Learning classifiers from linked data

by

Harris Lin

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:

Vasant Honavar, Major Professor

Samik Basu

Oliver Eulenstein

Giora Slutzki

Jin Tian

Iowa State University

Ames, Iowa

2013

DEDICATION

To my family: Yvonne, Tawen, and Catherine

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my advisor Dr. Vasant Honavar, for his continued and insightful mentoring, and his perseverant support to all stages of my graduate studies. I am also thankful for his boundless tolerance, that allows me to make mistakes and grow up from them.

I would also like to thank my program of study committee members, for supervising my graduate program and providing valuable feedbacks. My special thanks go to Dr. Oliver Eulenstein who provided a research assistantship that supported part of my graduate studies while I was still in the Masters program.

Many thanks also go to current and some of past members of the Artificial Intelligence Research Lab: Jie Bao for introducing me to the world of Semantic Web, Neeraj Koul for his constant inspiration and support, Yasser EL-Manzalawy for his kind leadership in the lab, Ngot Bui for helping with dataset preparation, Sanghack Lee for assisting with implementation on distributional data. I also appreciate many entertaining lunch discussions with all members, including Rafael Jordan, Jia Tao, Fadi Towfic, Rasna Walia, and Li Xue.

Furthermore I would like to extend my acknowledgement to Grandmaster Farrah Chen and Grandmaster Yong Chin Pak, who provided invaluable spiritual support during some of the difficult times throughout my graduate studies.

This research was supported in part by a grant (IIS 0711356) from the National Science Foundation (NSF), in part by the Iowa State University Center for Computational Intelligence, Learning, and Discovery, and in part by teaching assistantships in Computer Science Department.

ABSTRACT

The emergence of many interlinked, physically distributed, and autonomously maintained linked data sources amounts to the rapid growth of Linked Open Data (LOD) cloud, which offers unprecedented opportunities for predictive modeling and knowledge discovery from such data. However existing machine learning approaches are limited in their applicability because it is neither desirable nor feasible to gather all of the data in a centralized location for analysis due to access, memory, bandwidth, or computational restrictions. In some applications additional schema such as subclass hierarchies may be available and exploited by the learner. Furthermore, in other applications, the attributes that are relevant for specific prediction tasks are not known a priori and hence need to be discovered by the algorithm. Against this background, we present a series of approaches that attempt to address such scenarios. First, we show how to learn Relational Bayesian Classifiers (RBCs) from a single but remote data store using statistical queries, and we extend to the setting where the attributes that are relevant for prediction are not known a priori, by selectively crawling the data store for attributes of interest. Next, we introduce an algorithm for learning classifiers from a remote data store enriched with subclass hierarchies. Our algorithm encodes the constraints specified in a subclass hierarchy using latent variables in a directed graphical model, and adopts the Variational Bayesian EM approach to efficiently learn parameters. In retrospect, we observe that in learning from linked data it is often useful to represent an instance as tuples of bags of attribute values. With this inspiration, we introduce, formulate, and present solutions for a novel type of learning problem which we call distributional instance classification. Finally, building up from the foundations, we consider the problem of learning predictive models from multiple interlinked data stores. We introduce a distributed learning framework, and identify three special cases of linked data fragmentation then describe effective strategies for learning predictive models in each case. Further, we consider a novel application of a matrix reconstruction technique from the field

of Computerized Tomography to approximate the statistics needed by the learning algorithm from projections using count queries, thus dramatically reducing the amount of information transmitted from the remote data sources to the learner.

CHAPTER 1. LEARNING CLASSIFIERS FROM LINKED DATA: AN OVERVIEW

1.1 Introduction

The growing adoption of a set of best practices collectively referred to as Linked Data, for publishing structured data on the Web [Heath and Bizer (2011)] has made it possible to link and share many disparate, previously isolated, distributed, autonomously generated and managed data across virtually every domain of human endeavor. The community-driven Linked Open Data (LOD) effort allows structured data to be represented using Resource Description Framework (RDF) [Manola and Miller (2004)] in the form of subject-predicate-object triples (also called RDF triples), which describe a directed graph where the directed labeled edges encode binary relations between labeled nodes. RDF stores and associated query languages such as SPARQL [Prud'ommeaux and Seaborne (2008)] offer the means to store and query large amounts of RDF data. LOD also enables integration of previously isolated distributed data such as data stored in multi-relational databases [Hert et al. (2011)]. At present, LOD includes more than a few hundred linked data sets that together contain more than a few trillion RDF triples [Cyganiak and Jentzsch (2011)] covering domains including government, life sciences, geography, social media, and commerce. The emergence of LOD offers unprecedented opportunities for using disparate data sources in predictive modeling and decision making in a broad range of applications ranging from scientific discovery to public policy [Hert et al. (2011); Sahoo et al. (2009)]. However, there has been very limited work on effective approaches to learning predictive models from LOD. Existing machine learning approaches are limited in their applicability in this setting:

- Many data sets that are part of LOD are so large that it is not feasible to retrieve the

entire data set from an RDF store for local analysis due to main memory, bandwidth, or computational constraints. In other settings, access to data may be limited due to the privacy or confidentiality considerations [Aggarwal and Yu (2008); Wu et al. (2010)]. This calls for techniques for learning predictive models from an RDF store which supports only indirect access to data through a query interface (SPARQL access point) that supports a limited class of queries (e.g., counts of RDF triples that match some specified criteria) [Prud'ommeaux and Seaborne (2008)].

