STRUCTURE LEARNING WITH JOIN BAYESIAN NETWORKS

by

Tong Man

A report submitted in partial fulfillment of the requirements for the SFU-ZU dual degree of Bachelor of Science in the School of Computing Science Simon Fraser University and the College of Computer Science and Technology Zhejiang University

© Tong Man 2010

SIMON FRASER UNIVERSITY AND ZHEJIANG UNIVERSITY

Spring 2010

All rights reserved. This work may not be reproduced in whole or in part, by photocopy or other means, without the permission of the author.
Name: Tong Man

Degree: Bachelor of Science

Title of Report: Structure Learning with Join Bayesian Networks

Examing Committee:

________________________________________
Dr. Ramesh Krishnamuti, Supervisor

________________________________________
Dr. Qianping Gu, SFU Examiner

________________________________________
Hassan Khosravi, Supervisor

Date Approved: 

ii
Abstract

The vast majority of work in Machine Learning has focused on propositional data, however, many real world datasets are relational and most real world applications are characterized by the presence of uncertainty and complex relational structure where the data distribution is neither identical nor independent. An emerging research area, Statistical Relational Learning (SRL), attempts to represent, model, and learn in the relational domain. Markov Logic Networks (MLNs) [Domingos, 2007] are a prominent statistical relational model that consist of weighted first order clauses. Many of the current state-of-the-art algorithms for learning MLNs have focused on relatively small datasets with few descriptive attributes, where predicates are mostly binary and the main task is usually prediction of links between entities. In this work we use Bayesian networks to deliver structure learning algorithms of MLNs with significantly less memory and time.

Among all the SRL approaches, MLNs deliver decent results with the use of Markov Logic Networks. The Join Bayesian Network (JBN), is an alternative approach that aims at delivering better learning results with significantly less memory and time. We measure the performance enhancement of JBN by comparing the running time and result with an open-source benchmark system for MLNs. Evaluations indicate that the new approach provides more accurate predictions two orders of magnitude faster than MLNs.
Acknowledgments

I would like to thank Hassan, Khosravi and Prof. Oliver Schulte for the work on Join Bayesian Networks, and Xiaoyuan Xu for doing the evaluation together.
Contents

Approval  ii
Abstract   iii
Acknowledgments  iv
Contents    v
List of Tables vii
List of Figures viii

1 Introduction  1

2 Statistical Relational Learning  3
  2.1 Background and Notation  3
  2.1.1 Learning from Data  3
  2.1.2 Enumeration, Independence and Conditional Independence  4
  2.1.3 Markov Logic Networks  7
  2.1.4 Bayesian Network  8
  2.2 Statistical Relational Learning  8

3 Join Bayesian Network  10
  3.1 Implementation of Alchemy Package  10
  3.2 Join Bayesian Network: Structure Learning  11
4 Evaluation

4.1 Implementation

4.1.1 Accuracy

4.1.2 Conditional Log Likelihood (CLL)

4.1.3 Area Under Curve (AUC)

4.2 Datasets

4.2.1 University

4.2.2 Movielens

4.2.3 Mutagenius

4.2.4 Subsampling

4.3 Results

5 Conclusion Future Work

5.1 Conclusion

5.2 Future Work: Parameter learning with JBN

Bibliography

Index
# List of Tables

2.1 Sample data of 8 people .................................................. 4  
2.2 Enumeration of Toothache, Catch, Cavity and Weather .............. 5  
2.3 The Weather Table ....................................................... 5  
2.4 The Catch, Toothache, and Cavity Table .............................. 6  
2.5 The Probability of Toothache given Cavity ............................ 6  
2.6 The Probability of Catch given Cavity ................................ 6  
2.7 The Probability of Cavity ................................................ 7  

