copies of observed frontal slices X_k. Using these frontal slices ARE can model the fact that the relations in the kinships data set are mutually exclusive, by setting $w_{ii} > 0$ and $w_{ij} < 0$ for all $i \neq j$ which explains the large improvement over RESCAL.

3.3.4.3 Link Prediction on Semantic Web Data

The SWRC ontology[6] [93] models a research group in terms of people, publications, projects and research interests. The task in this set of experiments was to predict the affiliation relation, i.e. to map persons to research groups. The experimental setting follows the one in [62]: From the raw data, a $12058 \times 12058 \times 85$ tensor was constructed by considering all directly connected entities of persons and research groups in the data. In total, 168 persons and 5 research groups are considered in the evaluation data. The tensor $\mathbf{M}$, consisted again of a copy of $\mathbf{X}$. Additionally, the Common Neighbors pattern of the form $X_k X_k$ and $X_k^T X_k^T$ $(k = 1, \ldots, K)$ were included. These patterns were included to explain patterns such that people who share the same research interest are likely in the same affiliation, that a person is related to a department if the person belongs to a research group of that department, etc.. Furthermore, a

[5]`http://alchemy.cs.washington.edu/data/kinships/`
[6]`http://ontoware.org/swrc/`

sparsity penalty was imposed on W and inactive observed patterns were pruned away during iterations. Table 3.3 shows that ARE with a small rank of 15 has superior performance compared to the three state-of-the-art link prediction algorithms (SVD, Common Subtrees [62] and RESCAL with rank 45) in the Semantic Web area.

Table 3.2: Evaluation Results on the Kinship Data Set

Approach	Aera under PR Curve	Rank
IRM	66	-
MRC	86	-
BCTF	90	-
DEDICOM	69	-
CP	94	170
SME	90.7	-
Latent Factor Model	94.6	(50,50,500)
RESCAL	96	100
ARE	**96.9**	**90**

Table 3.3: Evaluation Results on the SWRC Data Set

Approach	SVD	Subtrees	RESCAL	MW	ARE
nDCG@all	0.8	0.95	0.96	0.59	**0.99**

3.3.4.4 Runtime Performance

As discussed in Section 3.2, rank is one of the most critical parameters of factorization methods for their computational complexity as well as for their generalization abilities. To evaluate the trade-off between runtime and predictive performance, the nDCG values of RESCAL and ARE were reported after each iteration of the respective ALS algorithms on the Cora citation database[7]. In these experiments, the classification variant of Cora was used in which all publications are organized in a hierarchy of topics with two to three levels and 68 leaves. The relational data consists of information about paper citations, paper and topics from which a tensor of size $28073 \times 28073 \times 3$ was constructed. The tensor consisted of a copy of $\mathbf{X}$ and the