- RDF triples in an RDF store have often associated with them, RDF Schema (RDFS, Brickley and Guha (2004)) that specifies set of classes that organize RDF objects (subjects and objects of predicates) and predicates into type hierarchies as well as domain and range restrictions on RDF predicates (i.e., the type of RDF objects that can appear as subjects or objects of a predicate respectively). Hence, there is a need for algorithms that can effectively exploit RDFS to construct compact and accurate predictive models from RDF data.

- RDF stores are sometimes subject to frequent updates. In such settings, it is necessary to update the predictive model to reflect the changes in the data source used to build the model. While in principle, the learning algorithm can be re-executed each time there is an update to the data source, it is of interest to explore more efficient solutions for incrementally updating the predictive model without the need to process the entire data set each time there is an update.

- LOD includes many loosely linked, physically distributed, autonomously maintained RDF stores where it is neither desirable nor feasible to gather all of the data in a centralized location for analysis, because of access, memory, bandwidth, and computational restrictions. Hence, there is a need for effective approaches to learning predictive models from distributed, loosely linked RDF stores.

Against this background, this dissertation aims to develop and evaluate novel statistical machine learning algorithms for learning predictive models from LOD.

1.2 Dissertation Overview

Chapter 1: Learning Classifiers from Linked Data: an Overview

We present a general introduction and motivation for the problem of learning from Linked Data, as well as an overview and a summary of contributions of this dissertation.

Chapter 2: Learning Relational Bayesian Classifiers from RDF Data

This chapter presents our first set of results, showing an approach to learning Relational Bayesian Classifiers (RBCs, [Neville et al. (2003b)]) from a single but remote RDF data store. Specifically, we show how to build RBCs from RDF data using statistical queries through the SPARQL endpoint of the RDF store. We compare the communication complexity of our algorithm with one that requires direct centralized access to the data and hence has to retrieve the entire RDF dataset from the remote location for processing. We establish the conditions under which the RBC models can be incrementally updated in response to addition or deletion of RDF data. We show how our approach can be extended to the setting where the attributes that are relevant for prediction are not known a priori, by selectively crawling the RDF data for attributes of interest.

This chapter has been published in [Lin et al. (2011)].

Chapter 3: Learning Classifiers From RDF Data with Subclass Hierarchies

We consider a more general problem than what is presented in Chapter 2, where we introduce an algorithm for learning classifier from a remote RDF data store enriched with subclass hierarchies. Our algorithm encodes the constraints specified in a subclass hierarchy using latent variables in a directed graphical model, and adopts the Variational Bayesian EM approach to efficiently learn parameters. We conduct experiments with several real world datasets and show that our solution achieves equal or better performance compared to other state-of-art models that incorporate subclass hierarchies, and is able to scale up to large hierarchies over few thousand nodes.

Chapter 4: Learning Classifiers from Distributional Data

We observe in the previous two chapters that in learning from linked data (or relational data in general) it is often useful to represent an instance as K-tuples of *bags of feature values*, however we believe that a precise probabilistic formulation of this representation is not sufficiently covered in the literature. Inspired by this representation, in this chapter we set aside our motivation of learning from linked data and investigate a novel type of learning problem which we call *distributional instance classification*. We motivate, precisely formulate, and present solutions for this problem. We will resume our setting of linked data in Chapter 5 where we will solve a more challenging problem that utilizes the results presented in this chapter.

This chapter has been published in [Lin et al. (2013)].

Chapter 5: Learning Classifiers from Chains of Multiple Interlinked RDF Data Stores

In this chapter we consider the problem of learning predictive models from *multiple interlinked* RDF stores. We extend Chapter 2 by incorporating the set of classifiers introduced in Chapter 4, and further extend the problem by considering a chain of multiple interlinked RDF stores. Specifically we: (i) introduce statistical query based formulations of several representative algorithms for learning classifiers from RDF data; (ii) introduce a distributed learning framework to learn classifiers from multiple interlinked RDF stores that form a chain; (iii) identify three special cases of RDF data fragmentation and describe effective strategies for learning predictive models in each case; (iv) consider a novel application of a matrix reconstruction technique from the field of Computerized Tomography [Herman (2009)] to approximate the statistics needed by the learning algorithm from projections using count queries, thus dramatically reducing the amount of information transmitted from the remote data sources to the learner; and (v) report results of experiments with a real-world social network data set (Last.fm), which demonstrate the feasibility of the proposed approach.