4.1 The output of the inference of Alchemy ................................. 16  
4.2 The true values retrieved from the dataset ............................ 16  
4.3 Size of datasets in total number of table tuples and ground atoms. Each descriptive attribute is represented as a separate function, so the number of ground atoms is larger than that of tuples. ................................. 19  
4.4 Predictive performance represent the training error measured on the database by the JBN method and the two Alchemy structures. ..................... 20  
4.5 Performance measured on the database of the JBN method and the two Alchemy structures. ................................................. 20  
4.6 Predictive performance represent the training error measured on the database our JBN method and two Alchemy structures ............................ 22
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A figure showing a possible relationship learned from table 1.1</td>
<td>4</td>
</tr>
<tr>
<td>2.2</td>
<td>A figure showing a rule in Markov Logic Networks</td>
<td>7</td>
</tr>
<tr>
<td>3.1</td>
<td>The three steps of learning in Alchemy</td>
<td>11</td>
</tr>
<tr>
<td>3.2</td>
<td>The structure learning steps in Joint Bayesian Network</td>
<td>12</td>
</tr>
<tr>
<td>3.3</td>
<td>Pseudocode for Joint Bayesian Network structure learning</td>
<td>13</td>
</tr>
<tr>
<td>4.1</td>
<td>Steps of evaluation plan</td>
<td>15</td>
</tr>
<tr>
<td>4.2</td>
<td>A relational schema for a university domain. Key fields are underlined</td>
<td>17</td>
</tr>
<tr>
<td>4.3</td>
<td>Sample data from the University Dataset</td>
<td>18</td>
</tr>
<tr>
<td>4.4</td>
<td>A refined version of evaluation plan, variables are transformed to boolean</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>versions and pushed into alchemy, the output Result C is designed to maximize</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the ability of Alchemy structure learner</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Next step of evaluation once parameter learning of JBN is implemented, we</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>will directly compare the combined learning results of jbn with alchemy</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The vast majority of work in learning has focused on propositional data which consists of identically structured entities that are assumed to be independent. However, many real world datasets are relational. Relational data consists of different types of entities where each entity is characterized by a different set of attributes. Relational data are more complex and reflect reality more accurately where examples are given as multiple related tables. The structure of relational data provides an opportunity for objects to carry additional information via their links and permits the model to show correlations among objects and their relationships.

Statistical Relational Learning (SRL) is a new branch of machine learning that tries to model a joint distribution over relational data[2]. SRL is a combination of statistical learning, which addresses uncertainty in data, and relational learning, which deals with complex relational structures. A statistical relational model for a given database shows not only the correlations between attributes of each table, but also dependencies among attributes of different tables. Statistical relational models are usually represented with graphical models and differ in their methods of representation, learning, and inference.

One of the most prominent SRL models is the Markov Logic Network (MLN)[1], which uses both first-order logic and a Markov network model. MLNs have achieved impressive performance on a variety of SRL tasks. Because they are based on undirected graphical models, they avoid the difficulties with cycles that arise in directed SRL models [1, 7].

Join Bayesian Network (JBN)[3] uses a Bayesian Network (BN) to construct the structure of MLN, and is expected to take much less runtime and deliver more accurate results compared to current approaches. The main technical contribution of JBN is a new algorithm for
learning the structure of an MLN. The algorithm upgrades a single-table Bayesian Network learner by decomposing the learning problem for the entire database into learning problems for smaller tables. The basic idea is to apply the Bayesian Network learner repeatedly to tables and join tables from the database, and combine the resulting graphs into a single graphical model for the entire database. As the computational cost of the merge step is negligible, the run-time of this learn-and-join algorithm is essentially the cost of applying the Bayesian Network learner to each (join) table separately. Thus our algorithm leverages the scalability of single-table Bayesian Network learning to achieve scalability for MLN structure learning.

My contribution to this project are as follows:

- Implemented the parameter learning of JBN.
- Created a way to convert dataset into first-order logic for MLN and to constant version of MLN (together with Xiaoyuan Xu).
- Created a way to transform a learned structure of constant MLN back to normal MLN for evaluation.
- Created a series of scripts to run evaluation automatically.

In the next chapter, we will first provide the background and notation to cover the basic concepts in Machine Learning in general: enumeration, independence, conditional independence, Markov Logic Networks and Bayesian Networks. At the end of Chapter 2 we will talk about the differences between Statistical Relational Learning(SRL) and propositional learning. In Chapter 3 we will give a thorough description of structure learning in Join Bayesian Network to support theoretically the expected improvement in outcome. Chapter 4 will cover our evaluation in detail, a description of our dataset, the rationale, methodology, and implementation. Evaluation results are given and analyzed to demonstrate the enhancement of Join Bayesian Network. In Chapter 5, we conclude with discussion on future work of Join Bayesian Network.
Chapter 2

Statistical Relational Learning

2.1 Background and Notation

2.1.1 Learning from Data

We may use data from the real world to form a probabilistic graphical model, and then perform learning on the model to predict unknown events in the world. Learning from data involves the following steps:

1. Read the data, and do statistical analysis on the data, such as frequency counts.
2. Construct a probabilistic graphical model with the statistics gathered in Step 1
3. Answer queries based on the learned result, that is, inference

To begin with, we will follow a naive learning approach in order to understand how learning proceeds. Suppose we have three binary incidents in the world, Toothache, Catch, and Cavity, and we have a dataset of 8 people as shown in Table 2.1

A set of rules may be learned based on the statistics, provided “if a person has cavity, he will have toothache” has a probability of 100% can be learned since in rows 5 to 8 in Table 2.1, a True in Cavity always comes with a True in Toothache. Another rule may be “If a person does not have catch, there is a 50% probability that he will not have cavity” based on the statistical information gathered from rows 1 to 4. The rules are then portrayed visually by a graphical model, like Figure 2.1 (Note that different Graphical models may have different visualizations):
Once we have the probabilistic graphical model, it is possible to predict the probability of events based on some evidence. This procedure is called Inference. In our example, a patient experiencing toothache has a 100% probability of having cavity, based on our sample data.