[7]`https://people.cs.umass.edu/~mccallum/data.html`

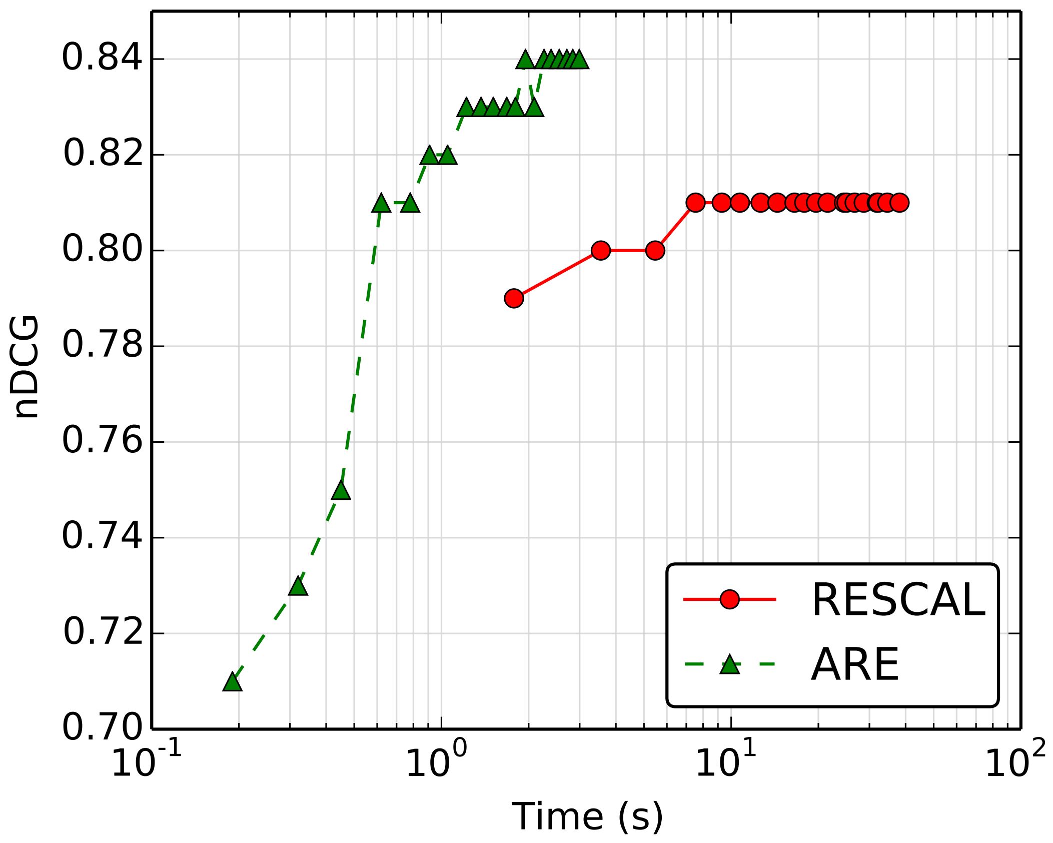


Figure 3.8: Runtime Performance on the Cora Data Set for RESCAL and ARE over iterations. The 20 markers respectively on the two curves show how the predictive performance increases after each iteration.

Common Neighbors patterns X_iX_j and $X_i^TX_j^T$ to model patterns such as that a cited paper shares the same topic, and that a cited paper shares the same author. The task of the experiments was to predict the leaf topic of papers by 5-fold cross-validation. The optimal rank 220 for RESCAL was determined out of the range $[10,300]$ via parameter selection, while ARE used a significantly smaller rank 20. Figure 3.8 shows the runtime of RESCAL and ARE compared to their predictive performance. It is evident that after a few iteration ARE outperforms RESCAL although the rank of the factorization is decreased by an order of magnitude. Moreover, ARE surpasses the best prediction results of RESCAL in terms of total runtime even before the first iteration of RESCAL-ALS ceased.

Chapter 4

Conclusions

4.1 Summary

Relational data is an efficient way to represent knowledge in form of subject - relation type - object triples. As the relational data available from different domains increases rapidly both in amount and complexity, scalable and efficient learning algorithms are needed for the development of predictive models. This thesis considers the problem of integrating prior knowledge into factorization approaches for learning from relational data. The goal is to retain the good scalability from factorization approaches and to improve the modeling power by including prior knowledge. Particularly, prior knowledge is the comprehensive information (either in form of unstructured texts or structured patterns) to describe the entities involved or to describe the relationships between entities.

A survey on the types of information that can be exploited from the relational data is given in Chapter 1. The two main contributions in the thesis, presented in Chapter 2 and Chapter 3, demonstrate how prior knowledge can be included into matrix factorizations and tensor decompositions respectively.

Chapter 2 proposes a novel and general framework M_{add} to include entity-specific information and entity-pair information in an additive model based on matrix factorizations. Triples representing entity-specific information and entity-pair information are added in form of a set of matrices describing the various instantiated relations between entities. Particularly, structure information that can be retrieved directly from the data, derived via aggregations, or derived via join operations, is considered. Unstructured information from text fields or textual documents describing the involved entities (e.g., from the entities' Wikipedia pages) is also considered. Additionally, a kernel solution with sensible kernels is discussed. Efficient learning of the additive model is achieved using an alternating least squares approach exploiting sparse matrix algebra and low-rank approximations. Experiments for relation prediction on various Semantic Web data sets, and for recommendation systems show that including prior knowledge in factorization methods can improve the predictive power.

Chapter 3 proposed another additive model ARE which stands for Additive Relational Effects, to combine the state-of-the-art tensor decomposition method RESCAL, and a Tucker 1 model. Essentially, factorization approaches are sufficient to exploit entity-specific information via the correlations in the unfolding matrices of a tensor representation of the data. RESCAL has strong relational learning ability compared to other factorization approaches, as it allows information propagation along the relational network. However, factorization approaches including RESCAL can only exploit the information based on the connectivity of the relational network. The entity-pair information which includes path-based or neighborhood-based patterns is not yet well employed in previous factorization approaches. The Tucker 1 part of ARE is responsible for including observable patterns. As RESCAL has been proved to scale well with large knowledge bases such as YAGO, the RESCAL part of ARE retains this good scalability. ARE therefore combines the strength of RESCAL, and Tucker 1, where the latter contains the prior knowledge in form of observable patterns on entity-pairs. It is shown analytically in Chapter 3 that including the entity-pair information in tensor models can reduce the rank requirement of tensor decompositions. As the rank of a tensor is an important factor for computational complexity, reducing the rank consequently reduces the computational time. Therefore, ARE is a predictive and scalable algorithm. Evaluations on various data sets show that ARE outperforms many state-of-the-art relational learning algorithms in both runtime and predictive performance.