This chapter has been published in [Lin and Honavar (2013)].

Chapter 6: General Conclusions

We conclude with a summary of the dissertation, the contributions, and broader directions for future work.

CHAPTER 2. LEARNING RELATIONAL BAYESIAN CLASSIFIERS FROM RDF DATA

The massive size and distributed nature of RDF data call for approaches to learning from RDF data in a setting where the data can be accessed only through a query interface, e.g., the SPARQL endpoint of the RDF store. In applications where the data are subject to frequent updates, there is a need for algorithms that allow the predictive model to be incrementally updated in response to changes in the data. Furthermore, in some applications, the attributes that are relevant for specific prediction tasks are not known a priori and hence need to be discovered by the algorithm. In this chapter we present an approach to learning Relational Bayesian Classifiers (RBCs) from RDF data that addresses such scenarios. Specifically, we show how to build RBCs from RDF data using statistical queries through the SPARQL endpoint of the RDF store. We compare the communication complexity of our algorithm with one that requires direct centralized access to the data and hence has to retrieve the entire RDF dataset from the remote location for processing. We establish the conditions under which the RBC models can be incrementally updated in response to addition or deletion of RDF data. We show how our approach can be extended to the setting where the attributes that are relevant for prediction are not known a priori, by selectively crawling the RDF data for attributes of interest. We provide open source implementation and evaluate the proposed approach on several large RDF datasets.

2.1 Introduction

The increasing availability of large RDF datasets on the web offers unprecedented opportunities for extracting useful knowledge or predictive models from RDF data, and using the

resulting models to guide decisions in a broad range of application domains. Hence, it is natural to consider the use of machine learning approaches, and in particular, statistical relational learning algorithms [Getoor and Taskar (2007)], to extract knowledge from RDF data [Kiefer et al. (2008); Bicer et al. (2011); Tresp et al. (2009)]. However, existing approaches to learning predictive models from RDF data have significant shortcomings that limit their applicability in practice. Specifically, existing approaches rely on the learning algorithm having direct access to RDF data. However, in many settings, it may not be feasible to transfer data a massive RDF dataset from a remote location for local processing by the learning algorithm. Even in settings where it is feasible to provide the learning algorithm direct access to a local copy of an RDF dataset, algorithms that assume in-memory access to data cannot cope with RDF datasets that are too large to fit in memory. Hence, there is an urgent need for approaches to learning from RDF data in a setting where the data can be accessed only through a query interface, e.g., the SPARQL endpoint for the RDF store. In applications where the data are subject to frequent updates, there is a need for algorithms that allow the predictive model to be incrementally updated in response to changes in the data. Furthermore, in some applications, the attributes that are relevant for specific prediction tasks are not known a priori and hence need to be discovered by the algorithm. We present an approach to learning Relational Bayesian Classifiers from RDF data that addresses such scenarios.

Our approach to learning Relational Bayesian Classifiers (RBCs) from RDF data adopts the general framework introduced by Caragea et al. (2005) for transforming a broad class of standard learning algorithms that assume in memory access to a dataset into algorithms that interact with the data source(s) only through statistical queries or procedures that can be executed on the remote data sources. This involves decomposing the learning algorithm into two parts: (i) a component that poses the relevant statistical queries to a data source to acquire the information needed by the learner; and (ii) a component that uses the resulting statistics to update or refine a partial model (and if necessary, further invoke the statistical query component). This approach has been previously used to learn a variety of classifiers from relational databases [Koul et al. (2008)] using SQL queries and from biomolecular sequence data [Koul et al. (2010)]. It has recently become feasible to use a similar approach to learning RBCs from RDF data due to the

incorporation of support for aggregate queries in SPARQL. (SPARQL 1.1 supports aggregate queries whereas SPARQL 1.0 does not).

We show how to learn RBCs from RDF data using only aggregate queries through the SPARQL endpoint of the RDF store. This approach does not require in-memory access to RDF data to be processed by the learning algorithm, and hence can scale up to very large data sets. Because the predictive model is built using aggregate queries against a SPARQL endpoint, it can be used to learn RBCs from large remote RDF stores without having to transfer the data to a local RDF store for processing (in general, the cost of retrieving the statistics needed for learning is much lower than the cost of retrieving the entire dataset). Under certain conditions which we identify in the paper, we show how the RBC models can be incrementally updated in response to changes (addition or deletion of triples) from the RDF store. We further show how our approach can be extended to the setting where the attributes that are relevant for prediction are not known a priori, by selectively crawling the RDF data for attributes of interest. We have implemented the proposed approach into INDUS [Koul and Lin (2013)], an open source suite of learning algorithms, that learn from massive data sets only using statistical queries. We describe results of experiments on several large RDF datasets that demonstrate the feasibility of the proposed approach to learning RBCs from RDF stores.

