INDEPENDENCE ASSUMPTIONS FOR MULTI-RELATIONAL CLASSIFICATION

by

Bahareh Bina
BEng, University of Tehran, 2007

a Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Spring 2011

APPROVAL

Name: Bahareh Bina

Degree: Master of Science

Title of Thesis: Independence Assumptions for Multi-Relational Classification

Examining Committee: Dr. Ke Wang
Chair

Dr. Oliver Schulte, Associate Professor, Computing Science
Simon Fraser University
Senior Supervisor

Dr. Jian Pei, Associate Professor, Computing Science
Simon Fraser University
Supervisor

Dr. Martin Ester, Professor, Computing Science
Simon Fraser University
SFU Examiner

Date Approved: January 18, 2011

Declaration of
Partial Copyright Licence

Abstract

Link-based classification (LBC) is the problem of predicting the class attribute of a target entity given the attributes of entities linked to it. A natural approach to relational classification is to upgrade standard classification methods from the propositional, single-table learning. Decision trees and Bayesian networks are two of the widely used single-table classification models. In this thesis, we propose two algorithms for upgrading decision tree and Bayes net learners for LBC. One of the issues that make LBC difficult compared to single table learning is the large number of different types of dependencies that a model may have to consider. A principled way to approach the complexity of correlation types is to consider model classes with explicitly stated independence assumptions. We define two independence assumptions weaker than relational naive Bayes assumption to formulate our classification formulas. We enhance the performance of our upgraded decision tree by applying logistic regression to prune irrelevant tables and consider different weights for information from different tables. Moreover, we describe a fast Bayes net structure learning algorithm for our upgraded Bayes net learner. We investigate the performance of our models on three real world datasets. Experimental results indicate that our models are very fast and achieve better classification performance compared to a variety of relational classifiers.

To my parents for their love and support.

Science is the belief in the ignorance of experts – Richard Feynman

Acknowledgments

I would like to express my special thanks to my supervisor Dr. Oliver Schulte for his help and guidance. I cannot thank him enough for introducing me to the exciting world of research. Without his support, encouragement, and collaboration, I would certainly not get what I have today.

I would also like to thank the other members of my committee Dr. Jian Pei and Dr. Martin Ester, for their help.

Besides, There are some friends who helped me pass the difficult student life through emotional challenges and research plateaus. I could not have done it without Maryam Sadeghi, Majid Razmara and Ehsan Iranmanesh. They were always there for me and offered infinite moral and intellectual support. I would like to express my appreciation to my lab mate at Simon Fraser University, Hassan Khosravi, for the discussions and friendship.

I owe to my family who always support me. To my father, who encouraged me all the time and to my mother, who taught me to tolerate difficulties and not lose hope, and thanks to my brother who helped me grew faster.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Many databases store structured information in relational format with different tables for each type of entities and for each relationship type (link) between the entities. In the real world, most data are relational. Example of such data include financial data of a bank, epidemiology data, and rating information provided by users for different movies.

Multi-relational data mining deals with knowledge discovery from relational databases consisting of multiple tables. Instead of transforming data from multiple tables into a single table, multi-relational data mining seeks to collect and analyze data directly from a relational database. It has attracted a significant amount of interest and has been applied to different domains.

Relational data offer additional opportunities to augment model accuracy and to improve prediction quality by providing more information that resides in objects, their linked objects and relationships. An important task in multi-relational data mining is link-based classification (LBC) which takes advantage of attributes of links and linked entities in addition to attributes of the target entity to predict the class label [31].

Algorithms for classifying a single table of data assume that instances are independent and identically distributed (i.i.d.), but relational data violate this assumption; objects in relational data are dependent through direct relationships or a chain of relationships.

Many traditional classification approaches deal with a single table of data. Therefore, one approach to classify relational data is to collapse relational data into a single table, which is known as propositionalization , and apply the traditional classification algorithms. In order to do that, two methods are proposed:

1. Flattening: join of all the tables to form a single table.

2. Feature construction: create new attributes that summarize or aggregate information from different links [28, 49, 24, 42].

The other approach is to upgrade traditional classification algorithms to deal with multi-relational data directly. Most methods with upgrading approach fall in the category of Inductive Logic Programming (ILP) [36, 35, 34]. ILP systems like Claudien and ICL are first order upgrades of propositional data mining algorithms that induce association rules. The other classification models are relational rule learners [9] and relational decision trees [24] which upgrade their single table model respectively. Probabilistic Relational Models [14] and Bayesian Logic Programs [26] are extensions of Bayesian networks to relational domains, and Relational Markov Random Fields [50] and Markov Logic Networks [10] upgrade Markov Random Fields. Naive Bayes classifier is a well known single table classifier which is easy to learn and implement; relational naive Bayes classifier [6] is its relational version with one more independence assumption.

This thesis describes two new upgrading classification algorithms: Simple Decision Forest that extends decision tree and Path Independence Classifier that upgrades Bayesian network to deal with relational data. Our goal is to minimize information loss by exploiting intra and inter table dependencies.

1.1 Motivation

As we explained before, relational classifiers are either based on propositionalization methods or upgrading approaches. Each of these approaches has its limitations. In the following, we discuss some of them.

The problems of flattening to propositionalize relational data are:

- The big join table may be too large and difficult to manage.
- Probabilities of events may not be computed correctly from the big join table.
- Data redundancy is increased in the big join table.

The drawbacks of feature construction to propositionalize relational data are:

- It is not easy to find appropriate new attributes.

- Summarizing information into one value causes information loss.

ILP techniques deal with relational data in the Prolog format; therefore, one of their main limitations is to change the input type as most real-world data are stored in relational databases. ILP methods are inefficient and time consuming as they search the whole hypothesis space for useful attributes or relational structural neighbors.

Most of the work on relational versions of decision trees, rule learners, and naive Bayes classifiers uses a mechanism for *relational feature generation*. Feature generation is done by using aggregate functions to summarize the information in links [28, 49, 24, 42] or learning an existentially quantified logical rule [29]. Several recent systems combine both aggregation and logical conditions (e.g., [53, 42]). However, the predictors in our model are the descriptive attributes as defined in the relational database schema. Relational feature generation is by far the most computationally demanding part of such approaches. For example, generating 100,000 features on the CiteSeer dataset, which is smaller than the databases we consider in this paper, can take several CPU days [42, Ch.16.1.2]. So using models that use independence assumptions rather than generated features to combine the information from different tables is orders of magnitude faster to learn, as our experiments confirm.

Probabilistic Relational Models (PRMs) [14] upgrade directed graphical models to deal with relational data. Getoor et al. [18] extend PRMs with link indicator nodes to model link structure between entities, and constrain the PRM such that the probability of a link indicator being true is a function of the attributes of the linked entities (cf. [50]). However, they did not give a structure learning algorithm for this model, and combined it with a Naive Bayes classifier.

Unlike directed graphical models which impose an acyclicity constraint, undirected ones do not have a cyclicity problem and are widely used for LBC [50, 10, 22]. Undirected graphical models can represent essentially arbitrary dependencies [50] compared to directed models. The trade-off for the expressive power of undirected models is higher complexity in learning, especially scalable model learning is a major challenge in the multi-relational setting [22].

One of the issues that makes LBC difficult compared to single-table learning is the large number of different types of dependencies that a model may have to consider [54]. A principled way to approach the complexity of correlation types is to consider model classes with explicitly stated independence assumptions. A prominent example of this approach are multi-relational Naive Bayes net classifiers (NBCs) [37, 6]. NBCs incorporate two different

kinds of independence assumptions:

1. *Across-table independencies*: information from different tables is independent given the target class label.

2. *Within-table independencies*: descriptive attributes from the same table are independent given the target class label.

All of these drawbacks severely limit the ability of current relational classifiers.

1.2 Our Approach

In this thesis, we propose two new relational classifiers. For the first one, we upgrade decision tree classifiers from the propositional, single-table testing. We use a decision tree as a base learner for dependencies within a single table; decision trees are able to select relevant features. The decision trees we consider provide probability estimates in their leaves, rather than simply class labels. Such trees are sometimes called probability estimation trees [43]. We learn a *forest of decision trees* that contains a tree over attributes of the target entity and a tree over attributes of each of the linked entity tables extended by the class label. To combine the predictions of decision trees, we formalize the cross-table Naive Bayes (NB) independence assumption and derive a closed-form classification formula. To compare with relational NBC, we maintain the across-table independencies assumption, but drop the within-table independencies assumption. This allows us to capture and exploit more dependencies that may hold between the attributes and the class label within each table. The formula is equivalent to a log-linear regression model where the log-probability of the class label is the dependent (predicted) variable, and the independent variables (predictors) are the probability estimates of the form $P(label|table_i)$, for the target table and linked tables. These estimates are obtained by applying a decision tree learner to each linked table. Then we allow different decision trees to contribute more or less strongly to the classification decision, with weights for the relative contributions learned by a logistic regression algorithm. Because the regression is not applied directly to data, but rather to decision trees learned from the data, we refer to it as *meta-regression*.

We call our method *Simple Decision Forest* as we learn different decision trees for different tables of the dataset and in this context Forest does not mean that we apply an

ensemble method. For clarification, we provide a list of differences of our method with ensemble methods or fusion methods [47] in the following.

1. Ensemble methods are usually used on propositional data and divide the training set into a number of subsets randomly. We do not use random subsampling techniques to build a set of classifiers. Instead, we use the database schema to determine a partition of the input feature space, and for each partition, there is a separate classifier.

2. The goal of Ensemble methods is to exploit the diversity among the classification methods by combining them, but our goal is to combine data in several tables.

3. Formalizing the independence assumption, we derive a formula for combining the results of different classifiers, whereas ensemble methods use classifier fusion methods to combine the results that do not have an explicit semantics in terms of independendence assumptions.

4. We apply a logistic regression algorithm scheme to learn weights for information from different tables and prune tables with weight 0. However, logistic regression used in ensemble methods has no interpretation in terms of feature or table pruning.

The second model is a multi-relational Bayesian network classifier based on the *Path Independence Assumption* (PI). A path is a chain of links that connect the target entity to a related entity. Under PI different paths associating an object to other objects are independent given the attributes of the objects. For example, two courses that an intelligent student has taken can be chosen independently from the same distribution given the difficulty of the courses. PI assumption is weaker than the assumption we used for combing different decision trees in the first model. The main advantages of the path independent assumption compared to other multi-relational classifiers are as follows.

1. The PI assumptions models all the attributes dependencies given the path structures in the relational data, but not correlations among the paths. Therefore, the BN models are simpler and have fewer parameters. However, the simplicity has empirically been shown not to hurt the classification performance.

2. There is no problem with cycles in the instantiated network (grounding) of the directed graph [50, 18].

3. Aggregation functions (e.g. average) [14] or combining rules [26] are not required. Thus no information is lost [6] and the computationally expensive search for aggregate features is avoided.

An empirical comparison of our random decision forest with other multi-relational decision tree learners on *three* benchmark datasets shows a large run-time advantage, and strong predictive performances of our models that are better than that of previous multi-relational decision tree learners, and a state-of-the-art multi-relational Naive Bayes classifier.

We compare the performance of our Path Independence Bayes net classifier on three benchmark datasets with multi-relational Naive Bayes classifier and undirected model[10]. Experimental results indicates that the Path Independence model achieves the best accuracy and high run-time efficiency.

Furthermore, two proposed algorithms are compared against each other. Results show that the Simple Decision Forest with meta regression algorithm outperform the Path Independence classifer.

1.3 Contributions

The main contributions of our work are as follows.

1. Propose two novel relational classifiers based on independence assumptions.
2. Apply a weaker assumption than relational naive Bayes classifiers to model more dependencies in data.
3. The formal definition of independence assumptions and derivation of closed form classification formulas.
4. Develop an algorithm to learn the structure of relational Bayesian network classifier
5. Use of logistic (meta) regression with Simple Decision Forest to assign weights for the contributions of different link tables, including zero weights for pruning. To our knowledge, our work is the first to apply logistic meta regression in multi-relational classification.
6. A comparison of our relational classifiers with each other, and with undirected graphical model (Markov logic networks) and relational decision trees n three real life

databases, demonstrating that our methods are very fast with good predictive accuracy.

1.4 Thesis Organization

We first describe related work and other multi-relational classification models. In chapter three, first we define our notation, then we formalize the independence assumptions for multi-relational learning and derive the classification formula. We describe the meta regression, simple forest induction and classification algorithms. In chapter four, we define our new relational BN classifier. Chapter five evaluates the classification performance of the different models on three benchmark datasets. Finally, we conclude our thesis in chapter six.

Chapter 2

Multi-Relational Data Classification

Multi-relational classification aims to discover useful patterns in relational databases. In this chapter, first we define relational data and their major methods of representation. Then we discuss the main approaches of multi-relational classification.

2.1 Relational Data

In traditional data classification, algorithms deal with propositional data, stored in a single table. However, real world data are mostly relational. In the following, the differences between relational and propositional data are discussed.

- Relational data often contain information about different types of objects, whereas single table data contain information about a single object type.
- Objects in single table data are typically assumed to be independent and identically distributed (iid); however, objects in relational data may be correlated with a different distribution for each type of objects.
- The set of attributes, learners can use propositional data mining is limited to that of each object, whereas in relational data mining, learners can use information not only from the target object but also from its direct or indirect links. Therefore, relational

data learners consider different pathways through which objects may be linked to each other.