### 2.1.2 Enumeration, Independence and Conditional Independence

To better understand the advantages of Bayesian network, we will discuss enumeration, independence and conditional independence first.

In an uncertain world, without any further information on incidents, we have to enumerate every situation in order to estimate the frequency counts. Thus, for N Boolean incidents, \(2^N\) probabilities are required in order to describe the world. For example, for 4 Boolean variables Toothache, Catch, Cavity, and Weather, a possible data table is given in Table 2.2.
To define independence between two incidents $A$ and $B$ in the world, we use the following formula ($P(A, B)$ means the probability of incident $A$ and $B$ happening at the same time):

$$P(A, B) = P(A)P(B)$$ (2.1)

If we can decompose this world into two partitions having $m$ and $p$ incidents respectively ($N = m + p$), we have $2^m + 2^p$ entries in total instead of $2^{m+p}$. In the above example, if we know that weather has nothing to do with Catch, Toothache, or Cavity, we can use Tables 2.3 and 2.4 to describe the world.

<table>
<thead>
<tr>
<th>Weather</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>0.9</td>
</tr>
<tr>
<td>F</td>
<td>0.1</td>
</tr>
</tbody>
</table>

Table 2.3: The Weather Table

Thus, for the 4 variables Weather, Catch, Toothache, and Cavity, instead of having 16 entries, we have $2 + 8 = 10$ entries taking independence into consideration. Now let us
CHAPTER 2. STATISTICAL RELATIONAL LEARNING

Consider the case of conditional independence. The product rule states that

\[ P(a, b) = P(a|b)P(b) = P(b|a)P(a) \quad (2.2) \]

In our example, we can write the full joint probability as follows, our data is represented in Tables 2.5, 2.6, and 2.7:

\[
P(\text{Toothache, Catch, Cavity}) = P(\text{Toothache} \mid \text{Catch, Cavity}) P(\text{Catch, Cavity})
= P(\text{Toothache} \mid \text{Catch, Cavity}) P(\text{Catch} \mid \text{Cavity}) P(\text{Cavity})
= P(\text{Toothache} \mid \text{Cavity}) P(\text{Catch} \mid \text{Cavity}) P(\text{Cavity})
\]

<table>
<thead>
<tr>
<th>Catch</th>
<th>Toothache</th>
<th>Cavity</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.4: The Catch, Toothache, and Cavity Table

This scheme uses only six entries, which indicates that by introducing conditional independence into our model, data can be represented with significantly less memory usage. Later in this report we will demonstrate how this technique can be used to enhance the performance of Markov Logic Networks.
2.1.3 Markov Logic Networks

Markov Logic Networks (MLNs) [1] are among the most well known methods proposed for statistical relational learning. Syntactically MLNs extend first-order logic and associate a weight with each formula. Semantically, they can represent a probability distribution over possible worlds using formulas and their corresponding weights. For example, a rule saying that Toothache, Catch, Cavity is closely related could be written as follows:

4.5 \( \text{Toothache(person1, true)} \land \text{Catch(person1, true)} \land \text{cavity(person1, true)} \) \hspace{1cm} (2.3)

The number 4.5 is the weight for the relationship, i.e. how strong the relationship is. The above rule can be represented as the graph in Figure 2.2.

![Figure 2.2: A figure showing a rule in Markov Logic Networks](image)

Thus, for each row in the joint probability table, a first-order logic sentence is required in Markov logic networks.
2.1.4 Bayesian Network

A Bayesian Network is a directed acyclic graph (DAG) \( G \), whose nodes comprise a set of random variables denoted by \( X \). When discussing a Bayesian Network, we refer interchangeably to its nodes or its variables. A Bayesian Network (BN) is a pair \( \langle G, \theta_G \rangle \) where \( \theta_G \) is a set of parameter values that specify the probability distribution of each node conditional on instantiations of its parents. These conditional probabilities are specified in a conditional probability table (CP-table) for variable \( X \). A Bayesian Network \( \langle G, \theta_G \rangle \) defines a joint probability distribution over \( V = \{ \text{node}_1, \ldots, \text{node}_n \} \). The joint probability of an assignment is obtained by multiplying the conditional probabilities of each value assignment to a node given its parent value assignments.