The rest of the chapter is organized as follows: Section 2.2 introduces a precise formulation of the problem of learning RBCs from RDF data. Section 2.3 describes how to build RBCs from RDF data using only aggregate queries. Section 2.4 identifies the conditions under which it is possible to incrementally update an RBC learned model from an RDF store in response to updates to the underlying RDF store. Section 2.5 presents an analysis of the communication complexity of learning RBCs from RDF stores. Section 2.6 describes how to extend to the setting where the attributes that are relevant for prediction are not known a priori, by selectively crawling the RDF data for attributes of interest. Section 2.7 describes results of experiments with several RDF datasets that demonstrate the feasibility proposed approach. Finally Section 2.8 concludes with a summary and a brief discussion of related work.

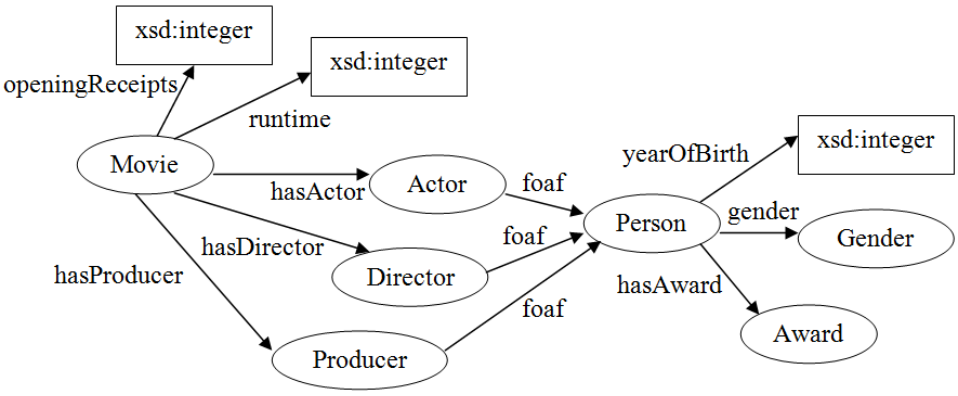

Figure 2.1 RDF schema for the movie domain

2.2 Problem Formulation

In this section we formulate the problem of learning predictive models from RDF data. Assume there are pairwise disjoint infinite sets I, B, L and V (IRIs, Blank nodes, Literals and Variables respectively). A triple $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ is called an RDF triple. In this triple, s is the subject, p the predicate, and o the object. An RDF graph is a set of RDF triples.

As a running example for the following definitions, we consider the RDF schema for the movie domain as shown in Figure 2.1. We wish to predict whether a movie receives more than \$2M in its opening week.

Definition 2.1. [Target Class] Given an RDF graph $\mathcal{G}$, a *target class* is a distinguished IRI of type *rdfs:Class* in $\mathcal{G}$. For example, Movie.

Definition 2.2. [Instances] Given an RDF graph $\mathcal{G}$ and a target class $\mathcal{T}$, the instances of $\mathcal{T}$, denoted $\mathcal{T}(\mathcal{G})$ is the set $\{x : \ (x, \text{rdf:type}, \mathcal{T}) \in \mathcal{G}\}$.

Definition 2.3. [Attribute] Given an RDF graph $\mathcal{G}$ and a target class $\mathcal{T}$, an *attribute* A (of a target class $\mathcal{T}$) is a tuple of IRIs $(p_1, \ldots, p_n)$ such that the domain of p_1 is $\mathcal{T}$, the range of p_i is the domain of p_{i+1}, and the range of p_n is a literal. For example, (hasActor, foaf, yearOfBirth). We also refer the range of the attribute A as the range of p_n.

Definition 2.4. [Attribute Graph] Given an instance x of the target class $\mathcal{T}$ in the RDF graph $\mathcal{G}$ and an attribute $A = (p_1, \ldots, p_n)$, the attribute graph of the instance x, denoted by $A(x)$,

is the union of the sets of triples that match the Basic Graph Pattern [Prud'ommeaux and Seaborne (2008)]

$$((x, p_1, ?v_1) \text{ AND } (?v_1, p_2, ?v_2) \text{ AND } \ldots \text{ AND } (?v_{n-1}, p_n, ?v_n)) \tag{2.1}$$

where $v_i \in V$ are variables.

Given an additional literal value a, we also define a filtered attributed graph, denoted $A(x, a)$, which includes the filter constraint FILTER$(?v_n = a)$ in the graph pattern (2.1). Further, if $\mathcal{A}$ is a tuple of attributes $(A_1, \ldots, A_n)$, then we define $\mathcal{A}(x)$ to be $(A_1(x), \ldots, A_n(x))$

Definition 2.5. [Target Attribute] Given an RDF graph $\mathcal{G}$ and a target class $\mathcal{T}$, a *target attribute* is a distinguished attribute denoted by C. For example, (openingReceipts).