- Attribute values of two objects with the same type may be correlated to each other in relational data which is called autocorrelation. For example, the gene type of a child is correlated to the genes of his parents. The presence of autocorrelation increases the complexity of relational learning.
- There is no fixed reference structure in relational data; the number of links of different objects is variable [17]. For instance, the number of friends of people is different.

Objects in relational data can be represented in different formats. The major types of relational data representation are:

1. Relational Data Base (RDB): A set of tables with different types of entities and relationships between the entities. Entities are linked to each other via foreign key constraints.
2. Main Memory Prolog Facts: A set of first-order logic statements in which entities correspond to populations, descriptive attributes to function symbols, relationships to predicate symbols, and foreign key constraints to type constraints on the first-order variables [31].

The next section discusses the main approaches to multi-relational data classification.

2.2 Multi-Relational Data Classification Methods

Multi-relational data mining aims to discover interesting knowledge from multiple tables. An important task in multi-relational data mining is classification. It takes advantage of attributes of links and linked entities in addition to attributes of the target entity to predict the class label [31]. To learn a classifier for relational data, existing approaches either *flatten* multiple relations into a single table, which is then used as the input of propositional learning methods [28, 49, 24, 42] or *upgrade* propositional algorithms to deal with relational data [20, 6, 23, 11]. Transforming data into a single table requires considerable time, results in the loss of information [6], and produces a large table with many additional attributes. Upgrading algorithms are faster and more accurate, therefore, we discuss them in detail in this chapter. Some of the main upgrading models are:

- Directed graphical models
- Undirected graphical models
- ILP rule learners
- Decision trees
- Relational Models with Independence Assumption

2.2.1 Directed Graphical Models

A major approach to probabilistic relational classification that has attracted a significant amount of interest is based on graphical models. Directed graphical models, specifically Bayesian networks, use a directed graph representation to encode a complete distribution over a database.

Brief Introduction to Bayesian Networks

A Bayesian network consists of two components:

- Structure: An acyclic graph showing correlations between different nodes (random variables) via directed edges.
- Parameters: A conditional probability table for each node storing the distribution over values of the node given possible assignments to its parents.

A random variable is a pair $X = \langle dom(X), P_X \rangle$ where $dom(X)$ is a set of possible values for X called the **domain** of X and $P_X : dom(X) \rightarrow [0, 1]$ is a probability distribution over these values. For simplicity, we assume that all random variables have finite domains (i.e., discrete or categorical variables). An **atomic assignment** assigns a value $X = x$ to random variable x, where $x \in dom(X)$. A **joint distribution** P assigns a probability to each conjunction of atomic assignments; we write $P(X_1 = x_1, ..., X_n = x_n) = p$, sometimes abbreviated as $P(x_1, ..., x_n) = p$. The **conditional probability** $P(X_1 = x_1 | X_2 = x_2)$ is defined as $P(X_1 = x_1, X_2 = x_2)/P(X_2 = x_2)$. Bayes nets factorize the joint probability over an assignment of values to attributes, as a product over the probability of each attribute value given its parents value assignment [39].

The Markov blanket of a node is the set of its parents, children, and parents' children. Every node in a BN is conditionally independent of others given its Markov blanket. For the general conditional independence in a Bayesian network, Pearl [16] proposed the *d-separation* concept. Two sets of nodes X and Y are d-separated in Bayesian networks by a third set Z, excluding X and Y, if and only if for every path between X and Y there is an intermediate variable $V \in Z$ (distinct from X and Y) such that:

- The connection through V is "tail-to-tail" or "tail-to-head".

- Or, the connection through V is "head-to-head" and neither V nor any of V's descendants have received evidence.

Figure 2.1 depicts different connection states of nodes in BNs.

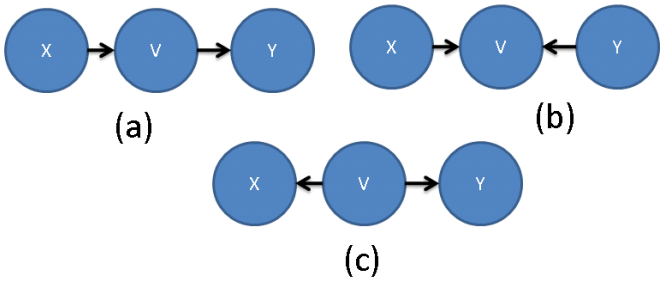


Figure 2.1: Different graphical patterns: (a) tail to head; (b) head to head; (c) tail to tail

D-separation implies that If two sets of nodes X and Y are d-separated in a Bayesian network by a third set Z, the corresponding variable sets X and Y are independent given the variables in Z. The minimal set of nodes which d-separates node A from all other nodes is its Markov blanket. If two nodes X and Y are not d-separated by Z in a BN, then X and Y are **d-connected** by Z.

Parameterized Bayesian nets (PBN) extend Bayesian networks to deal with relational database directly. In the following, PBNs are explained in detail.

Parameterized Bayesian Network (PBN):

Parametrized Bayes nets specifies a relational Bayes net as a template for a database probability distribution. We follow the original presentation of Poole [40]: A **population** is a set of individuals, corresponding to a domain or type in logic. A parametrized random variable is of the form $f(t_1, \ldots, t_k)$ where f is a **functor** (either a function symbol or a predicate symbol) and each t_i is a first-order variable or a constant. Each functor has a set of values

Student(<u>*student_id*</u>, *intelligence*, *ranking*,*sex*, *marital*, *age*)
Course(<u>*course_id*</u>, *difficulty*, *rating*)
Professor (<u>*prof_id*</u>, *teaching_ability*, *popularity*)
Registered (<u>*student_id*, *course_id*</u>, *grade*, *satisfaction*)
RA (<u>*student_id*, *prof_id*</u>, *salary*, *capability*)

Table 2.1: A relational schema for a university domain. Key fields are underlined.

(constants) called the **range** of the functor. An assignment of the form $f(t_1, \ldots, t_k) = a$, where a is a constant in the range of f is an **atom**; if all terms t_i are constants, the assignment is a **ground** atom. A Parametrized Bayes Net is a Bayes net whose nodes are functors. A **parametrized Bayes net structure** consists of

1. A directed acyclic graph (DAG) whose nodes are parametrized random variables.
2. A population for each first-order variable.
3. An assignment of a range to each functor.

The functor syntax is rich enough to represent a relational schema follows an entity-relationship model (ER model) [51, Ch.2.2] via the following translation: Entity sets correspond to populations, descriptive attributes to function symbols, relationship tables to predicate symbols, and foreign key constraints to type constraints on the first-order variables. Consider the university schema in table 2.1. Figure 2.2 shows a parametrized Bayes net structure for this schema.

We assume that a database instance (interpretation) assigns a constant value to each gnode $f(\mathbf{a})$, which we denote by $[f(\mathbf{a})]_{\mathcal{D}}$. PBNs represent generic statistical relationships found in the database. For instance, a PBN may encode the probability that a student is highly intelligent given the difficulty of a single course taken by the student. But the database may contain information about many courses the student has taken, which needs to be combined; we refer to this as the combining problem. To address the combining problem, one needs to use an aggregate function, as in PRMs, or a combining rule as in BLPs, or applying conditional independence assumptions as we do in this thesis. In PRMs and BLPs, the aggregate functions respectively combining rules add complexity to the model learning algorithm.

The PBN's template can be instantiated by the objects in the database to generate a massive Bayesian network, called the *ground Bayes net*. In the ground Bayes net the parameters for the attributes of objects with the same type are identical.

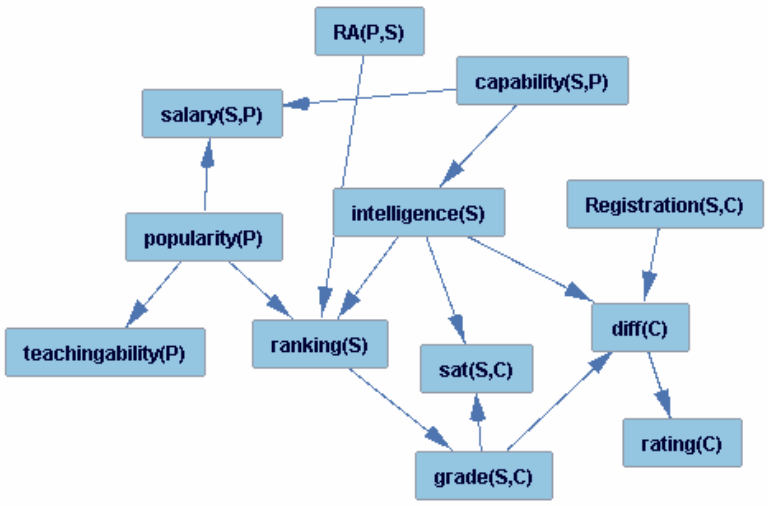


Figure 2.2: A parametrized BN graph for the relational schema of Table 2.1.

Directed SRL Models may face the *cyclicity problem* in their ground networks: there may be some cyclic dependencies between properties of individual entities (this is also known as relational autocorrelation [25]). For example, if there is generally a correlation between the smoking habits of friends, then we may have a situation where the smoking of Jane predicts the smoking of Jack, which predicts the smoking of Cecile, which predicts the smoking of Jane, where Jack, Jane, and Cecile are all friends with each other. In the presence of such cycles, neither aggregate functions nor combining rules lead to well-defined probabilistic predictions. Therefore, only an acyclic directed graph specifies a coherent probability distribution over objects in the database.

Probabilistic Relational Models

A PRM defines a probability distribution over the set of instances of a schema assuming that the set of objects and their relations are fixed [14].A PRM specifies a relational Bayes net as a template for a database probability distribution. In the relational Bayes net each node X can have two different types of parent Y:

- X and Y are from the same table.
- X and Y are from different tables but there is a *slot chain* between them. Slot chain is a foreign key path from one object to another which are not directly related.

Except in cases where the slot chain is guaranteed to be single valued, the second type of parent can be a set. For example, a student's rank may depend on the grades of the courses she has taken, so grade is a multi-valued parent node of ranking. To address the problem of combining different values for an attribute, PRMs use an aggregate function (e.g., mode, mean, median, maximum, and minimum). PRMs learning tasks consist of learning the parameters (conditional probability distribution) and the structure (DAG). The parameter estimation is the process of maximizing the likelihood of the data given the model. For structure learning three components need to be defined:

1. The hypothesis space specifies the valid structures that the algorithm can consider.
2. A scoring function evaluates the goodness of each hypothesis.
3. A search algorithm is a heuristic function that defines an ordering on the structures being considered, mostly greedy hill-climbing search.

Bayesian Logic programs (BLP)

BLPs are considered as one of the successful models of relational learning [26]. BLPs use logic programming to unify Bayesian networks with logic programming. This unification overcomes the propositional character of Bayesian networks and the purely logical nature of logical programs. In BLPs, the structure of the Bayesian network is represented by a set of Bayesian definite clauses. The parameters are as in BN stored in conditional probability tables (CPTs) which represent the distribution of the head of each clause conditional on its body. BLPs use combining rules to unite the information on a single literal that is the head of several clauses. A query can be answered by using any Bayesian net inference engine. Algorithms for learning BLPs (the structure and parameters) have been proposed by Kersting et al.[26]. The structure learning in BLPs follows the procedure of rule learning in ILP systems which have operators such as adding and deleting logically valid literals, instantiating variables, and unifying variables on literals or clauses.

2.2.2 Undirected Graphical Models

Undirected graphical models, specifically Markov random fields, use an undirected graph to specify the data distribution.

Brief Introduction to Markov Random Fields

In undirected graphs, independence can be established simply by graph separation: if every path from a node in X to a node in Y goes through a node in Z , we conclude that X and Y are independent given Z. Markov random fields are similar to Bayesian networks in their representation type of dependencies, but on one hand they can represent certain dependencies that a Bayesian network cannot (such as cyclic dependencies) as shown in Figure 2.3 (a), on the other hand, they cannot represent certain dependencies that Bayesian networks can (such as induced dependencies)[1] as shown in figure 2.3(b).

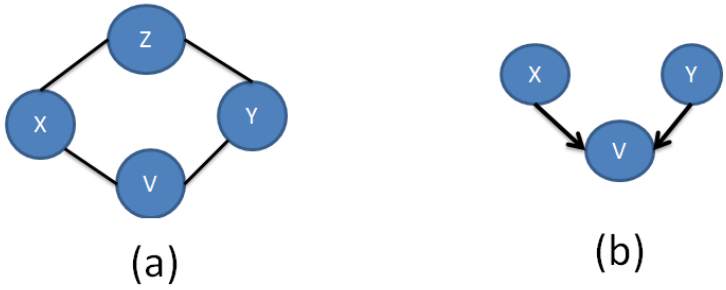


Figure 2.3: (a) shows the independence relation in Markov random fields that Bayesian nets cannot represent; (b) shows the independence relation in Bayesian nets that Markov random fields cannot represent.

Markov random fields factorize the joint probability distribution into a product over the clique potentials of the graph. A clique is a fully connected subset of nodes.

$$P(x_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) = \frac{1}{Z} \prod_{cliquesc} \Psi_c(\vec{x_c}) \tag{2.1}$$

where $\vec{x_c}$ shows the values for the variables in the clique c and Z is a normalization constant.