2.2 Statistical Relational Learning

The major difference between SRL and general machine learning discussed at the beginning of this chapter is the learning over relational data instead of propositional data. Most real world data is relational which provides more information about an object via its links. However, relational data has several major differences to propositional data, which makes it more challenging to learn models from. Several characteristics of relational data are the following:

1. The data representation for relational data and propositional data is different from propositional data. A relational database stores data in multiple tables that represent different types of entities and relationships between them. Propositional data is stored in a single table.

2. Propositional data consists of identically structured entities; however, relational data consists of different entities of different types which may be related to each other. For example, all tuples are of type student in propositional data and we assume all students are independent of each other. Relational data has tuples of different types (students and courses) and they may be dependent on each other.

3. In modeling propositional data, the number of different states is exponential in the number of attributes, e.g. with \( n \) binary attributes, the number of states is \( O(2^n) \). When modeling joint distribution of relational data, the number of different states is
exponential in the product of the attributes and objects, e.g. with \( m \) objects and \( n \) binary attributes in total, the number of states is \( O(2^{nm}) \).

4. The presence of **autocorrelation** is a feature of relational data which adds to the complexity of relational learning. A relationship between objects of the same entity is required to have autocorrelation. For example, \( \text{Friends}(s_1, s_2) = \text{true} \) is a relationship on entity \( \text{Student} \) where \( s_1 \) and \( s_2 \) are friends. There may be a pattern between the intelligence of friends, i.e., the attribute value \( \text{Intelligence}(s_1) \) and the attribute value \( \text{Intelligence}(s_2) \).

In our example, in order to do classification on a student, statistical relational learners may not only look at the courses that a student has taken, but also other students who have taken those courses.

5. One to many and many to many relationships exist only in relational data. For example, a student may be registered in a set of courses where the *ranking* of the student is related to the *difficulty* of a set of courses. Dealing with many to many relationships is a challenging problem for many SRL models.

Apart from the differences, relational data also has several unique features, compared with propositional data, which makes it more challenging to learn on. The most popular tasks introduced on relational data are the following.

- Linked based clustering groups together objects that have similar characteristics based on their own attributes and more importantly, the attributes of their links.

- Link prediction determines whether a relation exists between two objects from the attributes of the objects and their links.

We have have shown the challenges in SRL, and in the next chapter, we will demonstrate how Join Bayesian Network(JBN) tackles these challenges to learn relational data efficiently.
Chapter 3

Join Bayesian Network

3.1 Implementation of Alchemy Package

Alchemy is a software package developed by Prof Pedro Domingos and his team at the University of Washington[1]. It is developed for statistical relational learning, using a Markov logic network as the underlying implementation.

Alchemy performs SRL in 3 major steps:

1. **Structure learning**
   Structure learning is the first step; the general idea is to construct a Markov Logic Network to describe the relationship between entities in the world. The goal of structure learning is to find the correlation among nodes, represented by rules like as Equation 2.3 without the weight.

2. **Parameter learning**
   Given the learned graph from structure learning, the next step is to learn the weight of each arc in the graph. Parameter learning takes the structure and the original dataset as input, and outputs the weight of each relationship, i.e. arc in the graph.

3. **Inference**
   The final goal is to do inference. Inference is the process of drawing a conclusion by applying clues (of logic, statistics etc.) to observations or hypotheses. In SRL, inference takes the graph from parameter learning as input, as well as an evidence dataset, and outputs the probability of occurrence of events, by utilizing the information learned in the first two steps.
3.2 Join Bayesian Network: Structure Learning

Because a Markov Logic Network in SRL relies on first-order logic to represent the world with joint probability, its time and space complexities could potentially be improved using conditional probability as the underlying implementation. We develop a new algorithm to conduct the learning, called Join Bayesian Network. We replace the structure learning of MLNs with our new approach.

The basic idea of Join Bayesian Network is to do learning over each table, and then join the learned results. For a given dataset, we first run a Bayesian network learner on the entity tables, get some partial results, and then run the learner on the table joins. The second phase finds correlations among attributes of different tables, which were not captured in the first phase. The detailed phases are given below[3]:

1. **Analyze single tables**. Learn a Bayesian Network structure for the descriptive attributes of each entity table E of the database separately (with primary keys removed).

2. **Analyze single join tables**. Each relationship table is considered. The input table for the Bayesian Network learner is the join of the relational table with the entity tables linked by a foreign key constraint (with primary keys removed). Edges between attributes from the same entity table E are constrained to agree with the structure learned for E in phase (1)

3. **Analyze double join tables**. The input tables from the second phase are joined in pairs (if they share a common foreign key) to form the input tables for the Bayesian Network learner. Edges between variables considered in Phase (1) and (2) are constrained to agree with the structures previously learned.
4. **Satisfy slot chain constraints.** For each link $A \rightarrow B$ in the graph, where $A$ and $B$ are functors that correspond to attributes from different tables, arrows from Boolean relationship variables into $B$ are added if required to satisfy the following constraints: (1) $A$ and $B$ share a variable among their arguments, or (2) the parents of $B$ contain a chain of foreign key links connecting $A$ and $B$.