$C(x)$ is intended to describe the *class label* of the instance x, hence we assume that each instance has exactly one class label, i.e., $|C(x)| = 1$ for every $x \in \mathcal{T}(\mathcal{G})$. Given a target attribute $C = (p_1, \ldots, p_n)$, we define $v(C, x)$ to be the value of $?v_n$ matched by the graph pattern (2.1).

Definition 2.6. [Class Label] Given a *target attribute* $C = (p_1, \ldots, p_n)$, the set of *class labels* is the the range of p_n. For brevity we denote this set by $\mathcal{C}$.

Definition 2.7. [RDF Dataset] An *RDF dataset* D is a tuple $(\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ where $\mathcal{G}$ is an RDF graph, $\mathcal{T}$ a target class in $\mathcal{G}$, $\mathcal{A}$ a tuple of attributes, and C is a target attribute. We also denote the tuple $(\mathcal{T}, \mathcal{A}, C)$ as $Desc(D)$ corresponding to the descriptor of the dataset.

Definition 2.8. [Induced Attribute Graph Dataset] Given an RDF dataset $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$, its *induced attribute graph dataset*, denoted $I(D)$, is defined as $\{(\mathcal{A}(x), v(C, x)) : \ x \in \mathcal{T}(\mathcal{G})\}$.

We now formalize the the problem of learning from RDF data.

Problem 2.9. Given an RDF dataset $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ and its induced attribute graph dataset $I(D)$, a hypothesis class H, and a performance criterion P, the learning algorithm L outputs a classifier $h \in H$ that optimizes P. The input to the classifier h is $\mathcal{A}(x)$ where x is an instance of a target class $\mathcal{T}$, and the output $h(x) \in \mathcal{C}$ is a class label.

2.3 Learning from RDF data

We reduce the problem of learning from RDF data to the problem of learning from multiset attribute data which is defined below. This reduction allows for application of algorithms for learning from multiset attribute data (e.g. Relational Bayesian Classifier [Neville et al. (2003b)]) to this setting. Given an RDF dataset $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ and its induced attribute graph dataset $I(D)$, consider an attribute A and the attribute graph $A(x)$ of an instance $x \in \mathcal{T}(\mathcal{G})$. The attribute graph $A(x)$ can be viewed as a directed acyclic graph (DAG) rooted in x, and here we are interested in only the leaves of this DAG. The following definition captures this notion.

Definition 2.10. [Leaf] Given an attribute A_i, we define the leaf function $\mathcal{L}(A_i(x))$ that returns the multiset of leaves of $A_i(x)$, such that each leaf $a \in A_i(x)$ is replaced with n copies of a where n is the number of unique paths from x to a. For brevity we write $\mathcal{L}(A_i(x))$ as $\mathcal{L}_i(x)$ and $\mathcal{L}(A_i(x, a))$ as $\mathcal{L}_i(x, a)$.

Also, we overload the leaf function on a tuple of attributes $\mathcal{A} = (A_1, \ldots, A_n)$ by $\mathcal{L}(\mathcal{A}(x)) = (\mathcal{L}_1(x), \ldots, \mathcal{L}_n(x))$. Using the leaf function, we reduce $I(D)$ into a multiset attributed dataset $\mathcal{M}(D) = \{(\mathcal{L}(\mathcal{A}(x)), v(C, x)) : \ x \in \mathcal{T}(\mathcal{G})\}$. To learn from $\mathcal{M}(D)$ we focus our attention on Relational Bayesian Classifiers (RBC) motivated from modeling relational data [Neville et al. (2003b)]. RBC assumes that attribute multisets are independent given the class, and the most probable class of an instance is given by:

$$h_{RBC}(x) = \underset{c \in \mathcal{C}}{\operatorname{argmax}}\, p(c) \prod_i p(\mathcal{L}_i(x) : \ c) \tag{2.2}$$

Several methods to estimate the probabilities $p(\mathcal{L}_i(x) : \ c)$ are described by Neville et al. (2003b):

- Aggregation: $\hat{p}_{\text{agg}}(\mathcal{L}_i(x) : \ c) = \hat{p}(\text{agg}(\mathcal{L}_i(x)) : \ c)$, where agg is an aggregation function such as min, max, average for continuous attributes; and mode for discrete attributes.
- Independent Value: $\hat{p}_{\text{ind}}(\mathcal{L}_i(x) : \ c) = \prod_{a \in \mathcal{L}_i(x)} \hat{p}(a : \ c)$, which assumes each value in the multiset is independently drawn from the same distribution (attribute value independence).

- Average Probability: $\hat{p}_{\text{avg}}(\mathcal{L}_i(x) : c) = \frac{\sum_{a \in \mathcal{L}_i(x)} \hat{p}(a: c)}{|\mathcal{L}_i(x)|}$, which also assumes attribute value independence as in *Independent Value*, however during inference the probabilities are averaged instead of multiplied.