Relational Markov network and Markov logic network extend Markov random fields to deal with relational databases directly.

Relational Markov Network (RMN):

Relational Markov networks extend traditional Markov networks over relational data sets. The graphical structure of an RMN is based on the relational structure of the domain. An RMN specifies the cliques between attributes of related entities at the template level and

the potentials of cliques. Therefore, a single model provides a coherent distribution for any collection of instances from the schema. RMNs are a general log-linear model that do not use aggregate functions nor independence assumptions [50]. Furthermore, they do not impose the acyclicity constraint that hinders representation of many important relational dependencies in directed models. The trade-off for the expressive power of undirected models is the higher complexity in learning, especially scalable model learning is a major challenge in RMNs [27].

Markov Logic Network (MLN):

Markov logic networks combine first-order logic with probability theory. An MLN is a collection of formulas from first order logic with a real number, weight, assigned to each of them [10]. Considering an MLN as a Markov network, the grounding of each formula is considered as a clique and the weight of each formula is the clique potential. Moreover, the set of formulas in which a given atom appears are the neighbors of the node in the Markov network. The ground Markov network is used for inference and learning on the model. Example of an MLN a the university database can be seen in Figure 2.4.

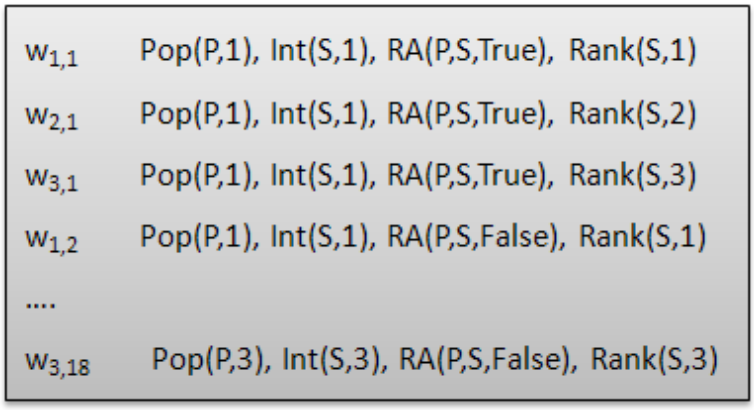


Figure 2.4: Example of an MLN for the university database

2.2.3 ILP Rule Learners

Inductive logic programming (ILP) is a general term assigned to relational rule learners using logic programming languages. Given a logic encoding of the known background knowledge and a set of positive and negative examples represented as a logical database of facts, an

ILP classifier will derive a hypothesised logic program which entails all the positive and none of the negative examples. FOIL (a top down first-order inductive learner)[45] is one of the first and best known ILP systems in the public domain. Foil uses information in a collection of relations to construct existentially quantified logical rules in Prolog. CrossMine [55] is an algorithm proposed to enhance the efficiency of rule-based multi relational classifiers. It uses tuple ID propagation to virtually join tables, and randomly select a subset of dominant instances to overcome the skewness of datasets. ICL (Inductive Classification Logic)[52] is an ILP learning system that learns first order logic formulas from examples which belong to two or more classes. There are two practical barriers that ILP systems may face applied to mining of large datasets.

1. Memory problem: Most ILP-based systems load data into main memory. The program either crash or grind to an effective halt in the case of memory shortage.

2. Time problem: The search space of ILP-based systems are very large as they consider the associations between function/predicate values. Furthermore, heuristic search methods are slow.

2.2.4 Relational Decision Tree

Decision trees are one of the most attractive classifier models. They are fast, easy to learn, and work well with very large datasets having large numbers of variables with different types (continuous, nominal, Boolean, etc.). Decision trees use a divide-and-conquer algorithm to induce the trees. The algorithm partitions a data set recursively at each node. Each leaf of the tree contains a subset of the data that meets the conditions along the path from the root. The goal of the decision-tree learning program is to make these subsets as pure as possible in terms of the mixture of class labels. After tree induction, the pruning stage tries to find a small, high-accuracy tree by deleting uninformative branches and nodes.

Decision trees classify unseen data by mapping instances, which contain a vector of attributes, to one of the class values. They are known to be a good classifier, however, they perform poorly on probability estimations. Inaccurate probability estimation of decision trees are because of decision-tree induction algorithms that focus on maximizing classification accuracy and minimizing tree size. Probability estimation trees have been introduced to estimate the probability of class membership with the same advantage as classification trees (e.g., comprehensibility, accuracy and efficiency in high dimensions and on large data sets).

Provost et al. [43] suggest a number of modifications applying to decision tree induction algorithms demonstrating the improvement in the probability estimates.

RPT (Relational Probability Tree)[24] uses aggregation functions (e.g. Average, Mode, count) to transform the relational data into a single table, then induces the tree by a single-table learner. Furthermore, RPTs are adjusted to deal with relational data characteristics such as auto-correlation and linkage disparity by running randomized tests during their feature selection. Although using several aggregate functions over attributes of links expands the feature space and makes it time-consuming to find the best feature in each node of the tree, tree induction is fairly fast.

TILDE (Top-down induction of first-order logical decision tree) [3] induces a logical decision tree from examples by a divide and conquer algorithm extending C4.5 [44] to relational data. Runtime efficiency is a challenge in Tilde as it potentially tests many clauses to find the best candidates. TILDE uses the existential quantifier to deal with the multi-valued attributes of links for an object.

MRDT (Multi Relational Decision Tree) [23] constructs the decision tree by using *selection graphs* as the nodes of the tree. A selection graph is a directed graph which imposes a set of constraints on incorporating information from several tables. Each node in a selection graph represents a table in the database with a set of conditions on the attributes of the table, and each edge shows that there is at least one record that respects the sets of conditions of the corresponding nodes. Guo et al in [21] speed up the MRDT algorithm by using id propagation to implement a virtual join operation that avoids the cost of physical joins.

FORF (First Order Random Forest)[53] is an ensemble of different decision trees each of which is constructed using a random subset of the feature set. Tests at each node of the trees are first order logic queries. First order random forests are categorized according to different levels of aggregation. FORF-NA is the most simple type with no aggregation function. FORF-SA uses simple aggregation, and FORF-RA employs refinements of aggregate queries. The final decision is made by averaging the results of the trees.

2.2.5 Relational Models with Independence Assumptions

One of the issues that makes multi-relational classification difficult compared to propositional learning is the large number of different types of dependencies that a model may have to consider. A principled way to approach the complexity of correlation types is to

consider model classes with explicitly stated independence assumptions. In the following, we discuss two main categories of models with explicit independence assumptions: relational naive Bayes classifiers and relational logistic regression classifiers.

Relational Naive Bayes Classifiers:

Naive Bayes classifier is well-known for its simplicity and robustness. Single-table naive Bayes classifier assumes that different attributes are independent given the class label. There has been extensive research on multi-relational probabilistic versions of the single-table naive Bayes classifier(NBC) [5, 12, 37, 6].

RNBC is based on the independent value assumption(IVA). To explain the intuition behind the independent value assumption, it is helpful to consider multi-relational classification in terms of multisets: Whereas in single-table classification, a given predictive feature has only one value, in relational classification, a feature may have several different values, corresponding to the number of links between the target object and predictive features entities. For example, if we seek to predict the intelligence of a student given her GPA and the difficulty of the courses she has taken, the feature difficulty has as many values as the number of courses the student has taken. IVA assumes that different values of a feature are independent of each other.

Graph-NB is based on the link independence assumption. It assumes that related links to an object are independent of each other. Furthermore, each attribute, either in the target table or other relationships, is independent of other attributes. Although these two methods have two different approaches to classification, they have the same rule for classification which is obtained by the product taken over attributes of the target entity conditioned on the class label, times another product whose factors are the probabilities of the features of linked entities, conditional on a class label, times the probability of the class label to find the most probable class label.

Data mining models tend to present independence assumptions in terms of database tables and tuples. Heterogeneous Naive Bayes classifier (HNBC) [32] considers different tables independent of each other given the class label. Its main difference with other relational naive Bayes classifiers is that the naive Bayes assumption is only applied between attributes of different tables not within the same table. Therefore, in their model different classifiers can be used for each table to learn dependencies among attributes. The posterior results of the classifiers are combined by a naive Bayes classifier. Manjunath et al. [32] describe

the advantages of independence assumptions for relational classification from a data mining point of view, especially for scalability to large enterprise databases which include the following:

1. Independence assumptions permit information from different tables to be analyzed separately.

2. Different classifiers can be chosen based on the data types.

3. Minimal preprocessing is required to prepare data for analysis; for instance, joins can be computed efficiently as view tables for analysis.

Combing ILP and statistical algorithms is a major direction in relational data classification. A number of models integrate in two steps the naive Bayes learning scheme as the statistical models with the inductive logic programmings. In the first step, a set of first-order features or rules are extracted, and in the second step, these features or rules are combined to compute probabilities according to the naive Bayesian formula. Although these methods can deal with relational databases, most of them require the transformation of tuples from tables into main-memory Prolog facts. 1BC, 1BC2 [12] are first-order Bayesian classifiers working on a set of main-memory Prolog facts, which correspond to items in relational databases. 1BC applies dynamic propositionalization to generate first-order features exhaustively within a given feature bias. 1BC2 learns from structured data by fitting various parametric distributions over data. 1BC2 puts a constraint on the number of literals per clause and larger values cause the system to crash. To address the disadvantages of using Prolog facts as input, Mr-SBC [5] is proposed to adopt an integrated approach in the computation of posterior probabilities which also makes use of first order classification rules. nFOIL [29] is another method that employs the naive Bayes criterion to directly guide its search. nFOIL integrates the feature construction and the statistical learning steps, whereas most models do them consecutively and independently of each other.

Logistic Regression Classifiers:

Logistic regression fits a curve to a set of data by learning weights for different features. It can be viewed as a discriminative version of the generative naive Bayes classifier (NBC) model [38]. For single-table classification, the advantages of logistic regression over simple

NBC have been studied in detail [38], and similar results have been reported for single-relation classification [31]. Popescul and Unger combine logistic regression with an expressive feature generation language based on SQL queries [42].

Another relational logistic regression classifier is *Structured logistic regression* which treats the influence on the class label of the target table and that of linked tables as independent [31]. Two independent logistic regressions are carried out one on the target table and the other on the set of linked tables. Information from links is summarized using aggregate functions. The product of the results determines the class label.

2.3 Summary

In this chapter, we have discussed two main relational data representation models and the main relational data classifiers. In the next chapters, we will propose a relational decision tree and a relational Bayesian net classifier which upgrade their corresponding propositional learners to deal with relational data.

Chapter 3

Decision Forests for Relational Classification

A widely used single-table classifier model are decision trees. In this chapter we propose a method for upgrading decision tree learning algorithms for relational data. We formalize an independence assumption and derive a classification formula. Informally, the assumption states that information from different data tables is independent given the class label. We apply logistic regression to prune irrelevant tables and consider different weights for information from different tables.

3.1 Preliminaries and Notation

Our research combines ideas from relational data mining and traditional Decision tree induction process. We deal directly with relational databases and upgrade decision trees to learn our new model. In this section, we begin by an introduction to relational schema. Then, we review decision trees and probability estimation trees.

3.1.1 Relational Schema

A standard **relational schema** contains a set of tables. We assume that the schema follows an entity-relationship model (ER model) [51, Ch.2.2], so the tables in the relational schema are divided into *entity tables* $E_1, E_2, \ldots$ and *relationship tables* $R_1, R_2, \ldots$ that link entity tables to each other by foreign key constraints. Both entity and relationship tables may

feature descriptive attributes. For the classification task, the table which contains the class label is called the target table and is represented by T. For simplicity, we assume that the target table is an entity table. The symbol t denotes the target object, and $c(t)$ denotes the class label. The non-class attributes of t are collectively denoted as $\boldsymbol{a}(t)$, such that the target table contains a tuple $(\boldsymbol{a}(t), c(t))$ of attribute values. The vector of attribute values in row r of table M_i is denoted by $M_{i,r}$ and the entry in column j of this vector by $M_{i,r,j}$. A **database instance** specifies the tuples contained in the tables of a given database schema.

Pathways and Join Tables. One of the key challenges in multi-relational classification is the need to consider different pathways or table joins through which the target entity may be linked to other entities. Han et al. [6] proposed a graphical way to structure the space of possible pathways (see also [32]). A **Semantic Relationship Graph** (SRG) for a database $\mathcal{D}$ is a directed acyclic graph (DAG) whose nodes are database tables in $\mathcal{D}$ and whose only source (starting point) is the target table. Self-joins and other repeated occurrences of the same table can be handled by duplicating the table [6]. If an edge links two tables in the SRG, then the two tables share at least one primary key. We consider SRG paths of the form $T, R_1, \ldots, R_k, E_k$ that end with an entity table. For each such path, there is a corresponding **path join** $T \bowtie R_1 \cdots \bowtie R_k \bowtie E_k$. The symbol $\bowtie$ shows the **natural join** of two tables which is the set of tuples from their cross product that agree on the values of fields common to both tables [51].

The number of attributes in a valid join may become quite large, therefore, Han et al. use only the attributes of the last table in the path, and the class attribute from the target table, which is called *propagating* the class attribute along the join path [6]. An **extended database** is a database that contains all valid join tables with selecting (projecting) the class attribute and attributes from the last table in each join.