Figure 3.2 provides an outline of the phases.

![Figure 3.2: The structure learning steps in Joint Bayesian Network](image)

A pseudo code of the algorithm is given in Figure 3.3.

In this chapter, we have shown JBN algorithmically. In the next chapter, we will show the evaluation steps of JBN.
**Input:** Database D with E1; ::Ee entity tables, R1; :::Rr Relationship tables,  
**Output:** MLN for D  
**Calls:** PBN: Any propositional Bayesian net learner that accepts edge constraints and a single table of cases as input. WL: Weight learner in MLN  
**Notation:** PBN(T; Econstraints) denotes the output DAG of PBN. Get-Constraints(G) specifies a new set of edge constraints, namely that all edges in G are required, and edges missing between variables in G are forbidden.  
1: Add descriptive attributes of all entity and relationship tables as variables to G. Add a boolean indicator for each relationship table to G.  
2: Econstraints = ; [Required and Forbidden edges]  
3: for m=1 to e do  
4:    Econstraints += Get-Constraints(PBN(Em , ;))  
5: end for  
6: for m=1 to r do  
7:    Nm := join of Rm and entity tables linked to Rm  
8:    Econstraints += Get-Constraints(PBN(Nm, Econstraints))  
9: end for  
10: for all Ni and Nj with a foreign key in common do  
11:    Kij := join of Ni and Nj  
12:    Econstraints += Get-Constraints(PBN(Kij, Econstraints))  
13: end for

Figure 3.3: Pseudocode for Joint Bayesian Network structure learning
Chapter 4

Evaluation

In this chapter, we will report a thorough evaluation by comparing our approach with the structure learner from the Alchemy package[5] using the following steps:

1. Run the structure learner of Join Bayesian Network on the dataset. Denote the learned structure $S_1$.
2. Use $S_1$ as input to run the weight learner of Alchemy. Denote the output $W_1$.
3. Use $W_1$ to run inference. Denote the result $O_1$.
4. Run the structure learner of Alchemy. Denote the learned structure $S_2$.
5. Use $S_2$ as input to run the weight learner of Alchemy. Denote the output $W_2$.
6. Use $W_2$ to run inference. Denote the result $O_2$.
7. Generate evaluation metrics of $O_1$ and $O_2$.

The steps are shown in Figure 4.1

4.1 Implementation

The steps given above clearly show that any difference between result $O_1$ and result $O_2$ should be due to the two different structure learning methods used to construct the results. In order to compare results of the two different methods, we use three evaluation metrics: Accuracy, Conditional log likelihood (CLL), and Area Under Curve (AUC). We use a running example in Table 4.1 and Table 4.2 in order to clarify our evaluation metrics.
4.1.1 Accuracy

Accuracy of a measurement is an indication of how close a measured quantity is to the actual value. For accuracy, we use maximum likelihood\cite{8} to do prediction. In the given example in Tables 4.1 and 4.2, *Intelligence(Jack, 1)*, *Intelligence(Jane, 1)*, and *Intelligence(Jim, 1)* are the more likely results. Comparing this with the true values, we use the following equation in order to calculate accuracy. The accuracy of the given example is $0.67\left(\frac{2}{3}\right)$.

\[
Accuracy = \frac{\#\text{correctly predicted incidents}}{\#\text{total incidents}}
\]
### Chapter 4. Evaluation

<table>
<thead>
<tr>
<th>Student</th>
<th>Intelligence</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack 1</td>
<td>1</td>
<td>0.70</td>
</tr>
<tr>
<td>Jack 2</td>
<td>2</td>
<td>0.30</td>
</tr>
<tr>
<td>Jane 1</td>
<td>1</td>
<td>0.99</td>
</tr>
<tr>
<td>Jane 2</td>
<td>2</td>
<td>0.01</td>
</tr>
<tr>
<td>Jin 1</td>
<td>1</td>
<td>0.51</td>
</tr>
<tr>
<td>Jim 2</td>
<td>2</td>
<td>0.49</td>
</tr>
</tbody>
</table>

Table 4.1: The output of the inference of Alchemy

<table>
<thead>
<tr>
<th>Student</th>
<th>Intelligence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jack</td>
<td>1</td>
</tr>
<tr>
<td>Jane</td>
<td>1</td>
</tr>
<tr>
<td>Jim</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 4.2: The true values retrieved from the dataset

#### 4.1.2 Conditional Log Likelihood (CLL)

The conditional log likelihood (CLL) is a measurement on how trustworthy the results are. It is directly related to the probability given in the inference table for the true value. To calculate conditional log likelihood, we check the true value of an incident, and then check the probability given in the inference table. A summation over the natural log of such probabilities is calculated as the conditional log likelihood. For our given example, the conditional log likelihood is calculated as

$$CLL = \log(0.7) + \log(0.99) + \log(0.49)$$

The closer our result is to 0, the more confident we are with our prediction.