4.2 Future Work Directions

In this thesis, the entity-pair information is included to the models via calculating the predefined score functions on each entity pair. Chapter 1 gives a survey on some path-based or neighborhood-based score functions, such as Common Neighbors. A global optimization procedure then learns the weights for these score functions. However, some heuristics for the score functions may result in a large number of possible relations. For example, applying the transitive rule to all the possible relation-pairs will result in a large number of new patterns. The solution from this thesis is to prune the inactive patterns during iterations by imposing sparsity on the weights. However, instead of using predefined measures on entity-pair information, one can also try to include some structure learning results as entity-pair features, such as the path features from the Path Ranking Algorithm (PRA) [57].

Moreover, the sparsity and the size of the matrices involved in the model computations are crucial for good scalability. One heuristic is to include type constraints when combining the predictions of different score functions for a known relation, to obtain sparse predicted relations. Another heuristic is to include type constraints to the latent matrices when calculating the interactions between different latent components for a specific relation. Particularly for the tensor decomposition model ARE proposed in Chapter 2, considering type constraints in each step in Algorithm 4 reduces the elementary operations needed for the calculations and therefore further speeds up the algorithm.

4.3 Application Domains

This thesis proposed two main algorithms for combining comprehensive structured or unstructured information to improve the predictive performance on multi-relational data. Evaluations on various publicly available data sets show that the proposed algorithm ARE outperforms state-of-the-art machine learning algorithms. For several data sets, ARE gives better predictions than the so far best published results. ARE can be applied in many ways:

ARE can be applied on knowledge base constructions: A recent work from Google research proposed the Knowledge Vault (KV) [14]. It constructs a knowledge base containing 1.6 billion facts collected from the internet. KV extracts information from both the Web content (text-based information such as the text documents, HTML trees, HTML tables, and Human annotated pages) and the prior knowledge (path-based patterns) derived from existing knowledge repositories. The algorithm behind KV to automatically construct facts is a combination of the Path Ranking Algorithm and a neural network model similar to RESCAL. The model ARE from this thesis is very similar to the KV algorithm in that the Tucker 1 part of ARE incorporates prior knowledge (both path-based patterns and neighborhood-based patterns), and the RESCAL part of ARE has collective learning ability. In comparizon to the KV algorithm, ARE has two advantages: first, the combination is globally optimized in ARE and, second, in ARE the rank of the factorization is decreased which is not the case for the rank of the neural network model.

ARE can be used for predictive maintenance problems: Take each component in an equipment as an entity, the interactions between components as different relations between the entities. Predictive maintenance has the goal of predicting the future trend of the equipment's condition, such as predicting failures. By analysing the data about the status and communications of different components, machine learning algorithms can predict those statistically repeated behaviors. Particularly ARE can include the expert knowledge for making predictions. For example, it is known that overweight on a conveyor can cause transportation errors. This piece of knowledge can be formulated as entity-pair information for the conveyor-equipment pair, with the weight on the conveyor as feature values that is correlated to the failure caused by the conveyor.

ARE can be used for predictive data analytics solutions: The Industry 4.0 is the German next-generation of factory automation based on Internet of Things (IoT) technologies. In the IoT context, huge amounts of sensors can do an accurate near-real time sensing and monitoring of physical devices. IoT as relational data considers each physical device as an entity, and considers the cooperation between devices as relations. Therefore ARE can detect the correlated events on different devices and can tackle tasks such as root cause analysis.

Besides, ARE can be used for recommendation systems, decision support and any predictive problems on data that can be modeled as relationships between entities.

Bibliography

[1] Acar, E., Harrison, R.J., Olken, F., Alter, O., Helal, M., Omberg, L., Bader, B., Kennedy, A., Park, H., Bai, Z., Kim, D., Plemmons, R., Beylkin, G., Kolda, T., Ragnarsson, S., Delathauwer, L., Langou, J., Ponnapalli, S.P., Dhillon, I., heng Lim, L., Ramanujam, J.R., Ding, C., Mahoney, M., Raynolds, J., Eldén, L., Martin, C., Regalia, P., Drineas, P., Mohlenkamp, M., Faloutsos, C., Morton, J., Savas, B., Friedland, S., Mullin, L., Loan, C.V.: Future directions in tensor-based computation and modeling (2009)

[2] Adamic, L.A., Adar, E.: Friends and neighbors on the web. SOCIAL NETWORKS 25, 211–230 (2001)