Example. As a running example, we use a university database. The SRG is shown in Figure 3.1. The schema has three entity tables: Student, Course and Professor, and three relationships: *Registration* records *courses* taken by each *student*, *Taughtby* records *courses* taught by each *professor*, and *RA* records research assistantship of students for professors. The class attribute is the *Intelligence* of the *Student*.

To illustrate join tables in an extended database, consider the university database of Figure 3.2 with the class attribute *Intelligence*. In the extended university database valid joins include the following:

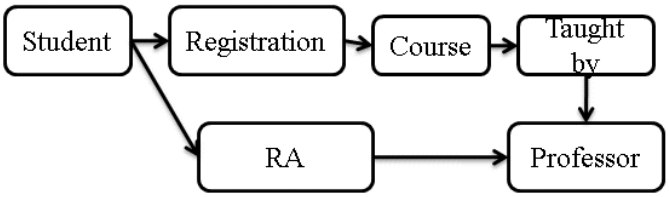
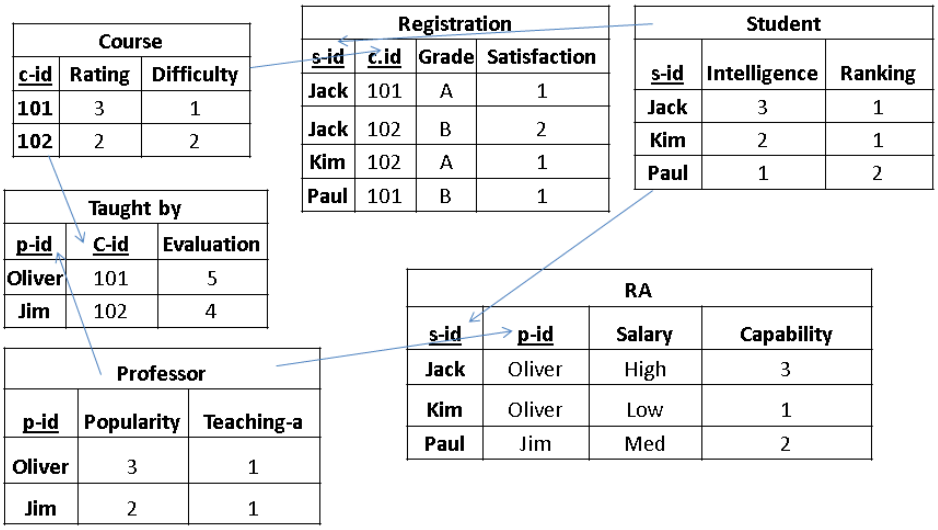


Figure 3.1: Semantic relationship graph for the university schema.



Course

c-id	Rating	Difficulty
101	3	1
102	2	2

Registration

s-id	c.id	Grade	Satisfaction
Jack	101	A	1
Jack	102	B	2
Kim	102	A	1
Paul	101	B	1

Student

s-id	Intelligence	Ranking
Jack	3	1
Kim	2	1
Paul	1	2

Taught by

p-id	C-id	Evaluation
Oliver	101	5
Jim	102	4

RA

s-id	p-id	Salary	Capability
Jack	Oliver	High	3
Kim	Oliver	Low	1
Paul	Jim	Med	2

Professor

p-id	Popularity	Teaching-a
Oliver	3	1
Jim	2	1

Figure 3.2: A small instance for the university database.

- $J_1 = Student \bowtie Registration \bowtie Course$, with the projection of the attributes from *Course*, *Registration*, and the class label from *Student*.
- $J_2 = Student \bowtie RA \bowtie Professor$, with the selection of the attributes from *Professor*, *RA*, and the class label from *Student*.
- $J_3 = Student \bowtie Registration \bowtie Course \bowtie Taughtby \bowtie Professor$, with the selection of the attributes from *Professor*, *Taughtby*, and the class label from *Student*.

Figure 3.3 shows the path tables for the university instance in Figure3.2.

In the next section, we explain how we apply different probability estimation trees to join tables independently to classify relational data.

J1

s-id	p-id	Salary	Capability	Intelligence	Pop	Teach-a
Jack	Oliver	High	3	3	3	1
Kim	Oliver	Low	1	2	3	1
Paul	Jim	Med	2	1	2	1

J_2

s-id	c.id	Grade	Satisfaction	Intelligence	Rating	Diff
Jack	101	A	1	3	3	1
Jack	102	B	2	3	2	2
Kim	102	A	1	2	2	2
Paul	101	B	1	1	3	1

J_3

s-id	c.id	p.id	Evaluation	Intelligence	Pop	Teach-a
Jack	101	Oliver	5	3	3	1
Jack	102	Jim	4	3	2	1
Kim	102	Jim	4	2	2	1
Paul	101	Oliver	5	1	3	1

Figure 3.3: The path join tables with selected coloumns for the university database.

3.2 Simple Decision Forest

In this section, we define two independence assumptions formally and derive a classification formula. Based on the formula, we define a meta logistic regression model for multi-relational classification. Applying the regression to decision trees defines the simple decision forest classification model.

3.2.1 Independence Assumptions

We define and discuss our independence assumptions, then derive a closed-form classification formula.

Definition 1 *Consider an extended database with target table T and join tables $J_1, \ldots, J_m$.*

*(1) The **Table Naive Bayes Assumption** (TNBA) states that the information in different tables is independent given the class label:*

$$P(T, J_1, \ldots, J_m | c(t)) = P(\boldsymbol{a}(t)|c(t)) \prod_{i=1}^{m} P(J_i|c(t))$$

(2) The ***Row Independence Assumption*** *(RIA) states that the information in different rows is independent given the class label, for each table i:*

$$P(J_i|c(t)) = \prod_{r=1}^{rows_i} P(J_{i,r}|c(t))$$

(3) The combined TNBA and RIA together imply:

$$P(T, J_1, \ldots, J_m|c(t)) = P(\boldsymbol{a}(t)|c(t)) \prod_{i=1}^{m} \prod_{r=1}^{rows_i} P(J_{i,r}|c(t))$$

Discussion. Multi-relational NB classifiers like that used by Han et al.[6] add a third assumption: the Column NB Assumption that within each row, the attributes are independent given the class table. This amounts to using the single-table NB classifier to model the distribution $P(c(t)|J_{i,r})$. In contrast, using only the Table NB and Row independence assumptions adds a degree of freedom that allows us to use a classifier other than single-table NB to model the conditional distribution $P(c(t)|J_{i,r})$. Figure 3.4 illustrates the TNB Assumption for the university schema using a Bayesian network [7].

We now discuss the plausibility and limitations of our assumptions. Building decision trees on the extended tables is more natural than learning them on random subsamples of features like FORF [53]. Attributes within tables in real world data are more related than those across tables. Therefore the learned trees convey much meaningful information. However, they cannot model dependencies between attributes across tables.

For the extended DB, since the class attribute is propagated to all join tables, the TNBA assumption requires conditioning on this common information. This is an instance of the general principle that structured objects may become independent if we condition on their shared components.

The RIA assumption says that, given the class label (common part), rows of the link table join are independent of each other. To apply a single table classifier to each extended table, this assumption is required.

The use of the relational assumptions is best viewed as a simplifying approximation to the actual dependencies in the dataset. Like the non-relational NB assumption, the assumption is often true enough to permit accurate predictions of an entity's attributes. Another view is that the assumptions expresse a choice of which correlations are modeled: It permits a model to represent and exploit dependencies between attributes and the class label given the link structure, but not correlations among the links or among the attributes of other

entities. Analogously, the Naive Bayes assumption allows a model to present correlations between features and the class label, but not correlations among the features. In order to investigate the impact of the assumptions empirically, we derive a classification formula from it.

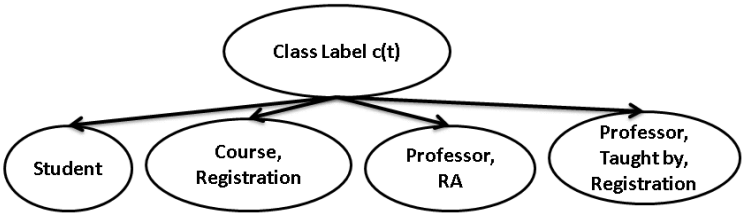


Figure 3.4: An example of the Table Naive Bayes Assumption in the university domain. Attributes of different tables extended by the class label are independent of each other given the class label.

3.2.2 The Decision Forest Model

To derive a classification formula from the assumptions, for simplicity we assume that we have a binary class label, so $c(t) \in \{0, 1\}$, although it can be easily extended to multi-class problems (one versus all). Given a testing example X (all the information in the extended tables) the posterior proportion (odds) is defined as:

$$\frac{P(c(t)=0|X)}{P(c(t)=1|X)} = \frac{P(T, J_1, \ldots, J_m|c(t)=0)P(c(t)=0)}{P(T, J_1, \ldots, J_m|c(t)=1)P(c(t)=1)}$$

The class label is 0 if the posterior odds is larger than 1 and 1 otherwise.

Applying the combined TNBA and RIA Assumption 2(1), we have:

$$= \frac{P(c(t)=0)P(\boldsymbol{a}(t)|c(t)=0)\prod_{i=1}^{m}\prod_{r=1}^{rows_i}P(J_{i,r}|c(t)=0)}{P(c(t)=1)P(\boldsymbol{a}(t)|c(t)=1)\prod_{i=1}^{m}\prod_{r=1}^{rows_i}P(J_{i,r}|c(t)=1)}$$

Substituting all probabilities with posteriors using Bayes' theorem leads to the final classification formula:

$$\begin{aligned}\frac{P(c(t)=0|X)}{P(c(t)=1|X)} = \\ \frac{P(c(t)=0|\boldsymbol{a}(t))}{P(c(t)=1|\boldsymbol{a}(t))}\prod_{j=1}^{m}\prod_{r=1}^{rows_i}\frac{P(c(t)=1)}{P(c(t)=0)}\cdot\frac{P(c(t)=0|J_{i,r})}{P(c(t)=1|J_{i,r})}\end{aligned} \tag{3.1}$$

Using Formula 3.1, any classification algorithm can be employed on individual tables to find the posterior probabilities. We apply decision trees for the following reasons.

1. Decision trees are fast and prune irrelevant nodes.
2. Decision trees can return the **probability** of each value of the class label, not just the most probable class label.
3. There are a number of relational decision trees in the literature, so for the sake of comparison decision trees are a good option.

Furthermore, we use single table Bayesian network to learn a model over each table(cf. 5.2).

Algorithm 1 shows how to compute Formula 3.1.

3.2.3 Weighted Log Linear Simple Decision Forest

Classification formula 3.1 assigns the same weight for the information obtained from different sources, although information from links closer to the target object is typically most important. Furthermore, if an object has a large number of links, the information from links contributes many more terms in the equation rather than the target entity's attributes. Therefore, the class label is largely determined by the information from links. For example if Jack has taken 6 courses, the information of Jack like *ranking* is overwhelmed by information about courses. Given the importance of information from the target table and closer links, we consider a meta regression method that assigns different weights to information from different decision trees. Moreover, we scale information from linked tables by the size of the table. Adding a scale factor is motivated by considering the log-linear version of the formula 3.1:

$$\log(\frac{P(c(t)=0|X)}{P(c(t)=1|X)}) = \log\left(\frac{P(c(t)=0|\boldsymbol{a}(t))}{P(c(t)=1|\boldsymbol{a}(t))}\right) + \sum_{i=1}^{m}\sum_{r=1}^{rows_i}\log\left(\frac{P(c(t)=1)}{P(c(t)=0)}\right) + \log\left(\frac{P(c(t)=0|J_{i,r})}{P(c(t)=1|J_{i,r})}\right)$$

Adding weight parameters w_i and scaling to the table size lead to the final log-linear classification model.

$$w_0 + w_1\log\left(\frac{P(c(t)=0|\boldsymbol{a}(t))}{P(c(t)=1|\boldsymbol{a}(t))}\right) + \sum_{i=1}^{m}\frac{w_i}{rows_i}\sum_{r=1}^{rows_i}\log\left(\frac{P(c(t)=1)}{P(c(t)=0)}\right) + \log\left(\frac{P(c(t)=0|J_{i,r})}{P(c(t)=1|J_{i,r})}\right) \tag{3.2}$$

The weight vector $\vec{w} = (w_0, w_1, \ldots, w_m)$ controls the effect of information from different links in the prediction of the class label. This mitigates the negative effect of Table Naive Bayes Assumption where it does not hold, because the learned weights reflect the relative importance of different links or paths. Adding scaling factor has been motivated in different contexts. Domingos et al. [10]

showed that the addition of the scaling weight factors (like $rows_i$) improves learning weights for Markov Logic Networks. McCallum et al. [46] observed that considering different scaled weights for information from different section of a text improves the results in the text classification problem.

Algorithm 2 shows the classification algorithm that corresponds to Formula 3.2. Firstly, the weighted log of the posterior odds given the information in the target table is computed. Secondly, for each of the link table join and for each row in it, the log of the posterior odds given the attributes of the link divided by the prior probability of the class label is estimated. This measures how much the information from the link changes the prior probability of a class label. Thirdly, the information from each extended table is weighted and scaled by the number of its rows. Finally, these different estimates are added together to predict the class label.