For estimating the parameters in (2.2), we assume that the learner does not have access to the RDF graph $\mathcal{G}$ but instead only has knowledge $\mathcal{T}, \mathcal{A}$, and C. In addition, we assume that the RDF store answers statistical queries over the RDF graph $\mathcal{G}$ which in our setting correspond to aggregate SPARQL queries submitted to a SPARQL endpoint. Given a descriptor $Desc(D) = (\mathcal{T}, \mathcal{A}, C)$ where $C = (c_1, \ldots, c_m)$ we assume that the RDF store supports the following type of primitive queries:

(Q1) $S(\mathcal{G}, \mathcal{T}) = |\mathcal{T}(\mathcal{G})|$, the number of instances of target type $\mathcal{T}$ in $\mathcal{G}$. This corresponds to the SPARQL query:

```
SELECT COUNT(*) WHERE { ?x rdf:type <T> . }
```

(Q2) $S(\mathcal{G}, \mathcal{T}, C = c) = |\{x \in \mathcal{T}(\mathcal{G}) : v(C, x) = c\}|$, the number of instances of target type $\mathcal{T}$ in which the target attribute takes the class label c. This corresponds to the SPARQL query:

```
SELECT COUNT(*) WHERE {
  ?x rdf:type <T> .
  ?x <c1> ?c1 .  ...  ?cm-1 <cm> c .
}
```

(Q3) $S(\mathcal{G}, \mathcal{T}, C = c, A_i) = \sum_{x \in \mathcal{T}(\mathcal{G}) \text{ and } v(C,x)=c} |\mathcal{L}_i(x)|$. Assuming the attribute $A_i = (p_1, \ldots, p_j)$ this corresponds to the SPARQL query:

```
SELECT COUNT(*) WHERE {
  ?x rdf:type <T> .
  ?x <c1> ?c1 .  ...  ?cm-1 <cm> c .
  ?x <p1> ?v1 .  ...  ?vj-1 <pj> ?vj .
}
```

(Q4) $S(\mathcal{G}, \mathcal{T}, C = c, A_i = a) = \sum_{x \in \mathcal{T}(\mathcal{G}) \text{ and } v(C,x)=c} |\mathcal{L}_i(x, a)|$. Assuming the attribute $A_i = (p_1, \ldots, p_j)$ this corresponds to the SPARQL query:

```
SELECT COUNT(*) WHERE {
  ?x rdf:type <C> .
  ?x <c1> ?c1 .  ...  ?cm-1 <cm> c .
  ?x <p1> ?v1 .  ...  ?vj-1 <pj> a .
}
```

(Q5) $S(\mathcal{G}, \mathcal{T}, C = c, A_i, \text{agg}, [v_l, v_h])$.

```
SELECT COUNT(*) WHERE {
  { SELECT (agg(?vj) AS ?aggvalue) WHERE {
      ?x rdf:type <T> .
      ?x <c1> ?c1 .  ...  ?cm-1 <cm> c .
      OPTIONAL { ?x <p1> ?v1 .  ...  ?vj-1 <pj> ?vj . }
    } GROUP BY ?x
  } FILTER(?aggvalue >= vl && ?aggvalue <= vh) }
```

We now proceed to describe how an RBC can be built using the supported SPARQL queries without requiring access to the underlying dataset. The RBC estimates the following probabilities from training data:

1. $\hat{p}(c)$
2. $\hat{p}(\text{agg}(A_i) : c)$ for each attribute A_i where aggregation is used to estimate probabilities. For simplicity, we discretize the aggregated values and predetermine the bins prior to learning. Hence, we estimate $\hat{p}(\text{agg}(A_i) \in [v_l, v_h] : c)$ for each bin $[v_l, v_h]$
3. $\hat{p}(a : c)$ where a is in the range of A_i, for each attribute A_i where independent value or average probability is used to estimate the probabilities.

The above three probabilities can be estimated (using Laplace correction for smoothing) as follows:

1. $\hat{p}(c) = \frac{S(\mathcal{G},\mathcal{T},c)+1}{S(\mathcal{G},\mathcal{T})+m}$ where m is the number of class labels

2. $\hat{p}(\text{agg}(A_i) \in [v_l, v_h] : c) = \frac{S(\mathcal{G},\mathcal{T},C=c,A_i,\text{agg},[v_l,v_h])+1}{S(\mathcal{G},\mathcal{T},c)+m}$ where m is the number of bins (ranges)

3. $\hat{p}(a : c) = \frac{S(\mathcal{G},\mathcal{T},C=c,A_i=a)+1}{S(\mathcal{G},\mathcal{T},C=c,A_i)+m}$ where a is in the range of A_i and m is the size of range of A_i

Hence, it is possible to learn RBCs from an RDF graph by interacting with the RDF store only through SPARQL queries. This approach does not require access to the underlying dataset and in most practical settings requires much less bandwidth as compared to transferring the data to a local store for processing (see Section 2.5).