#### 4.1.3 Area Under Curve (AUC)

The AUC curves were computed by changing the conditional log likelihood threshold above which a ground atom is predicted true [3]. A curve is drawn out of the calculated points. The area of the curve is defined as the area under curve. A useful example of AUC is the prediction of cancer. Given the probability that a person has cancer is 1%, we could have the following situations:

1. A person does not have cancer and the learner predicts no cancer (true negative)
2. A person does not have cancer but the learner predicts cancer (false positive)

3. A person does have cancer and the learner predicts cancer (true positive)

4. A person does have cancer but the learner predicts no cancer (false negative)

Regardless of the person, a learner can always achieve an accuracy of 99% and a high CLL score by always reporting negative on queries of cancer. However, in the case of false negative, the learner will get the wrong result on this most important case (We usually want the learner to report cancer, not healthy). The more frequently a false negative occurs, the lower the AUC.

4.2 Datasets

In this report, we use two well-known real-world datasets and one synthetic dataset. We will discuss the structure and data of the university dataset in detail. The other two datasets will be briefly introduced to support the credibility of the sample datasets.

4.2.1 University

The university dataset has the schema in Figure 4.2

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>(student id, intelligence, ranking)</td>
</tr>
<tr>
<td>Course</td>
<td>(course id, difficulty, rating)</td>
</tr>
<tr>
<td>Professor</td>
<td>(professor id, teaching ability, popularity)</td>
</tr>
<tr>
<td>Registered</td>
<td>(student id, course id, grade, satisfaction)</td>
</tr>
<tr>
<td>RA</td>
<td>(student id, prof id, salary, capability)</td>
</tr>
</tbody>
</table>

Figure 4.2: A relational schema for a university domain. Key fields are underlined

The university dataset contains some information of a typical university, where each student has an id as primary key, intelligence and ranking as attributes. A student can take courses, which is represented by the Registration table, and each student has a grade and a satisfaction value for each course taken. A student can also be an RA for a professor, which is represented by the RA table, and each RA has a salary and capability. A course has an
CHAPTER 4. EVALUATION

18

(a) Student

(b) Course

(c) Registration

(d) RA

(e) Professor

Figure 4.3: Sample data from the University Dataset

id as primary key, a difficulty level and ranking as attributes. Professor is another table in this dataset, with id as primary key, and teaching ability and popularity as attributes. Values of attributes are carefully designed to be highly relevant to each other. With only 171 tuples and 513 Ground factors, it is a very small dataset aimed at quickly testing the quality of learning. Figure 4.3 is a small example of the university dataset.

4.2.2 Movielens

The second dataset is the MovieLens dataset from the UC Irvine machine learning repository. It contains two entity tables: User with 941 tuples and Item with 1,682 tuples, and one relationship table Rated with 80,000 ratings. The User table has three descriptive attributes age, gender, occupation. The table Item represents information about the movies. It has 17 Boolean attributes that indicate the genres of a given movie. We performed a preliminary data analysis and omitted genres that have only weak correlations with the rating or user attributes, leaving a total of three genres.

4.2.3 Mutagenius

This dataset is widely used in ILP research [6]. Mutagenesis has two entity tables, Atom with three descriptive attributes, and Mole, with 188 entries and 5 descriptive attributes,
including two attributes that have ten possible values each (logp and lumo). It features two relationships *MoleAtom* indicating which atoms are parts of which molecules, and *Bond* which relates two atoms and has one descriptive attribute.

### 4.2.4 Subsampling

Our experiments show that Alchemy cannot handle all the datasets due to limitation of memory. Therefore, we subsampled the movielens and mutagenesis dataset, as described below:

- We took the first 100 items from each table in the movielens dataset to produce movielens subsample1.
- We took 300 items from each table in movielens, to produce movielens subsample2.
- We took 300 items from each table in mutagenesis, to produce mutagenesis subsample1.
- We took 500 from each table in mutagenesis, to produce mutagenesis subsample2.

The subsamples keep the general complexity of the dataset, and as the number of samples increase, the learner will be able to learn the same structure as if it is learning from the whole dataset.