Algorithm 1 Multi-Relational Data Classification

Input:
(1) A new target instance t.
(2) Extended data base tables $J_1, \ldots, J_k$.
(3) Regression weights $\vec{w}$.
(4) Probabilistic Classifier $\mathcal{C}_T$ for the target table, $\mathcal{C}_i$ for each extended table.
Output: A predicted class label

$TP := w_0$
$TP+ = w_1 \cdot (\log(\mathcal{C}_T(c = 0|a(t)) - \log(\mathcal{C}(c = 1|a(t)))$ {Posteriors from the target table}
for all (extended table J_i) **do**
  $LP := 0$
  for each (row $J_{i,r}$ containing the target entity t) **do**
    $LP+ = \log\left(\frac{\mathcal{C}_i(c(t=0)|J_{i,r})\cdot\mathcal{C}_i(c(t)=1)}{\mathcal{C}_i(c(t=1)|J_{i,r})\cdot\mathcal{C}_i(c(t)=0)}\right)$
  end for
  $TP+ = w_i \cdot \frac{1}{rows_i} \cdot LP$
end for
if $(TP > 0)$ **then**
  return 0
else
  return 1
end if

3.3 Learning Simple Decision Forests With Meta Regression

The classifier of Algorithm 1 requires as input three components. We discuss the construction of each of these components in turn.

(1) For constructing the extended database, in our experimental datasets it suffices simply to enumerate the possible valid joins based on SRGs and add the corresponding join tables as views to

Algorithm 2 Multi-Relational Simple Decision Forest Induction and meta regression weight learning

Input: Extended data base tables $J_1, \ldots, J_m$.
Output:
(1) A forest of decision tree: probabilistic Classifier $\mathcal{C}_T$ for target table, $\mathcal{C}_i$ for each extended table.
(2) Regression weights $\vec{w}$.
Call: decision tree learner DT, and logistic regression weight learner LR

Fix a training set for DT learning, and a validation set for LR.
{Start Decision Trees induction} $\mathcal{C}_T :=$ Call DT (target table T).
for each table J_i in D **do**
  $\mathcal{C}_i :=$ Call DT (J_i).
end for
{Start Logistic Meta Regression.} Create matrix M with $m+1$ columns.
for each target object t_k in the validation set with the class label $c(t_k)$ **do**
  $M_{k,0} := c(t_k)$
  $M_{k,1} := \log(\mathcal{C}_T(c=0|a(t_k)) - \log(\mathcal{C}(c=1|a(t_k))$
  for all (extended tables J_i) **do**
    $LP := 0$
    for each (row $J_{i,r}$ containing the target entity t_k) **do**
      $LP+ = \log\left(\frac{\mathcal{C}_i(c(t_k=0)|J_{i,r})\cdot\mathcal{C}_i(c(t_k)=1)}{\mathcal{C}_i(c(t_k=1)|J_{i,r})\cdot\mathcal{C}_i(c(t_k)=0)}\right)$
    end for
    $M_{k,i} := \frac{1}{rows_i}\cdot LP$
  end for
end for
$\vec{w} =$ Call LR(M)

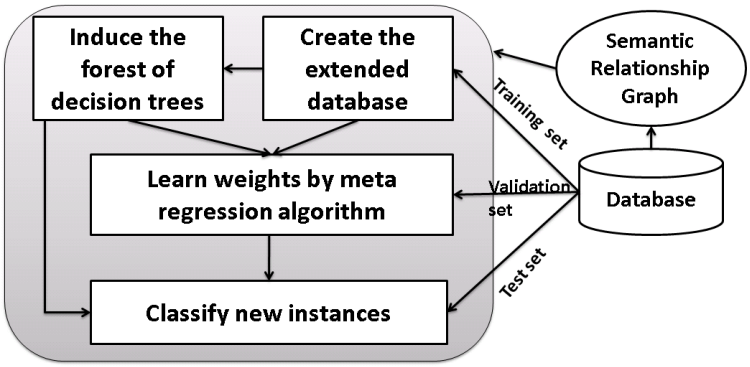


Figure 3.5: The process of classifying relational data using the Decision Forest

the database.

(2) For the classifier $\mathcal{C}_i$ for the join table J_i, any decision tree learner can be applied to J_i. To find probabilities and have probability estimation trees, we use Laplace correction, with no post-pruning algorithm [43].

(3) To learn the meta-regression weights, we apply standard single-table logistic regression as follows. Define

$$b_1 = \log\left(\frac{P(c(t)=0|\boldsymbol{a}(t))}{P(c(t)=1|\boldsymbol{a}(t))}\right)$$

and

$$b_i = \frac{1}{rows_i}\sum_{r=1}^{rows_i}\log\left(\frac{P(c(t)=0|J_{i,r})\cdot P(c(t)=1)}{P(c(t)=1|J_{i,r})\cdot P(c(t)=0)}\right).$$

Then Formula 3.2 can be rewritten as a logistic regression formula (Logit of posterior odds)

$$\log(\frac{P(c=0|X)}{P(c=1|X)}) = w_0 + w_1 b_1 + w_2 b_2 + \ldots + w_m b_m \tag{3.3}$$

Therefore, weights can be learned by an optimized logistic regression over the b_i values. The b_i values for a given target entity can be viewed as *meta features*: They are not part of the original features in the extended data table J_i, but represent a probability estimate that are computed from the features (attributes) in J_i using decision tree models. Since regression is applied to predictions rather than directly to the information from which the predictions are derived, we refer to it as *meta regression.*

Algorithm 2 describes the model learning phase. First, we divide the learning input data into training and validation set. we learn the classifiers on the training set and weights on a different subset, validation set, to avoid overfitting. In lines 1 to 5 different trees on each table in the extended

database are learned. To learn weights, for each instance t_k in the set, feature vectors or independent variables $b_1(t_k), \ldots, b_m(t_k)$ are computed. A matrix with one row for each training instance t_k, one column for the class label $c(t_k)$, and one column for each predictor $b_i(t_k)$ is formed. This matrix is the input for a logistic regression package. We used the Weka simple logistic regression procedure. Figure 3.5 represents the whole process of classifying relational data using the decision forest.

3.4 Summary

Decisions trees are a well-established predictive method for propositional single table data. We have proposed a new way of utilizing them in relational data classification. The basic idea is to independently learn different decision trees for different related tables, and then combine them in a log-linear model to predict class probabilities.

In the next chapter, we will define a weaker assumption to consider dependencies between attributes of different tables as well. We will upgrade Bayesian networks to learn a new relational classifier.

Chapter 4

The Path Independence Bayes Net Classifier

We upgrade Bayesian networks to classify relational data. To approach the complexity of correlation types in relational data we define an independence assumption which is weaker than Table Naive Bayes Assumption (cf., Definition 2) and considers the correlations between attributes from different tables. In this chapter, First we review background material and define our notation. Then we propose our new relational Bayes net classifier.

4.1 Preliminaries and Notation

Our research combines ideas from relational data mining and statistical relational learning (SRL). We deal directly with relational databases and modify probabilistic relational models (PRMs) to define and learn our new model. In this section, we begin by an introduction to relational schema. Then, since Bayesian networks are the foundation of PRMs, we review them before introducing PRMs.

4.1.1 Relational Schema

We follow the notation explained in the previous chapter and for each database D we create a Semantic Relationship Graph (SRG) to structure the space of possible join pathways. However, instead of considering the attributes of the last entity, of the last relationship table in the path, and the class attribute from the target table in each valid join, we use all the attributes. Therefore, an extended database is a database that contains all valid join tables which we simply refer to as **path tables** with all attributes. We introduce some special notation to refer to the value of columns of a

path table:

- K denotes the key columns in a join table.
- R denotes the fields corresponding to attributes of *relationships* that appear in the join.
- E denotes the fields corresponding to attributes of *entities* that appear in the join.
- T denotes the fields corresponding to attributes of the *target table* that appear in the join.

These symbols also index the table entries in a given set of columns as follows, for row r and join table J_i:

- **Key field values** are denoted as $J_{[i,r,K]}$.
- **Link attribute values** are denoted as $J_{[i,r,R]}$.
- **Entity attribute values** are denoted as $J_{[i,r,E]}$.
- **Target table attributes** are denoted as $J_{[i,r,T]}$.

The union notation $\cup$ denotes unions of columns, so for instance $K \cup R$ refers to the key fields together with the relationship attribute fields. Therefore, for each row r in path table J_i that includes the target entity t in its key fields, the tuple of values in the row satisfies

$$J_{i,r} = J_{[i,r,K\cup R\cup E\cup T]} = J_{[i,r,K\cup R\cup E]}, \boldsymbol{a}(t), c(t).$$

Example. We use the university schema explained in Chapter 3. In the extended university database based on the SRG in Figure 3.1 valid joins include the following:

- $J_1 = \mathit{Student} \bowtie \mathit{RA} \bowtie \mathit{Professor}$.
- $J_2 = \mathit{Student} \bowtie \mathit{Registration} \bowtie \mathit{Course}$.
- $J_3 = \mathit{Student} \bowtie \mathit{Registration} \bowtie \mathit{Course} \bowtie \mathit{Taughtby} \bowtie \mathit{Professor}$.

Figure 4.1 shows the path tables and K, R, E, T division of their columns for the university instance in Figure 3.2. Table 4.1 illustrates it for both J_2 and J_3.

If the target entity is $t =$ jack, then we have $\boldsymbol{a}(t)=$ {Ranking(jack)}, and $c(t) =$ {Intelligence(jack)}.

In the next section, we use the background and notation explained to define the Path Independence classifier.

4.2 The Path Independence Classifier

We define and discuss our main independence assumption, then derive a closed-form classification formula and describe a structure learning algorithm.

J1

K		R		T		E	
s-id	**p-id**	**Salary**	**Capability**	**Intelligence**	**Ranking**	**Pop**	**Teach-a**
Jack	Oliver	High	3	3	1	3	1
Kim	Oliver	Low	1	2	1	3	1
Paul	Jim	Med	2	1	2	2	1

J_2

K		R		T		E	
s-id	**c.id**	**Grade**	**Satisfaction**	**Intelligence**	**Ranking**	**Rating**	**Diff**
Jack	101	A	1	3	1	3	1
Jack	102	B	2	3	1	2	2
Kim	102	A	1	2	1	2	2
Paul	101	B	1	1	2	3	1

J_3

K			R			T		E			
s-id	c.id	p.id	Grade	Satisfaction	Evaluation	Intelligence	Ranking	Rating	Diff	Pop	Teach-a
Jack	101	Oliver	A	1	5	3	1	3	1	3	1
Jack	102	Jim	B	2	4	3	1	2	2	2	1
Kim	102	Jim	A	1	4	2	1	2	2	2	1
Paul	101	Oliver	B	1	5	1	2	3	1	3	1

Figure 4.1: The path join tables for the university database.

4.2.1 Definition and Classification Formula

Consider an extended database with target attributes T, entity tables' attributes $E_1, \ldots, E_k = \mathbf{E}$, and path tables $J_1, \ldots, J_m = \mathbf{J}$. We view a table M_i as a conjunction of the information pieces in it, that is, as a conjunction of value assignments $\wedge_{r,j}(M_{i,r,j} = value)$. For key fields, the meaning of the statement that $M_{i,r,K} = ids$ is that the entities denoted by *ids* are linked together in the pathway that defines the table join. Thus in the definitions below $M_{i,r,K}$ denotes a Boolean random variable. For instance, in the join table J_2 in figure 4.1, the event $J_{2,1,K} = (jack, 101)$ is true if Jack is registered in course 101.

Definition 2 *The Path Independence Assumption (PIA) states that different paths, represented in different rows in join tables, are conditionally independent given the attributes of the entities in the path:*

$$P(\mathbf{J}|E,T) = \prod_{i=1}^{m} \prod_{r=1}^{rows_i} P(J_{[i,r,K\cup R]}|J_{[i,r,E\cup T]}) \tag{4.1}$$

The symbol $rows_i$ is the number of rows in join table J_i. To derive an LBC formula under the path independence assumption, we consider the distribution of the attributes given the existing

$X =$	K	R	E	T
$J_{2,1,X}$	(jack,101)	(A,1)	(3,1)	(3,1)
$J_{3,1,X}$	(jack,101,oliver)	(A,1,5)	(3,1,3,1)	(3,1)

Table 4.1: To illustrate our notation for the value tuples associated with different columns in a join table.

paths. The effect of conditioning on existing paths (primary keys) is to neglect pathways that involve "missing links". This restriction is common in ILP and SRL [6, 14]. The goal is to find the most probable class label:

$$c = \operatorname*{argmax}_c P(\mathbf{J}, \mathbf{E}, T).$$

From the definition of Path Independence (4.1) the classification problem is equivalent to the maximization.

$$\operatorname*{argmax}_c P(c(t)|\boldsymbol{a}(t)) \cdot \prod_{i=1}^{m} \prod_{r=1}^{rows_i} P(J_{[i,r,K\cup R]}|J_{[i,r,E]}, c(t), \boldsymbol{a}(t)) \tag{4.2}$$

Using Bayes' theorem we have

$$P(J_{[i,r,K\cup R]}|J_{[i,r,E]}, c(t), \boldsymbol{a}(t)) \propto \frac{P(c(t)|J_{[i,r,K\cup R\cup E]}, \boldsymbol{a}(t))}{P(c(t)|J_{[i,r,E]}, \boldsymbol{a}(t))} \tag{4.3}$$

where we have omitted a factor that does not depend on $c(t)$.