2.4 Updatable Models

In many settings, the RDF store undergoes frequent updates i.e., addition or deletion of sets of RDF triples. In such settings, it is necessary to update the predictive model to reflect the changes in the RDF store used to build the model. While in principle, the algorithm introduced in Section 2.3 can be re-executed each time there is an update to the RDF store, it is of interest to explore more efficient solutions for incrementally updating the RBC model by updating only the relevant statistics.

Given a dataset D and a learning algorithm L, let $L(D)$ be a predictive model built from the dataset D. Let θ be a primitive query required over the dataset D to build $L(D)$.

Definition 2.11. [Updatable Model [Koul et al. (2010)]] Given datasets D_1 and D_2 such that $D_1 \subseteq D_2$, we say that a primitive query θ is updatable iff we can specify functions f and g such that:

1. $\theta(D_2) = f(\theta(D_2 - D_1), \theta(D_1))$

2. $\theta(D_1) = g(\theta(D_2), \theta(D_2 - D_1))$

We say that the predictive model constructed using L is updatable iff all primitive queries required over the dataset D to build $L(D)$ are updatable.

The following propositions show that the primitive query $(Q1)$ of the RBC model is updatable, whereas the rest of the queries are not updatable. Hence, in general, the RBC model is not updatable.

Proposition 2.12. *The primitive query $S(\mathcal{G}, \mathcal{T})$ is updatable.*

Proof. This query counts the number of instances of target type $\mathcal{T}$ in $\mathcal{G}$, which is the cardinality of $\{x : (x, \text{rdf:type}, \mathcal{T}) \in \mathcal{G}\}$. Since $\mathcal{G}_1 \subseteq \mathcal{G}_2$ we have $S(\mathcal{G}_2, \mathcal{T}) = S(\mathcal{G}_2 - \mathcal{G}_1, \mathcal{T}) + S(\mathcal{G}_1, \mathcal{T})$, and also $S(\mathcal{G}_1, \mathcal{T}) = S(\mathcal{G}_2, \mathcal{T}) - S(\mathcal{G}_2 - \mathcal{G}_1, \mathcal{T})$. □

Proposition 2.13. *The primitive query $S(\mathcal{G}, \mathcal{T}, C = c, A = a)$ is not updatable.*

Proof. We prove by showing a counter example. Let the target class be $\mathcal{T}$, the target attribute be $C = (c_1)$, an attribute $A = (p_1, \ldots, p_i, \ldots, p_n)$, and suppose we have the following RDF graphs: $S_1 = \{(x, \text{rdf:type}, \mathcal{T}), (x, c_1, c)\}$, $S_2 = \{(x, p_1, o_1), \ldots, (o_{i-1}, p_i, o_i)\}$, and $S_3 = \{(o_i, p_{i+1}, o_{i+1}), \ldots, (o_{n-1}, p_n, a)\}$. Suppose the graph before update is $\mathcal{G}_1 = S_1 \cup S_2$ and after an insertion of S_3 the graph becomes $\mathcal{G}_2 = S_1 \cup S_2 \cup S_3$. For brevity let $\theta(\mathcal{G}) = S(\mathcal{G}, \mathcal{T}, C = c, A = a)$. We will show that there exists no functions f for the query $\theta(\mathcal{G}_2)$, which counts the total number of leaves of $A(x, a)$ such that x has the class label c. In $\mathcal{G}_2$ the attribute graph $A(x)$ is $S_2 \cup S_3$, and hence $\theta(\mathcal{G}_2) = 1$. However, $A(x)$ is partitioned among S_2 and S_3, so $\theta(\mathcal{G}_1) = 0$ and $\theta(\mathcal{G}_2 - \mathcal{G}_1) = 0$, therefore in this case $f(\theta(\mathcal{G}_2 - \mathcal{G}_1), \theta(\mathcal{G}_1)) = f(0, 0) = 1 = \theta(\mathcal{G}_2)$. Now consider another case where initially the graph is $\mathcal{G}_3 = S_1$ and after insertion of S_3 the graph becomes $\mathcal{G}_4 = S_1 \cup S_3$. In this case we have $\theta(\mathcal{G}_4) = 0$, $\theta(\mathcal{G}_3) = 0$, and $\theta(\mathcal{G}_4 - \mathcal{G}_3) = 0$, and so $f(0, 0) = 0$. Since a function can not map an input to more than one output, this shows that there exists no function f to maintain the query result. □

Similarly, it can be shown that the primitive queries $S(\mathcal{G}, \mathcal{T}, C = c)$, $S(\mathcal{G}, \mathcal{T}, C = c, A)$, and $S(\mathcal{G}, \mathcal{T}, C = c, A, \text{agg}, [v_l, v_h])$ are not updatable.