Table 4.3 is a summary of the datasets we are using

<table>
<thead>
<tr>
<th>Dataset</th>
<th># tuples</th>
<th>#Ground atoms</th>
</tr>
</thead>
<tbody>
<tr>
<td>University</td>
<td>171</td>
<td>513</td>
</tr>
<tr>
<td>Movielens</td>
<td>82623</td>
<td>170143</td>
</tr>
<tr>
<td>MovieLens subsample1</td>
<td>1468</td>
<td>3485</td>
</tr>
<tr>
<td>MovieLens subsample2</td>
<td>12714</td>
<td>27134</td>
</tr>
<tr>
<td>Mutagenesis</td>
<td>15218</td>
<td>35973</td>
</tr>
<tr>
<td>Mutagenesis subsample1</td>
<td>3375</td>
<td>5868</td>
</tr>
<tr>
<td>Mutagenesis subsample2</td>
<td>5675</td>
<td>9058</td>
</tr>
</tbody>
</table>

Table 4.3: Size of datasets in total number of table tuples and ground atoms. Each descriptive attribute is represented as a separate function, so the number of ground atoms is larger than that of tuples.
4.3 Results

Data reformatting was used by Kok and Domingos [4]. To illustrate, the predicate \textit{Salary} with values (high, medium, low) is represented with a single binary predicate Salary(Student, \textit{Salary\_type}) in the standard Alchemy input format. The reformatted file contains instead three unary predicates \textit{Salary\_high}(Student), \textit{Salary\_med}(Student), \textit{Salary\_low}(Student). The effect is that the arguments to all predicates are primary keys, which tends to yield better results with Alchemy. To evaluate predictions, the MLNs learned with the reformatted data were converted to the format of the original database. We denote the new way to interpret data as constant MLN (CMLN). The flow chart for constant MLN is given in Figure 4.4.

We have three learning methods to evaluate. All experiments were done on a machine with a QUAD CPU Q6700 of 2.66GHz and 8GB of RAM. The results are shown in Table 4.4. The runtime of our experiments are shown in Table 4.5.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>JBN</th>
<th>MLN</th>
<th>CMLN</th>
<th>JBN</th>
<th>MLN</th>
<th>CMLN</th>
<th>JBN</th>
<th>MLN</th>
<th>CMLN</th>
</tr>
</thead>
<tbody>
<tr>
<td>University</td>
<td>0.85</td>
<td>0.37</td>
<td>0.51</td>
<td>-0.4</td>
<td>-5.79</td>
<td>-3.24</td>
<td>0.88</td>
<td>0.45</td>
<td>0.68</td>
</tr>
<tr>
<td>Movielens sample1</td>
<td>0.67</td>
<td>0.43</td>
<td>0.43</td>
<td>-0.75</td>
<td>-4.09</td>
<td>-2.83</td>
<td>0.7</td>
<td>0.46</td>
<td>0.53</td>
</tr>
<tr>
<td>Movielens sample2</td>
<td>0.65</td>
<td>0.42</td>
<td>0.42</td>
<td>-1</td>
<td>-3.55</td>
<td>-3.94</td>
<td>0.69</td>
<td>0.49</td>
<td>0.51</td>
</tr>
<tr>
<td>Movielens</td>
<td>0.69</td>
<td>NT</td>
<td>NT</td>
<td>-0.7</td>
<td>NT</td>
<td>NT</td>
<td>0.73</td>
<td>NT</td>
<td>NT</td>
</tr>
<tr>
<td>Mutagenesis sample1</td>
<td>0.81</td>
<td>0.36</td>
<td>0.55</td>
<td>-0.6</td>
<td>-4.7</td>
<td>-3.38</td>
<td>0.9</td>
<td>0.56</td>
<td>0.6</td>
</tr>
<tr>
<td>Mutagenesis sample2</td>
<td>0.81</td>
<td>NA</td>
<td>0.35</td>
<td>-0.6</td>
<td>NA</td>
<td>-4.65</td>
<td>0.9</td>
<td>NA</td>
<td>0.56</td>
</tr>
</tbody>
</table>

Table 4.4: Predictive performance represent the training error measured on the database by the JBN method and the two Alchemy structures.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>JBN</th>
<th>MLN</th>
<th>CMLN</th>
</tr>
</thead>
<tbody>
<tr>
<td>University</td>
<td>0.03+0.032</td>
<td>5.02</td>
<td>11.44</td>
</tr>
<tr>
<td>MovieLens</td>
<td>1.2+120</td>
<td>NT</td>
<td>NT</td>
</tr>
<tr>
<td>MovieLens Subsample 1</td>
<td>0.05+0.33</td>
<td>44</td>
<td>121.5</td>
</tr>
<tr>
<td>MovieLens Subsample 2</td>
<td>0.12+5.10</td>
<td>2760</td>
<td>1286</td>
</tr>
<tr>
<td>Mutagenesis</td>
<td>0.5 +NT</td>
<td>NT</td>
<td>NT</td>
</tr>
<tr>
<td>Mutagenesis subsample 1</td>
<td>0.1+5</td>
<td>3360</td>
<td>900</td>
</tr>
<tr>
<td>Mutagenesis subsample 2</td>
<td>0.2+12</td>
<td>NT</td>
<td>3120</td>
</tr>
</tbody>
</table>