We now use the independence fact

$$P(c(t)|J_{[i,r,E]}, \boldsymbol{a}(t)) = P(c(t)|\boldsymbol{a}(t)) \tag{4.4}$$

which holds since attributes of entities distinct from the target entity are independent of the target class, unless we condition on the existence of a path. Now substituting Equation (4.4) into Equation (4.3) and applying the result to Equation (4.2) yields:

$$\operatorname*{argmax}_c P(c(t)|\boldsymbol{a}(t)) \cdot \prod_{i=1}^{m} \prod_{r=1}^{rows_i} \frac{P(c(t)|J_{[i,r,K\cup R\cup E]}, \boldsymbol{a}(t))}{P(c(t)|\boldsymbol{a}(t))} \tag{4.5}$$

The formula can be read as follows. For each path join table and for each row in it, compute the probability of the class label given the row entries, and divide the result by the class posterior in the target table. Thus the formula measures the additional information gained by considering each path, relative to considering only the target table.

4.2.2 Interpretation and Discussion

We provide a graphical and a tabular interpretation of the PI assumption.

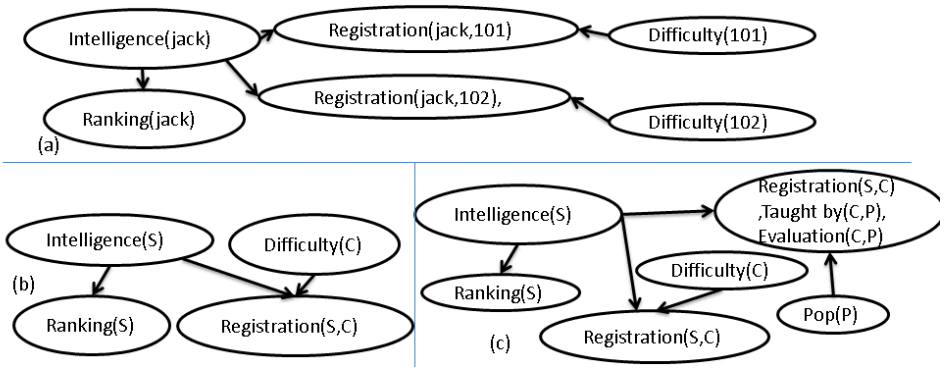


Figure 4.2: BN representation of Path Independence. (a): a ground network of target entity jack with two courses. (b):1st-order template for (a). (c): template with a path node containing 2 relationships.

Graphical Interpretation. Figure 4.2 provides a BN interpretation of the PI assumption. Subfigures (b) and (c) uses nodes that refer to columns, links, or paths, using generic 1st-order variables. The former is a BN for join table J_2 and the latter is a BN for J_3, the join table with two relationships. Subfigure (a) illustrates how this pattern is instantiated at the level of specific entities, using the standard grounding method for relational BNs [40, 20]. A BN G satisfies path independence if each path node $(J_{[i,r,R]}, J_{[i,r,keys]})$ is a sink node (with no children) and each of its parents is an attribute node from $c(t)$ or $a(t)$ or $J_{[i,r,E]}$. The path independence constraint is the natural BN representation of the following independence facts: attributes of different entities are unconditionally d-separated, hence independent, but *given a relationship between the entities*, they become d-connected, hence potentially dependent. For instance, the difficulty of a course and the intelligence of a student become relevant to each other only when the student has registered in the course.

Tabular Interpretation. Join tables have common fields from common entities, which leads to dependencies between rows, so the path independence assumption requires conditioning on this common information. This is an instance of the general principle that structured objects may become independent if we condition on their shared components. For instance, the grade of Jack in course 101 is assumed to be independent of the grade of Jack in course 102 given the attributes of the courses (E) and those of Jack (T), which are common to the first two rows in the J_2 table of Figure 4.1.

Impact of Assumption. In a given dataset, the assumption may not be entirely but only approximately true. For the PIA, the grades of Jack in different courses are normally (unconditionally) correlated, and we would expect some correlation to persist even conditional on the attributes of Jack and the courses. Likewise, links from different tables may well be correlated as well. The use of the

path independence assumption is best viewed as a simplifying approximation to the actual dependencies in the dataset. Like the non-relational Naive Bayes assumption, the assumption is often true enough to permit accurate predictions of an entity's attributes. The Naive Bayes assumption allows a model to present correlations between features and the class label, but not correlations among the features. Analogously, the path independence assumption represents correlations between attributes and the class label given the path structure, but not correlations among the paths. This is demonstrated by the PI formula (4.5), which incorporates dependencies of the class label on attributes of related entities and links, but not correlations among links.

4.2.3 Learning

To learn a PI model, the basic idea is to apply a standard single-table BN learner repeatedly to join tables, where learning for larger joins is constrained by the results of learning for subjoins. In other words, we perform a level-wise search through the table join lattice. The learning algorithm is as follows.

1. Learn a single table BN structure for the attributes in the target table.
2. Learn a BN for the attributes in each of the join tables that involve a single relationship, with two constraints: (i) edges present or absent in the target table BN are also present or absent in the join table BN. (ii) attributes of relationships are sink nodes without children
3. For each join table with a relationship chain of length two, apply the single table BN learner with the same constraints as before, with the edges learned in phase 2 in place of those from phase 1.
4. For each edge $A \rightarrow B$ that was added by the BN learner when analyzing join table J_i, add edges from A and from B to a corresponding path variable and delete the direct edge.

The output of the algorithm is a single multi-relational BN model that satisfies the constraint that relationship/path nodes are sinks without children, which corresponds to the Path Independence Assumption (see Section 4.2.2). After learning a BN for a given data set, we use the Markov blanket of the class attribute for classification (neighbors and spouses).

The PI algorithm is an adaptation of the recent learn-and-join (LAJ) Bayes net algorithm for multi-relational data; please see [27] for further discussion. Khosravi *et al.* provide a complexity analysis that shows that although the table join sizes increase, the edge inheritance constraints essentially keep the model search constant in each application of the BN learner.

Example. We illustrate the algorithm with reference to Figure 4.2 for the university database. We do not consider join table J_1 in this example for simplicity.

-Phase (1) finds the edge from $\mathit{Intelligence}(S) \rightarrow \mathit{Ranking}(S)$.

-Phase (2) applies the BN learner to join table J_2 (Figure 4.1) with that edge fixed, and finds an edge $\mathit{Intelligence}(S) \rightarrow \mathit{Difficulty}(C)$.

-In phase (3), the join table J_3 is analyzed, with the previous edges inherited as constraints, finding two edges $\mathit{Intelligence}(S) \rightarrow \mathit{Evaluation}(P)$ and $\mathit{Pop}(P) \rightarrow \mathit{Evaluation}(P)$.

-Phase (4) adds edges to link/path nodes, resulting in the final BN shown in Figure 4.2(c).

4.3 Summary

In this chapter, we have reviewed the definitions of relational schema. Then we have introduced our novel multi-relational Bayes net classifier based on the path independence assumption and have proposed an efficient BN model learning algorithm. In the next chapter, we will compare our proposed path independence Bayes net classifier with the simple decision forest and a variety of relational classifiers.

Chapter 5

Evaluation

In this chapter, we compare different configurations of our proposed models with various relational classification models. We describe the datasets, basic setting of our experiments, and results in different evaluation metrics.

5.1 Datasets

We used three standard relational datasets.

Financial Dataset. This dataset was used in PKDD CUP 1999. It contains 8 tables. *Loan* is the target table with 682 instances (606 of loans are successful). The Semantic relationship graph of this dataset is depicted in figure 5.1. Since 86% of the examples are positive, the data distribution is quite skewed. We followed the CrossMine [55], and Graph-NB [6] modification and randomly selected 324 positive instances, and all the negative loans to make the numbers of positive tuples and negative tuples more balanced. The number of join tables in the extended database was 8.

Hepatitis Database. This dataset is a modified version of the PKDD'02 Discovery Challenge database. We followed the modification of [13]. *Biopsy* is the target table with 206 instances of Hepatitis B, and 484 cases of Hepatitis C. The *inhospital* table was modified such that we put each unique test in a column and all tests for a patient on a given year in a single tuple. Removing tests with null values, 10 different tests as the table descriptive attributes remained. The Semantic relationship graph of this dataset is depicted in figure 5.2. The number of join tables in the extended database was 4.

Mondial Database. This dataset contains data from multiple geographical web data sources [33]. We predict the religion of a country as Christian (positive) with 114 instances vs. all other

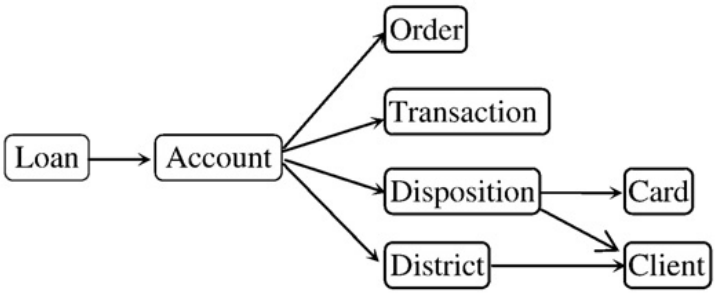


Figure 5.1: Semantic Relationship Graph for the Financial Dataset.

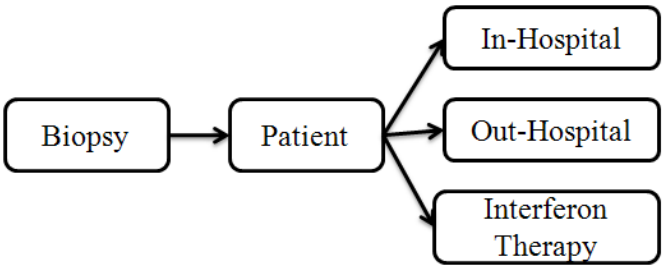


Figure 5.2: Semantic Relationship Graph for the Hepatitis Dataset.

religions with 71 instances. We followed the modification of [48]. To make the classification task more challenging for evaluating learners, we use a subset of the tables and features. Border is a relationship between Country and Country. To create a semantic relationship graph for this database, we did decyclization by making duplications. The semantic relationship graph that we used for this dataset is depicted in Figure 5.3. The number of join tables in the extended database was 5.

5.2 Experimental Setting, Systems, and Performance Metrics.

All experiments were done on a Pentium 4 CPU 2.8Ghz and 3GB of RAM system.

Researchers have proposed several ways to construct an extended database [6, 32]. We adopted the simplest approach, which is to enumerate all possible join paths in a data pre-processing step. If self-joins are possible, this requires a user-defined bound on the maximum join length [6, Sec.3].

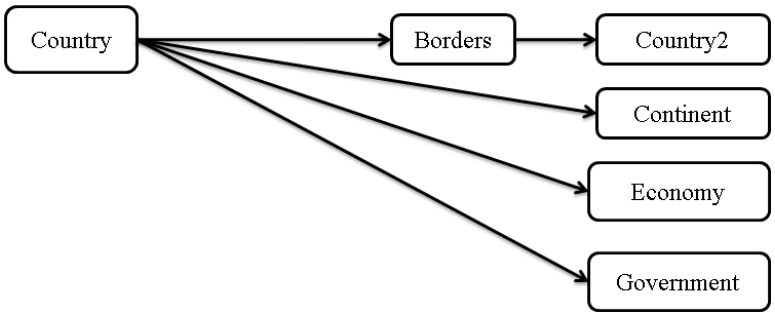


Figure 5.3: Semantic Relationship Graph for the Mondial Dataset.

Because of foreign-key constraints, the number of valid joins is typically much smaller than the total number of possible joins.

We run the experiments on three variations of our Simple Decision Forest, the Table Naive Bayes net Classifier, and the Path Independence Bayesian net classifier.

1. Normalized Decision Forest: weighted log linear decision forest with scaling factors (formula 3.2).
2. Unnormalized Decision Forest: weighted log linear decision forest with no scaled weights (formula 3.2 with $rows_i = 1$).
3. Naive Decision Forest: decision forest with no weights for different trees (Formula 3.1).
4. TNBC: Table Naive Bayes net Classifier that learns a Bayes net for each table and combine the results using Formula 3.1
5. PIBC: Path Independence Bayesian net Classifier.

The implementation of the Simple Decision Forest used many of the procedures of Weka which is a data mining software written in Java [15]. We used J48 as a propositional decision tree learner on each join table. J48 implements the C4.5 decision tree algorithm. We used the probability estimation tree setting that turns off pruning and applies the Laplace correction [43]. We weight the posterior probabilities computed by decision trees by the simple logistic procedure of Weka.

To learn the structure under the Path Independence Assumption, for a single table Bayes net learner we apply the GES search algorithm [8]. We use a generative learner because the Naive Bayes classifier is a generative model, so our experiments avoid conflating the impact of different independence assumptions with the impact of generative vs. discriminative training. For parameter learning, we use the maximum likelihood estimates (database frequencies).

For the sake of comparison of PIBC with other relational Bayesian net classifiers, we also apply a single table Bayes net learner instead of decision tree learner to the Naive Decision Forest Classifier

and derive TNBC. TNBC is a classifier that assumes independency between different join tables and learns its model by learning a Bayesian net classifier for each table.