[3] Airoldi, E.M., Blei, D.M., Fienberg, S.E., Xing, E.P.: Mixed membership stochastic blockmodels. Journal of Machine Learning Research 9, 1981–2014 (2008), `http://jmlr.org/papers/volume9/airoldi08a/airoldi08a.pdf`

[4] Bell, R.M., Koren, Y., Volinsky, C.: All together now: A perspective on the netflix prize. Chance (2010)

[5] Bennett, J., Lanning, S.: The netflix prize. In: Proceedings of the KDD Cup Workshop 2007. pp. 3–6. ACM, New York (Aug 2007), `http://www.cs.uic.edu/~liub/KDD-cup-2007/NetflixPrize-description.pdf`

[6] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13, 281–305 (2012), `http://dblp.uni-trier.de/db/journals/jmlr/jmlr13.html#BergstraB12`

[7] Bloehdorn, S., Sure, Y.: Kernel methods for mining instance data in ontologies. In: Aberer, K., Choi, K.S., Noy, N.F., Allemang, D., Lee, K.I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G.,

Cudré-Mauroux, P. (eds.) The Semantic Web — Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference (ISWC 2007 + ASWC 2007), November 11-15, 2007, Busan, Korea. Lecture Notes in Computer Science, vol. 4825, pp. 58–71. Springer, Berlin–Heidelberg, Germany (2007)

[8] Buja, A., Hastie, T., Tibshirani, R., Buja, A., Tibshirani, R.: Linear smoothers and additive models. The Annals of Statistics (1989)

[9] Candès, E.J., Recht, B.: Exact matrix completion via convex optimization. Computing Research Repository - CORR (2008)

[10] Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Jr., E.H., Mitchell, T.: Toward an architecture for never-ending language learning. In: Proceedings of the Conference on Artificial Intelligence (AAAI). pp. 1306–1313. AAAI Press (2010)

[11] Caroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via n-way generalization of Eckart–Young decomposition. Psychometrika 35, 283–319 (1970)

[12] Davis, J., Domingos, P.: Bottom-up learning of markov network structure. In: Fürnkranz, J., Joachims, T. (eds.) ICML. pp. 271–278. Omnipress (2010), http://dblp.uni-trier.de/db/conf/icml/icml2010.html#DavisD10

[13] Denham, W.: The detection of patterns in Alyawarra nonverbal behavior. Ph.D. thesis, University of Washington, Seattle, WA (1973)

[14] Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., Zhang, W.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 601–610. KDD '14, ACM, New York, NY, USA (2014), http://doi.acm.org/10.1145/2623330.2623623

[15] Fader, A., Soderland, S., Etzioni, O.: Identifying relations for open information extraction. In: EMNLP. pp. 1535–1545. ACL (2011), http://aclweb.org/anthology/D/D11/D11-1142.pdf

[16] Fensel, D., van Harmelen, F., Andersson, B., Brennan, P., Cunningham, H., Valle, E.D., Fischer, F., Huang, Z., Kiryakov, A., il Lee, T.K., Schooler, L., Tresp, V., Wesner, S., Witbrock, M., Zhong, N.: Towards larkc: A platform for web-scale reasoning. In: ICSC. pp. 524–529 (2008)

[17] Fortunato, S.: Community detection in graphs. CoRR abs/0906.0612 (2010)

[18] Franz, T., Schultz, A., Sizov, S., Staab, S.: Triplerank: Ranking semantic web data by tensor decomposition. In: International Semantic Web Conference (ISWC) (10 2009), `http://www.uni-koblenz.de/~staab/Research/Publications/2009/ISWC-triplerank-revised-version.pdf`

[19] Freeman, L.: A Set of Measures of Centrality Based on Betweenness. Sociometry 40, 35–41 (1977)

[20] Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: In IJCAI. pp. 1300–1309. Springer-Verlag (1999)

[21] Gärtner, T., Flach, P., Wrobel, S.: On graph kernels: Hardness results and efficient alternatives. In: Schölkopf, B., Warmuth, M.K. (eds.) Computational Learning Theory and Kernel Machines — Proceedings of the 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop (COLT/Kernel 2003) August 24-27, 2003, Washington, DC, USA. Lecture Notes in Computer Science, vol. 2777, pp. 129–143. Springer, Berlin–Heidelberg, Germany (August 2003)

[22] Gärtner, T., Lloyd, J.W., Flach, P.A.: Kernels and distances for structured data. Machine Learning 57(3), 205–232 (2004)

[23] Genesereth, M., Nilsson, N.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann, San Mateo, CA (1987)

[24] Getoor, L., Mihalkova, L.: Learning statistical models from relational data. In: Sellis, T.K., Miller, R.J., Kementsietsidis, A., Velegrakis, Y. (eds.) SIGMOD Conference. pp. 1195–1198. ACM (2011), `http://dblp.uni-trier.de/db/conf/sigmod/sigmod2011.html#GetoorM11`