Table 4.5: Performance measured on the database of the JBN method and the two Alchemy structures.
The runtime table provides proof that the Join Bayesian method is orders of magnitude faster than the state-of-the-art methods for MLNs, while also being able to improve on accuracy, conditional log likelihood and AUC significantly. Table 4.4 shows results on experiments where the same data is used for both learning and prediction. This experiment shows that the correct frequency counts are learned during the structure learning phase. When the same data is used for learning and inferencing, an effect called over fitting may occur. In order to examine a generalization of the learning, we run another test with 2/3 trained data and 1/3 test data, the results of which are given in Table 4.6.
### Table 4.6: Predictive performance represent the training error measured on the database our JBN method and two Alchemy structures

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Accuracy</th>
<th>Log likelihood</th>
<th>Area under Curve</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>JBN</td>
<td>MLN</td>
<td>CMLN</td>
</tr>
<tr>
<td>Movielens sample1</td>
<td>0.63</td>
<td>0.39</td>
<td>0.45</td>
</tr>
<tr>
<td>Movielens sample2</td>
<td>0.59</td>
<td>0.42</td>
<td>0.46</td>
</tr>
<tr>
<td>Mutagenesis sample1</td>
<td>0.60</td>
<td>0.34</td>
<td>0.47</td>
</tr>
<tr>
<td>Mutagenesis sample2</td>
<td>0.68</td>
<td>NT</td>
<td>0.53</td>
</tr>
</tbody>
</table>

Table 4.6 provides evidence that the new Join Bayesian Network is superior to state-of-the-art methods of learning MLNs. It is clear that for the same dataset, JBN improves the accuracy of MLN by at least 20%, at greater confidence (indicated by lower values of log likelihood), and less false negatives (indicated by high values of AUC).
Chapter 5

Conclusion Future Work

5.1 Conclusion

Using the join-and-learn approach, we were able to obtain at least 20% more accurate results, at two orders of magnitude faster in runtime over three datasets. The main contribution of the project is to provide a brand-new approach to SRL. The idea of divide and conquer will not only decrease running time, but also make it possible to run the learning concurrently over multiple CPUs, which enables the implementation of the algorithm in a distributed system in the future. Our approach of transforming a Bayesian network to a Markov Logic Network at any stage of the learning will also help researchers compare the two approaches.

5.2 Future Work: Parameter learning with JBN

An extension of the current research is to also learn the parameters without using MLNs. An outline of how this could be done is given in the following steps.

1. Scan the entity tables, looking for nodes whose direct parents are in the same table as the node itself, or nodes who do not have parents. Learn parameters over that table to fill in the conditional probability table on that node.

2. Scan the two table joins, looking for nodes whose direct parents are in the same joint view as the node itself. Learn parameters over the joint view to fill in the conditional probability table on that node.
3. Repeat Step 2 for multiple table joins until all the conditional probability tables on all nodes in the Bayesian Network have been filled in.

Due to the introduction of Relationship entities, such as $B_{RA}(prof\_id,stu\_id)$, some conditional probabilities of a node cannot be calculated from any of the tables or joint views. For example, the probability of $(GPA = 4|B_{RA} = true)$ can be learned by joining the student and RA table, but the probability of $(GPA = 4|B_{RA} = *)$ should be learned by just looking at student table. Moreover, the probability of $(GPA = 4|B_{RA} = false)$ can only be calculated by using $Probability(GPA = 4|B_{RA} = *) - Probability(GPA = 4|B_{RA} = true)$. So whenever we are faced with a relational entity value such as $B_{RA}$, we need to look at its actual value in order to decide how to learn it.

Once the parameter learning is implemented, we are able to redo the evaluation with the pattern in Figure 5.1. The new evaluation plan will take in the same dataset, then run the structure learning and parameter learning with JBN and Alchemy respectively.
Figure 5.1: Next step of evaluation once parameter learning of JBN is implemented, we will directly compare the combined learning results of JBN with Alchemy.
Bibliography


Index

Accuracy, 15
Alchemy, 10
Area Under Curve, 16
autocorrelation, 9
Bayesian Network, 8
Conclusion, 23
Conditional Independence, 5
Conditional Log Likelihood, 16
CP-table, 8
Enumeration, 4
Evaluation, 14

Future Work: Parameter learning with JBN,
23

Independence, 5
Inference, 3

Join Bayesian Network: Structure Learning,
11

Markov Logic Networks, 7
over fitting, 21

Realational Data, 1
Results, 20
Runtime, 20

Statistical Relational Learning, 1, 8
Subsampling, 19