We compared our proposed methods with the following Multi-relational classifiers:

- TILDE : Top-down Induction of Logical Decision Trees [3].
- FORF-NA: First Order Random Forest with No Aggregates [4].
- Graph-NB: multi-relational naive Bayes classifier [30].
- MLN: Markov Logic Network [10]

TILDE and FORF-NA are included in the ACE data mining system [2]. We run TILDE with the default setting. For FORF-NA we used the out-of-bag setting, chose 25% of features to consider for testing in each node, and fixed the number of trees in the forest to 33. Vens *et al.* report that this setting leads to the most efficient results in terms of accuracy and running time [4]. We implemented Graph-NB using the single table naive Bayes Classifier procedure of Weka as its code is not available online. We used MLN with learn-and-join (LAJ) structure learning algorithm [27], discriminative parameter learning [22], and MC-SAT inference algorithm [41] implemented in Alchemy package. We used the LAJ algorithm because (1) its predictive accuracy outperforms other MLN structure learning algorithms, and (2) it is the only current MLN learning algorithm that scales to the benchmark datasets [27].

To evaluate classification performance, we used the following metrics:

- Run time: induction time of the model.
- Accuracy: percentage of correctly classified instances.
- AUC: the area under the ROC curve.
- F-measure: the weighted harmonic mean of precision and recall.

We evaluated predictive metrics with ten-fold cross-validation. For Simple Decision Forests (Normalized and unnormalized Decision Forests), in each run we learn the decision tree on eight random folds of data, learn the weights of logistic regression on one fold (validation set), and test the model on the remaining fold. For the Naive decision forest, Table Naive Bayes net Classifier, and Path Independence Bayesian net classifier, in each run we learn the model on a random 9-fold and test the model on the remaining fold. All results reported in this section are averages over 10 random test fold.

5.3 Results

We discuss run times for learning, then predictive performance.

5.3.1 Learning Times

Table 5.1 reports the induction times of different relational classifiers on the three datasets. It demonstrates the dramatic improvement in the induction time of Simple Decision Forest compared to the other relational decision tree learners (TILDE and FORF-NA): The systems based on independence assumptions are faster than systems searching the hypothesis to find relevant rules. Furthermore, PIBC is much faster than MLN while they roughly have the same structure learning algorithm.

Algorithm	Financial	Hepatitis	Mondial
Normalized DF	2.3	1.1	0.26
Unnormalized DF	2.3	1.1	0.26
Naive DF	1.4	0.54	0.25
TNBC	1.5	0.32	0.20
PIBC	28.31	7.43	0.28
Graph-NB	0.8	0.21	0.18
TILDE	2429	853	0.30
FORF-NA	54006	10515	7.07
MLN	NT	39.02	5.44

Table 5.1: Average Induction time of different algorithms in sec

For Normalized and Unnormalized Decision Forests, the runtime is basically the sum of the runtimes of the Naive Decision Forest learner and the logistic regression. Normalized and Unnormalized Decision Forest are just different in scaling factors so they have the same induction time.

The structure learning algorithm of PIBC allows for the most complex cross-table dependencies and therefore takes the most time among our proposed models, but it is still very fast compared to other models even on fairly large and complex data sets like *Financial*. Our Simple Decision Forest learner and TNBC considers dependencies just within tables; therefore, they have shorter induction times compared to PIBC, but longer induction times compared to the Multi-relational Naive Bayes classifier (Graph-NB). The runtime of the Graph-NB is given by the sum of the runtimes of applying a single-table NB learner to join tables.

For TILDE we report the decision tree induction time. For FORF-NA runtime is the time needed to induce 33 different decision trees. TILDE and FORF-NA search a large feature space to find the best attribute in each node of the tree which is a time-consuming task.

For MLN, the runtime is calculated by adding the structure learning time of the Learn And Join with the parameter learning time of the alchemy package. NT stands for "Not Terminated". The structure algorithm (LAJ) is fast, whereas the parameter learning takes too much time.

Predictive Performance.

To compare different models easily we report the results of the decision trees and graphical model classifiers in separate tables. Table 5.2 and 5.3 show the Accuracy, AUC, and F-measure of the different classifiers on the Financial dataset, table 5.4 and and 5.5 show these measures on the Hepatitis dataset, and table 5.6 and 5.7 represent them on the Mondial dataset.

We make the following observations by comparing different Decision trees results in Tables 5.2,5.4,5.6.

- Overall, the Normalized Decision Forest achieves the best classification performance using regression weights with scaling.
- Comparing Naive Decision Forest with Naive Bayes net classifier, taking into account dependencies between attributes of each table is beneficial.
- Using weights for different trees to decrease or increase the effect of different links tended to improve the results.
- Learning different trees on features of each table instead of learning on randomly selected subset of features tended to enhance the performance.

Comparing the results of different graphical models on the three datasets in Tables 5.3,5.5,5.7, we observe that:

- BN classifier with the *Path Independence assumption* achieves the best classification performance compared to other relational graphical models.
- Comparing PIBC with TNBC, using fewer(weaker) independence assumptions, accuracy tends to be better.
- Unlike single table classifiers, in relational data Bayes Net classifier always outperforms naive Bayes net classifier.

The poor classification performance of the MLN is an example of how directional information can improve both parameter learning and inference. Overall, our results provide evidence that the path independence model makes an attractive trade-off between improving classification accuracy with a modest increase in learning time. It effectively uses BN model selection methods to prune classification-irrelevant tables and attributes. Therefore it does not require as much effort with additional pruning methods as multi-relational NB classifiers [5, 12, 37, 6].

In Figures 5.4,5.5,5.6 we compare the performance of our proposed methods against each other. Results show that

- Normalized Decision Forest have the best performance among the multi-relational classification models.

Algorithm	Accuracy	AUC	F-measure
Normalized Decision Forest	**92%**	**0.88**	**0.89**
Unnormalized Decision Forest	87%	0.85	0.84
Naive Decision Forest	91%	0.85	**0.89**
TILDE	89%	0.69	0.88
FORF-NA	89%	0.75	0.87

Table 5.2: Performance of different Decision Tree classifiers on the Financial dataset

Algorithm	Accuracy	AUC	F-measure
PIBC	91%	0.85	0.89
TNBC	89%	0.85	0.87
Graph-NB	81%	0.82	0.79
MLN	NT	NT	NT

Table 5.3: Performance of different graphical model classifiers on the Financial dataset

Algorithm	Accuracy	AUC	F-measure
Normalized Decision Forest	**84%**	**0.88**	**0.79**
Unnormalized Decision Forest	**84**%	0.86	**0.79**
Naive Decision Forest	80%	0.80	0.75
TILDE	61%	0.61	0.59
FORF-NA	63%	0.64	0.61

Table 5.4: Performance of different Decision Tree classifiers on the Hepatitis dataset

Algorithm	Accuracy	AUC	F-measure
PIBC	80%	0.84	0.70
TNBC	76%	0.80	0.69
Graph-NB	75%	0.79	0.68
MLN	77%	0.78	0.68

Table 5.5: Performance of different graphical model classifiers on the Hepatitis dataset

Algorithm	Accuracy	AUC	F-measure
Normalized Decision Forest	**84%**	0.85	**0.83**
Unnormalized Decision Forest	**84%**	**0.86**	0.82
Naive Decision Forest	83%	**0.86**	0.81
TILDE	71%	0.75	0.78
FORF-NA	71%	0.79	0.77

Table 5.6: Performance of different Decision Tree classifiers on the Mondial dataset

Algorithm	Accuracy	AUC	F-measure
PIBC	83%	0.85	0.82
TNBC	80%	0.83	0.77
Graph-NB	73%	0.74	0.75
MLN	76%	0.82	0.75

Table 5.7: Performance of different graphical model classifiers on the Mondial dataset

- Naive Decision Forest outperforms TNBC in all the metrics introduced. Naive Decision Forest uses Decision tree and TNBC uses Bayesian network to learn a model for each join table; however, they apply the same classification formula to combine the results of different classifiers.
- Naive Decision Forest and PIBC have the same accuracy on the three datasets, whereas for the other two metrics none of them is superior on the all datasets.
- Overall, meta regression algorithm looks more effective than using weaker assumptions(i.e., using PI assumption rather than the Table Naive Bayes Assumption).

To show how linear regression assigns weights to the information from different tables, the weights learned by Normalized and Unnormalized decision forests for the target table and each extended table of each dataset are listed in tables 5.8,5.9.

There are two extended join tables involving the Client relation (w_8 and w_9). This reflects the fact that in the SRG of Financial there are two pathways to the Client table (see Figure 5.1) that correspond to two different join tables.

The 0 weights shown in the tables demonstrate the ability of regression to prune uninformative tables. Also, the fact that the nonzero weights are far from uniform shows that regression learns an importance ranking of the information from different tables.

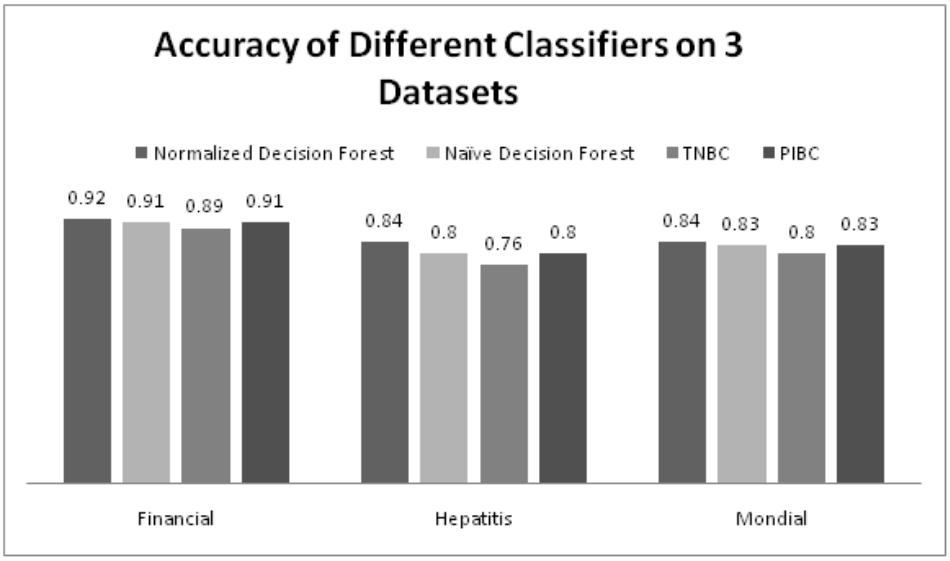


Figure 5.4: Accuracy of our proposed models on the 3 datasets

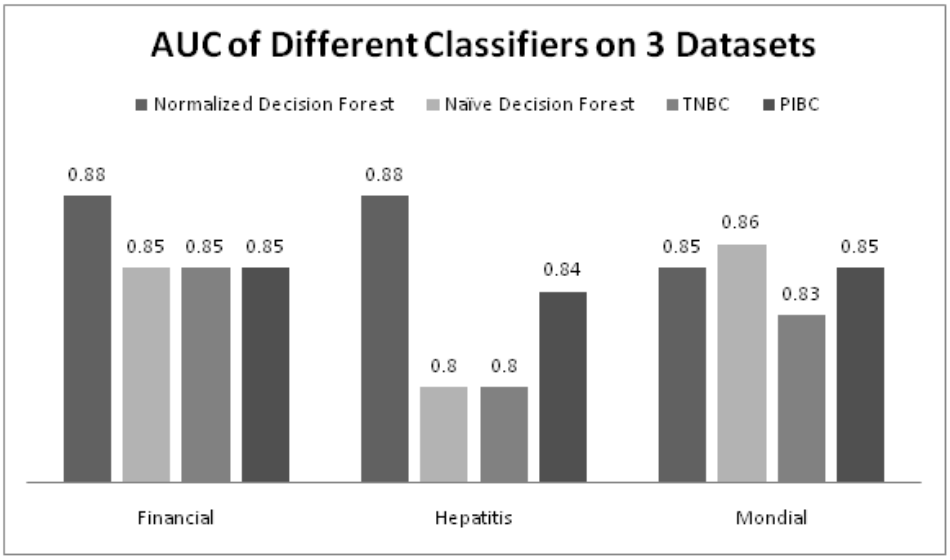


Figure 5.5: AUC of our proposed models on the 3 datasets

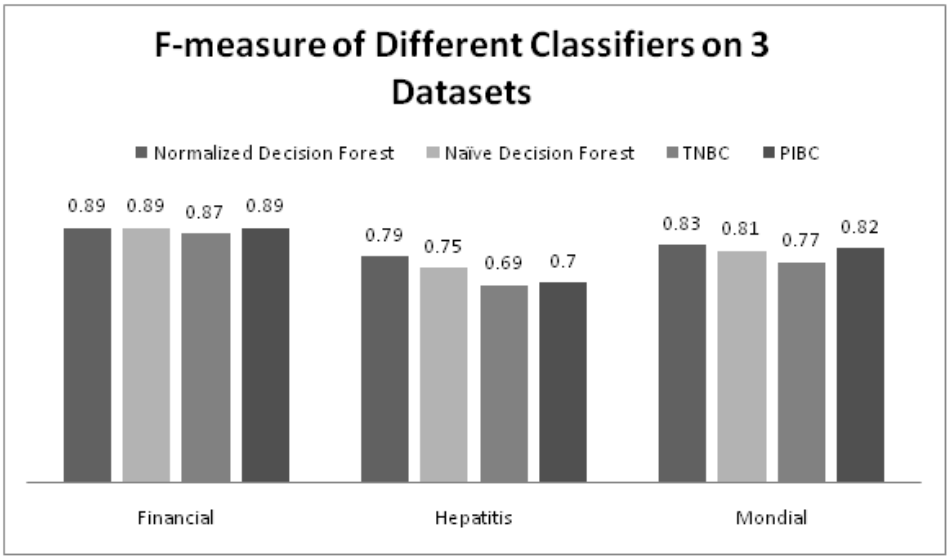


Figure 5.6: F-measure of our proposed models on the 3 datasets

Weight	Financial	Hepatitis	Mondial
w_0	$w_0 = -0.19$	$w_0 = 0.29$	$w_0 = 0$
w_1	$w_{Loan} = 0.52$	$w_{Biopsy} = 0.3$	$w_{Country} = 0.94$
w_2	$w_{Account} = 0$	$w_{Patient} = 0.2$	$w_{Border} = 0$
w_3	$w_{Order} = 0$	$w_{In-Hosp} = 0.9$	$w_{Country2} = 1.43$
w_4	$w_{Trans} = 0.2$	$w_{Out-Hosp} = 0$	$w_{Continent} = 1.23$
w_5	$w_{Disp} = 0.02$	$w_{Inerferon} = 0.3$	$w_{Economy} = 0.86$
w_6	$w_{District} = 0$	-	$w_{Gov} = 0.70$
w_7	$w_{Card} = 0$	-	-
w_8	$w_{ClientDist} = 0$	-	-
w_9	$w_{ClientDisp} = 0$	-	-