[25] Getoor, L., Taskar, B. (eds.): Introduction to Statistical Relational Learning (2007)

[26] Glorot, X., Bordes, A., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data. CoRR abs/1301.3485 (2013), http://dblp.uni-trier.de/db/journals/corr/corr1301.html#abs-1301-3485

[27] Golub, G.H., Van Loan, C.F.: Matrix Computations (3rd Ed.). Johns Hopkins University Press, Baltimore, MD, USA (1996)

[28] Harshman, R.A.: Foundations of the PARAFAC procedure: models and conditions for an explanatory multimodal factor analysis. UCLA Working Papers in Phonetics 16, 1–84 (1970)

[29] Harshmann, R.: Models for analysis of asymmetrical relationships among n objects or stimuli. First Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology (1978), http://www.psychology.uwo.ca/faculty/harshman/asym1978.pdf

[30] Heckerman, D., Chickering, D.M., Meek, C., Rounthwaite, R., Kadie, C.M.: Dependency networks for inference, collaborative filtering, and data visualization. Journal of Machine Learning Research pp. 49–75 (2000)

[31] Heckerman, D., Meek, C., Koller, D.: Probabilistic models for relational data. Tech. Rep. MSR-TR-2004-30, Microsoft Research (March 2004), http://research.microsoft.com/apps/pubs/default.aspx?id=70050

[32] Heckerman, D.: A tutorial on learning with bayesian networks. Tech. rep., Learning in Graphical Models (1996)

[33] Heckerman, D., Meek, C.: Probabilistic entity-relationship models, prms, and plate models. In: In Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields. pp. 55–60 (2004)

[34] Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. J. Math. Phys 6(1), 164–189 (1927)

[35] Hitchcock, F.L.: Multiple invariants and generalized rank of a p-way matrix or tensor. J. Math. Phys 7(1), 39–79 (1927)

[36] Horváth, T., Gärtner, T., Wrobel, S.: Cyclic pattern kernels for predictive graph mining. In: Kim, W., Kohavi, R., Gehrke, J., DuMouchel, W. (eds.) Proceedings of the 10t ACM SIGKDD International Conference on Knowledge

Discovery and Data Mining (KDD 2004), August 22-25, 2004, Seattle, WA, USA. pp. 158–167. ACM Press, New York, NY, USA (2004), `http://doi.acm.org/10.1145/1014052.1014072`

[37] Huang, Y., Tresp, V., Bundschus, M., Rettinger, A., Kriegel, H.P.: Multivariate structured prediction for learning on semantic web. In: Frasconi, P., Lisi, F.A. (eds.) Proceedings of the 20th International Conference on Inductive Logic Programming (ILP 2010). Lecture Notes in Computer Science, vol. 6489, pp. 92–104. Springer Verlag, Firenze, Italy (2010)

[38] Järvelin, K., Kekäläinen, J.: Ir evaluation methods for retrieving highly relevant documents. In: SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 41–48. ACM, New York, NY, USA (2000), `http://portal.acm.org/citation.cfm?id=345508.345545&coll=PORTAL&dl=ACM&type=series&idx=SERIES278&part=series&WantType=Proceedings&title=SIGIR&CFID=23706865&CFTOKEN=89753871`

[39] Jeh, G., Widom, J.: SimRank: A Measure of Structural-Context Similarity. In: Proc. of International Conference on Knowledge Discovery and Data Mining (SIGKDD). ACM, Edmonton, Alberta, Canada (July 2002)

[40] Jenatton, R., Roux, N.L., Bordes, A., Obozinski, G.: A latent factor model for highly multi-relational data. In: Bartlett, P.L., Pereira, F.C.N., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) NIPS. pp. 3176–3184 (2012), `http://dblp.uni-trier.de/db/conf/nips/nips2012.html#JenattonRBO12`

[41] Jensen, D., Neville, J., Gallagher, B.: Why collective inference improves relational classification. In: Kim, W., Kohavi, R., Gehrke, J., DuMouchel, W. (eds.) KDD. pp. 593–598. ACM (2004), `http://dblp.uni-trier.de/db/conf/kdd/kdd2004.html#JensenNG04`

[42] Jiang, X., Huang, Y., Nickel, M., Tresp, V.: Combining information extraction, deductive reasoning and machine learning for relation prediction. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, s., Presutti, V. (eds.) ESWC. Lecture Notes in Computer Science, vol. 7295, pp. 164–178. Springer (2012), `http://dblp.uni-trier.de/db/conf/esws/eswc2012.html#JiangHNT12`

[43] Jiang, X., Tresp, V., Denis, K.: A logistic additive model for relation prediction in multi-relational data. In: The European Conference on Machine Learning and

Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD) 2013 Workshop on Tensor Methods for Machine Learning (2013)

[44] Jiang, X., Tresp, V., Huang, Y., Nickel, M.: Link prediction in multi-relational graphs using additive models. In: de Gemmis, M., Noia, T.D., Lops, P., Lukasiewicz, T., Semeraro, G. (eds.) SeRSy. CEUR Workshop Proceedings, vol. 919, pp. 1–12. CEUR-WS.org (2012), `http://dblp.uni-trier.de/db/conf/semweb/sersy2012.html#JiangTHN12`

[45] Jiang, X., Tresp, V., Huang, Y., Nickel, M., Kriegel, H.P.: Scalable relation prediction exploiting both intrarelational correlation and contextual information. In: Flach, P.A., Bie, T.D., Cristianini, N. (eds.) ECML/PKDD (1). Lecture Notes in Computer Science, vol. 7523, pp. 601–616. Springer (2012), `http://dblp.uni-trier.de/db/conf/pkdd/pkdd2012-1.html#JiangTHNK12`

[46] Jonsson, P.F., Bates, P.A.: Global topological features of cancer proteins in the human interactome. Bioinformatics 22(18), 2291–2297 (2006), `http://dblp.uni-trier.de/db/journals/bioinformatics/bioinformatics22.html#JonssonB06`

[47] Jordan, M.I.: Graphical models. Statistical Science 19, 140–155 (2004), `http://www.cs.berkeley.edu/~jordan/papers/statsci.ps`

[48] Kemp, C., Tenenbaum, J.B.: Learning systems of concepts with an infinite relational model. In: In Proceedings of the 21st National Conference on Artificial Intelligence (2006)

[49] Kersting, K., Raedt, L.D.: Bayesian Logic Programming: Theory and Tool, chap. 10. MIT Press (2007)

[50] Kok, S., Domingos, P.: Statistical predicate invention. In: In Z. Ghahramani (Ed.), Proceedings of the 24th annual international conference on machine learning (ICML-2007. pp. 433–440 (2007)

[51] Kok, S., Domingos, P.: Learning markov logic network structure via hypergraph lifting. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) ICML. ACM International Conference Proceeding Series, vol. 382, p. 64. ACM (2009), `http://dblp.uni-trier.de/db/conf/icml/icml2009.html#KokD09`

[52] Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM REVIEW 51(3), 455–500 (2009)

[53] Koller, D., Friedman, N., Getoor, L., Taskar, B.: Graphical models in a nutshell. In: Getoor, L., Taskar, B. (eds.) An Introduction to Statistical Relational Learning. MIT Press (2007)

[54] Koller, D., Pfeffer, A.: Probabilistic frame-based systems. In: In Proc. AAAI. pp. 580–587. AAAI Press (1998)

[55] Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 426–434. ACM (2008), `http://scholar.google.de/scholar.bib?q=info:oARx59_hO4MJ:scholar.google.com/&output=citation&hl=de&as_sdt=0,5&ct=citation&cd=0`

[56] Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer 42(8), 30–37 (Aug 2009), `http://dx.doi.org/10.1109/MC.2009.263`

[57] Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Machine Learning 81(1), 53–67 (2010), `http://dblp.uni-trier.de/db/journals/ml/ml81.html#LaoC10`

[58] de Lathauwer, L., de Moor, B., Vandewalle, J.: On best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of high-order tensors. SIAM J. Matrix Anal. Appl. 21, 1324–1342 (2000)

[59] Lehoucq, R.B., Sorensen, D.C., Yang, C.: ARPACK: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods. Available from netlib@ornl.gov (1997)

[60] Levin, J.: Three-mode factor analysis. Psychological Bulletin 64, 442–452 (1965)

[61] Liben-Nowell, D., Kleinberg, J.: The link-prediction problem for social networks. Journal of the American society for information science and technology 58(7), 1019–1031 (2007), `http://scholar.google.com/scholar.bib?q=info:U1ggap2AFkIJ:scholar.google.com/&output=citation&hl=en&as_sdt=0,5&as_vis=1&ct=citation&cd=0`

[62] Lösch, U., Bloehdorn, S., Rettinger, A.: Graph kernels for rdf data. In: Simperl, E., Cimiano, P., Polleres, A., Corcho, s., Presutti, V. (eds.) The Semantic Web:

Research and Applications - 9th Extended Semantic Web Conference, ESWC 2012. Lecture Notes in Computer Science, vol. 7295, pp. 134–148. Springer Verlag (May 2012), best Research Paper Award