Table 5.8: Weights learned by Normalized Decision Forest

Weight	Financial	Hepatitis	Mondial
w_0	$w_0 = -0.34$	$w_0 = 0.97$	$w_0 = 0$
w_1	$w_{Loan} = 0.38$	$w_{Biopsy} = 0.76$	$w_{Country} = 0.87$
w_2	$w_{Account} = 0$	$w_{Patient} = 0.4$	$w_{Border} = 0.3$
w_3	$w_{Order} = 0.34$	$w_{In-Hosp} = 0.22$	$w_{Country2} = 1.09$
w_4	$w_{Trans} = 0.2$	$w_{Out-Hosp} = 0.03$	$w_{Continent} = 1.11$
w_5	$w_{Disp} = 0$	$w_{Interferon} = 0.91$	$w_{Economy} = 0.79$
w_6	$w_{District} = 0$	-	$w_{Gov} = 0.6$
w_7	$w_{Card} = 0$	-	-
w_8	$w_{ClientDist} = 0$	-	-
w_9	$w_{ClientDisp} = 0$	-	-

Table 5.9: Weights learned by Unnormalized Decision Forest

Chapter 6

Summary and Future Work

A goal of relational classification is to make predictions that utilize information not only about the target table but also about related objects. The large number of different dependencies of different types that are potentially relevant for link-based prediction are a challenge for model selection algorithms. In this thesis, we propose two new relational classifiers by upgrading decision tree and Bayesian network classifiers.

In the first model, we independently learn different decision trees for different related tables, and then combine them in a log-linear model to predict class probabilities. The log-linear model is derived from two explicitly stated independence assumptions. Features that distinguish this method from other relational decision tree learners include the following. (1) Aggregation functions are not used, which avoids some information loss and allows for efficient learning. (2) Information from all links is considered in classification; we do not use the existential quantifier.

While a search for informative aggregate features is computationally expensive, when it succeeds, the new aggregate features can substantially increase the predictive accuracy (e.g., [13, 53]). A promising direction for future work is therefore to combine our approach with propositionalization techniques. There are several possibilities for a combined hybrid approach. (1) Once good aggregate features are found, they can be treated like other features and used with meta-regression. (2) A simple decision forest is fast to learn and can establish a strong baseline for evaluating the information gain due to a candidate aggregate feature. (3) The meta regression weights can be used to quickly prune uninformative join tables with 0 or small weights, which allows the search for aggregate features to focus on the most relevant link paths.

The second model is a novel classifier based on the *path independence assumption* saying that different paths from the target entity to related entities are independent. We explained an efficient Bayesian network model learning algorithm and derived closed-form classification formulas for our model. It is promising to explore combining discriminative Bayes net learning algorithms with the

path independence assumption.

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] H. Blockeel, L. Dehaspe, J. Ramon, J. Struyf, A. Van Assche, C. Vens, and D. Fierens. *The ACE Data Mining System: Users Manual*. http://dtai.cs.kuleuven.be/ACE/doc/ACEuser-1.2.16.pdf, 2009.

[3] Hendrik Blockeel and Luc De Raedt. Top-down induction of first-order logical decision trees. *Artificial Intelligence*, 101(1-2):285–297, 1998.

[4] H. Blockeel C. Vens, A. Van Assche and S. Dzeroski. First order random forests with complex aggregates. In *Proceedings of the 14th International Conference on Inductive Logic Programming*, 2004.

[5] Michelangelo Ceci, Annalisa Appice, and Donato Malerba. Mr-sbc: A multi-relational naive Bayes classifier. In *PKDD*, pages 95–106, 2003.

[6] Hailiang Chen, Hongyan Liu, Jiawei Han, and Xiaoxin Yin. Exploring optimization of semantic relationship graph for multi-relational Bayesian classification. *Decision Support Systems*, July 2009.

[7] J. Cheng and R. Greiner. Learning Bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137:43–90, 2002.

[8] David Maxwell Chickering and Christopher Meek. Finding optimal Bayesian networks. In *UAI*, pages 94–102, 2002.

[9] Luc de Raedt. *Logical and Relational Learning*. Cognitive Technologies. Springer, 2008.

[10] Pedro Domingos and Matthew Richardson. Markov logic: A unifying framework for statistical relational learning. In *Introduction to Statistical Relational Learning* [19].

[11] Daan Fierens, Hendrik Blockeel, Maurice Bruynooghe, and Jan Ramon. *Logical Bayesian Networks and Their Relation to Other Probabilistic Logical Models*, pages 121–135. 2005.

[12] Peter A. Flach and Nicolas Lachiche. Naive Bayesian classification of structured data. *Mach. Learn.*, 57(3):233–269, 2004.

[13] Richard Frank, Flavia Moser, and Martin Ester. A method for multi-relational classification using single and multi-feature aggregation functions. In *proceeding of Pkdd*, 2007.

[14] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *In IJCAI*, pages 1300–1309. Springer-Verlag, 1999.

[15] Stephen R. Garner. Weka: The waikato environment for knowledge analysis. In *In Proc. of the New Zealand Computer Science Research Students Conference*, pages 57–64, 1995.

[16] Dan Geiger, Thomas Verma, and Judea Pearl. d-separation: From theorems to algorithms. In *UAI '89: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pages 139–148, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co.

[17] Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Benjamin Taskar. Probabilistic relational models. In *Introduction to Statistical Relational Learning* [19], chapter 5, pages 129–173.

[18] Lise Getoor, E. Segal, Benjamin Taskar, and Daphne Koller. Probabilistic models of text and link structure for hypertext classification. In *IJCAI01 Workshop on Text Learning: Beyond Supervision*, 2001.

[19] Lise Getoor and Ben Tasker. *Introduction to statistical relational learning*. MIT Press, 2007.

[20] Lise Getoor Getoor, Nir Friedman, and Benjamin Taskar. Learning probabilistic models of relational structure. pages 170–177. Morgan Kaufmann, 2001.

[21] Jing-Feng Guo, Jing Li, and Wei-Feng Bian. An efficient relational decision tree classification algorithm. *International Conference on Natural Computation*, 3:530–534, 2007.

[22] Tuyen N. Huynh and Raymond J. Mooney. Discriminative structure and parameter learning for markov logic networks. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *ICML*, pages 416–423. ACM, 2008.

[23] Implementation, Experiments, Anna Atramentov, Hector Leiva, and Vasant Honavar. A multi-relational decision tree learning algorithm -. In *Proceedings of the 13th International Conference on Inductive Logic Programming (ILP 2003)*, pages 38–56. Springer-Verlag, 2003.

[24] Neville Jennifer, Jensen David, Friedland Lisa, and Hay Michael. Learning relational probability trees. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 625–630, 2003.

[25] David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning (2002). In *In Proceedings of the 19th International Conference on Machine Learning*, 2002.

[26] Kristian Kersting and Luc de Raedt. Bayesian logic programming: Theory and tool. In *Introduction to Statistical Relational Learning* [19], chapter 10, pages 291–318.

[27] Hassan Khosravi, Oliver Schulte andTong Man, Xiaoyuan Xu, and Bahareh Bina. Structure learning for Markov logic networks with many descriptive attributes. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI).*, pages 487–493, 2010.

[28] Wim Van Laer and Luc de Raedt. How to upgrade propositional learners to first-order logic: A case study. In *Relational Data Mining*. Springer Verlag, 2001.

[29] Niels Landwehr, Kristian Kersting, and Luc De Raedt. nfoil: Integrating naïve bayes and foil. In Manuela M. Veloso and Subbarao Kambhampati, editors, *AAAI*, pages 795–800. AAAI Press / The MIT Press, 2005.

[30] Hongyan Liu, Xiaoxin Yin, and Jiawei Han. An efficient multi-relational naïve Bayesian classifier based on semantic relationship graph. In *MRDM '05: Proceedings of the 4th international workshop on Multi-relational mining*, pages 39–48, New York, NY, USA, 2005. ACM.

[31] Qing Lu and Lise Getoor. Link-based classification. In Tom Fawcett, Nina Mishra, Tom Fawcett, and Nina Mishra, editors, *ICML*, pages 496–503. AAAI Press, 2003.

[32] G. Manjunath, M.N. Murty, and D. Sitaram. A heterogeneous naive-bayesian classifier for relational databases. Technical report, HP Laboratories, 2009.

[33] Wolfgang May. Information extraction and integration with florid: The mondial case study. Technical report, Universitat Freiburg, Institut fur Informatik, 1999.

[34] Stephen Muggleton. Inductive logic programming. *New Gen. Comput.*, 8(4):295–318, 1991.

[35] Stephen Muggleton and John Firth. Relational rule induction with cprogol4.4: A tutorial introduction. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, chapter 7. Springer Verlag, 2001.

[36] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

[37] J. Neville, D. Jensen, B. Gallagher, and R. Fairgrieve. Simple estimators for relational bayesian classifiers. In *Proceedings of the 3rd IEEE international conference on data mining*, pages 609–612. Citeseer, 2003.

[38] Andrew Y. Ng and Michael L. Jordan. On discriminative versus generative classifiers: An empirical comparison of logistic regression and Naive Bayes. In *Proceedings of* NIPS*, Neural Information Processing Systems 14.*, 2002.

[39] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[40] David Poole. First-order probabilistic inference. In Georg Gottlob and Toby Walsh, editors, *IJCAI*, pages 985–991. Morgan Kaufmann, 2003.

[41] Hoifung Poon and Pedro Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*. AAAI Press, 2006.

[42] Alexandrin Popescul and Lyle Ungar. Feature generation and selection in multi-relational learning. In *An Introduction to Statistical Relational Learning* [19], chapter 8.

[43] Foster J. Provost and Pedro Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52(3):199–215, 2003.

[44] J. Ross Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[45] J. Ross Quinlan and R. Mike Cameron-Jones. Foil: A midterm report. In *ECML*, volume 667, pages 3–20. Springer, 1993.

[46] Rajat Raina, Yirong Shen, Andrew Y. Ng, and Andrew Mccallum. Classification with hybrid generative/discriminative models. In *In Advances in Neural Information Processing Systems 16*. MIT Press, 2003.

[47] Dymitr Ruta and Bogdan Gabrys. An overview of classifier fusion methods. *Computing and Information Systems*, pages 1–10, 2000.

[48] Rong She, Ke Wang, and Yabo Xu. Pushing feature selection ahead of join. In *Proceedings of the Fifth SIAM International Conference on Data Minig*, 2005.

[49] Kramer Stefan, Lavravc Nada, and Flach Peter. *Propositionalization approaches to relational data mining.* Springer-Verlag New York, Inc., New York, NY, USA, 2000.

[50] Benjamin Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In Adnan Darwiche and Nir Friedman, editors, *UAI*, pages 485–492. Morgan Kaufmann, 2002.

[51] J. D. Ullman. *Principles of database systems.* 2. Computer Science Press, 1982.

[52] Laer Wim Van, Raedt Luc De, and Dzeroski Saso. On multi-class problems and discretization in inductive logic programming. In *Proceedings of the 10th International Symposium on Foundations of Intelligent Systems*, pages 277–286, 1997.

[53] C. Vens, A. Van Assche, H. Blockeel, and S. Dzeroski. First order random forests with complex aggregates. In *Proceedings of the 14th International Conference on Inductive Logic Programming*, 2004.

[54] Xiaoxin Yin and Jiawei Han. Exploring the power of heuristics and links in multi-relational data mining. In *ISMIS'08: Proceedings of the 17th international conference on Foundations of intelligent systems*, pages 17–27, Berlin, Heidelberg, 2008. Springer-Verlag.

[55] Xiaoxin Yin, Jiawei Han, Jiong Yang, and Philip S. Yu. Crossmine: Efficient classification across multiple database relations. In *Constraint-Based Mining and Inductive Databases*, pages 172–195, 2004.