[63] Macskassy, S.A., Provost, F.: A simple relational classifier. In: Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003. pp. 64–76 (2003)

[64] Madan, A., Cebrian, M., Moturu, S., Farrahi, K., Pentland, S.: Sensing the "health state" of a community (2012), `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6072198`

[65] Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. SIAM (2001)

[66] Murphy, K.P., Weiss, Y., Jordan, M.I.: Loopy belief propagation for approximate inference: An empirical study. In: In Proceedings of Uncertainty in AI. pp. 467–475 (1999)

[67] Nakashole, N., Weikum, G., Suchanek, F.M.: Discovering and exploring relations on the web. PVLDB 5(12), 1982–1985 (2012), `http://dblp.uni-trier.de/db/journals/pvldb/pvldb5.html#NakasholeWS12`

[68] Neville, J., Jensen, D.: Relational dependency networks. Journal of Machine Learning Research 8, 653–692 (2007), `http://dblp.uni-trier.de/db/journals/jmlr/jmlr8.html#NevilleJ07`

[69] Nickel, M.: Tensor factorization for relational learning (August 2013), `http://nbn-resolving.de/urn:nbn:de:bvb:19-160568`

[70] Nickel, M., Jiang, X., Tresp, V.: Reducing the rank in relational factorization models by including observable patterns. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 27, pp. 1179–1187. Curran Associates, Inc. (2014)

[71] Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: Getoor, L., Scheffer, T. (eds.) ICML. pp. 809–816. Omnipress (2011), `http://dblp.uni-trier.de/db/conf/icml/icml2011.html#NickelTK11`

[72] Nickel, M., Tresp, V., Kriegel, H.P.: Factorizing yago: scalable machine learning for linked data. In: Mille, A., Gandon, F.L., Misselis, J., Rabinovich, M., Staab,

S. (eds.) WWW. pp. 271–280. ACM (2012), `http://dblp.uni-trier.de/db/conf/www/www2012.html#NickelTK12`

[73] Nienhuys-Cheng, S.H., de Wolf, R. (eds.): Foundations of Inductive Logic Programming, Lecture Notes in Computer Science, vol. 1228. Springer (1997)

[74] Niu, F., Ré, C., Doan, A., Shavlik, J.W.: Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. PVLDB 4(6), 373–384 (2011), `http://dblp.uni-trier.de/db/journals/pvldb/pvldb4.html#NiuRDS11`

[75] Niu, F., Zhang, C., Re, C., Shavlik, J.W.: Scaling inference for markov logic via dual decomposition. In: Zaki, M.J., Siebes, A., Yu, J.X., Goethals, B., Webb, G.I., Wu, X. (eds.) ICDM. pp. 1032–1037. IEEE Computer Society (2012), `http://dblp.uni-trier.de/db/conf/icdm/icdm2012.html#NiuZRS12`

[76] Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. In: Proceedings of the 7th International World Wide Web Conference. pp. 161–172. Brisbane, Australia (1998), `citeseer.nj.nec.com/page98pagerank.html`

[77] Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. Proceedings of KDD Cup and Workshop pp. 39–42 (2007)

[78] Perlich, C., Provost, F.J.: Aggregation-based feature invention and relational concept classes. In: Getoor, L., Senator, T.E., Domingos, P., Faloutsos, C. (eds.) KDD. pp. 167–176. ACM (2003), `http://dblp.uni-trier.de/db/conf/kdd/kdd2003.html#PerlichP03`

[79] Perlich, C., Provost, F.J.: Distribution-based aggregation for relational learning with identifier attributes. Machine Learning 62(1-2), 65–105 (2006), `http://dblp.uni-trier.de/db/journals/ml/ml62.html#PerlichP06`

[80] Quinlan, J.R.: Learning logical definitions from relations. MACHINE LEARNING 5, 239–266 (1990)

[81] Raedt, L.D., Dehaspe, L.: Clausal discovery. Machine Learning 26(2), 99–146 (1997), `http://dx.doi.org/10.1023/A:1007361123060`

[82] Richardson, M.: Predicting clicks: Estimating the click-through rate for new ads. In: In Proceedings of the 16th International World Wide Web Conference (WWW-07. pp. 521–530. ACM Press (2007)

[83] Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62(1-2), 107–136 (2006), http://dblp.uni-trier.de/db/journals/ml/ml62.html#RichardsonD06

[84] Sabidussi, G.: The centrality index of a graph. Psychometrika 31(4), 581–603 (December 1966), http://ideas.repec.org/a/spr/psycho/v31y1966i4p581-603.html

[85] Salton, G., McGill, M.J.: Introduction to modern information retrieval. McGraw-Hill, New York (1983)

[86] Sarawagi, S.: Information extraction. Foundations and Trends in Databases 1(3), 261–377 (2008)

[87] Sarkar, S., Dong, A.: Community detection in graphs using singular value decomposition. Phys. Rev. E 83(4), 046114 (Apr 2011)

[88] Sarwar, B.M., Karypis, G., Konstan, J.A., Riedl, J.T.: Application of dimensionality reduction in recommender system – a case study. In: IN ACM WEBKDD WORKSHOP (2000)

[89] Sen, P., Namata, G.M., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 29(3), 93–106 (2008), http://www.cs.iit.edu/~ml/pdfs/sen-aimag08.pdf

[90] Shervashidze, N., Borgwardt, K.: Fast subtree kernels on graphs. In: Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., Culotta, A. (eds.) Advances in Neural Information Processing Systems 22 – Proceedings of the 2001 Neural Information Processing Systems Conference NIPS 2009, December 7-10, 2009 Vancouver, British Columbia, Canada, pp. 1660–1668. Neural Information Processing Systems Foundation (2009), http://books.nips.cc/papers/files/nips22/NIPS2009_0533.pdf

[91] Singh, A.P., Gordon, G.J.: Relational learning via collective matrix factorization. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 650–658. KDD '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1401890.1401969

[92] Strang, G.: Introduction to Linear Algebra. Wellesley-Cambridge Press, Wellesley, MA (2009)

[93] Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., Oberle, D.: The swrc ontology - semantic web for research communities. In: Proceedings of the 12th Portugese Conference on Artificial Intelligence - Progress in Artificial Intelligence (EPIA 2005), volume 3803 of LNCS, pages 218 - 231, Covilha. pp. 218 –231. Springer (2005)

[94] Sutskever, I., Salakhutdinov, R., Tenenbaum, J.B.: Modelling relational data using bayesian clustered tensor factorization. In: Bengio, Y., Schuurmans, D., Lafferty, J.D., Williams, C.K.I., Culotta, A. (eds.) NIPS. pp. 1821–1828. Curran Associates, Inc. (2009), `http://dblp.uni-trier.de/db/conf/nips/nips2009.html#SutskeverST09`

[95] Takács, G., Pilászy, I., Németh, B., Tikk, D.: Major components of the gravity recommendation system. SIGKDD Explorations 9(2), 80–83 (2007), `http://dblp.uni-trier.de/db/journals/sigkdd/sigkdd9.html#TakacsPNT07`

[96] Takacs, G., Pilaszy, I., Nemeth, B., Tikk, D.: On the gravity recommendation system. In: Proceedings of KDD Cup and Workshop 2007 (2007)

[97] Taskar, B., Abbeel, P., Koller, D.: Discriminative probabilistic models for relational data. In: Darwiche, A., Friedman, N. (eds.) UAI. pp. 485–492. Morgan Kaufmann (2002), `http://dblp.uni-trier.de/db/conf/uai/uai2002.html#TaskerPK02`

[98] Tresp, V., Huang, Y., Bundschus, M., Rettinger, A.: Materializing and querying learned knowledge. RMLeS (2009)

[99] Tresp, V., Nickel, M.: Relational models. In: Encyclopedia of Social Network Analysis and Mining. Springer (2014)

[100] Tresp, V., Yu, K.: Tutorial summary: Learning with dependencies between several response variables. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) ICML. ACM International Conference Proceeding Series, vol. 382, p. 174. ACM (2009), `http://dblp.uni-trier.de/db/conf/icml/icml2009.html#TrespY09`

[101] Tucker, L.R.: Implications of factor analysis of three-way matrices for measurement of change. In: Harris, C.W. (ed.) Problems in measuring change., pp. 122–137. University of Wisconsin Press, Madison WI (1963)

[102] Tucker, L.R.: The extension of factor analysis to three-dimensional matrices. Contributions to mathematical psychology pp. 109–127 (1964)

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

[104] Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika 31, 279–311 (1966c)

[105] Tucker, L.: Implications of factor analysis of three-way matrices for measurement of change. Problems in measuring change pp. 122–137 (1963)

[106] Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. J. Mach. Learn. Res. 11, 1201–1242 (Aug 2010), `http://dl.acm.org/citation.cfm?id=1756006.1859891`

[107] Xu, Z., Tresp, V., Yu, K., peter Kriegel, H.: Infinite hidden relational models. In: In Proceedings of the 22nd International Conference on Uncertainity in Artificial Intelligence (UAI) (2006)

[108] Zheleva, E., Getoor, L., Golbeck, J., Kuter, U.: Using friendship ties and family circles for link prediction. In: SNAKDD. pp. 97–113 (2008)

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.

Name, Vorname

....	....
Ort, Datum	Unterschrift Doktorand/in

Formular 3.2