Corollary 2.14. *RBC model is not updatable.*

The proof of Proposition 2.13 shows that when an attribute graph is partitioned across multiple updates, there exists no function to update the required counts. This raises the question as to whether we can ensure updatability by requiring that each update involves only complete attribute graphs. However, this requirement is not sufficient for the query to be updatable. To see why, consider $\mathcal{G}_1 = \{(x, \text{rdf:type}, \mathcal{T}), (x, c_1, c), (x, p_1, o), (o, p_2, a)\}$ and $\mathcal{G}_2 - \mathcal{G}_1 = \{(y, \text{rdf:type}, \mathcal{T}), (y, c_1, c), (y, p_1, o), (o, p_2, b)\}$, then $f(0, 1) = 2$. The extra count from $\theta(\mathcal{G}_2)$

is due to o being shared between two datasets despite the fact that each attribute graph is complete. This motivates the restriction of not allowing the update to *reuse* certain subjects or objects. We formalize this notion as follows.

Definition 2.15. [Clean Update] Assume $\mathcal{G}_1 \subseteq \mathcal{G}_2$, and let $V(\mathcal{G}) = \{s : (s,p,o) \in \mathcal{G}\} \cup \{o : (s,p,o) \in \mathcal{G}\}$ denote the set of all subjects and objects of an RDF graph $\mathcal{G}$. An update (from $\mathcal{G}_1$ to $\mathcal{G}_2$ by insertion, or from $\mathcal{G}_2$ to $\mathcal{G}_1$ by deletion) is said to be clean if $[\forall (s,p,o) \in \mathcal{G}_2][s \notin V(\mathcal{G}_1) \cap V(\mathcal{G}_2 - \mathcal{G}_1)]$. That is, triples in $\mathcal{G}_2 - \mathcal{G}_1$ share objects with only the leaves of attribute graphs in $\mathcal{G}_1$.

Proposition 2.16. *RBC models are updatable if every update is clean.*

Proof. Let D_1 and D_2 be two RDF datasets such that $D_1 \subseteq D_2$. We first consider the primitive query $\theta(\mathcal{G}) = S(\mathcal{G}, \mathcal{T}, C = c, A = a)$. Since every update is clean, the attribute graphs $A(x)$ for all attributes in A, and all instances $x \in \mathcal{T}(\mathcal{G}_1)$ and $x \in \mathcal{T}(\mathcal{G}_2 - \mathcal{G}_1)$ remain the same after insertion (or deletion). Hence, $\mathcal{M}(D_2) = \mathcal{M}(D_1) \cup \mathcal{M}(D_2 - D_1)$ and similarly $\mathcal{M}(D_1) = \mathcal{M}(D_2) - \mathcal{M}(D_2 - D_1)$ for the multiset attributed dataset reductions. It follows that $\theta(\mathcal{G}_2) = \theta(\mathcal{G}_1) + \theta(\mathcal{G}_2 - \mathcal{G}_1)$ and $\theta(\mathcal{G}_1) = \theta(\mathcal{G}_2) - \theta(\mathcal{G}_2 - \mathcal{G}_1)$. Similar argument also holds true for the other queries used for learning a RBC. □

Thus, RBC model can be updated incrementally in a restricted setting where every update is clean in the sense defined above. When clean updates are not available, RBC models can still be incrementally updated if we are willing to sacrifice some accuracy; and rebuild the model periodically by querying the entire RDF store, with the frequency of rebuild chosen based on the desired tradeoff between computational efficiency and model accuracy. Regardless of whether the RBC model is updatable or not, answering of aggregate queries from RDF stores answering can be optimized using an aggregate view maintenance algorithm [Hung et al. (2005)]. Since we assume that the data descriptor does not change as frequently as the data, the aggregate queries needed by the RBC model can be set up and maintained as views on the RDF store.

2.5 Communication Complexity

In this section, we analyze the communication complexity, i.e., the amount of data transfer needed to build an RBC model. We compare the communication complexity of building an RBC model from RDF data in the following two scenarios: (i) posing statistical queries needed for learning the model against a remote RDF store which is the approach proposed in this paper; and (ii) retrieving the entire RDF dataset from a remote RDF store for local processing.

Given an RDF dataset $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ where $\mathcal{A} = (A_1, \ldots, A_n)$. Suppose the RDF store holds the RDF graph $\mathcal{G}$, and let $|\mathcal{G}|$ denote the size of this graph. The communication complexity in scenario (ii) is simply $O(|\mathcal{G}|)$. We now analyze the communication complexity in scenario (i). Let l_C denote the length of tuple C, let r_C denote the size of range of C, and let l_A denote the maximum length of an attribute tuple. Also let r_A^1 denote the maximum number of bins of those attributes estimated by aggregation, let r_A^2 denote the maximum size of range of the remaining attributes, and we define r_A to be $\max(r_A^1, r_A^2)$.

The size of query expressed in SPARQL, is $O(1)$ for $(Q1)$, $O(l_C)$ for $(Q2)$, and $O(l_C + l_A)$ for $(Q3)$, $(Q4)$, and $(Q5)$. Further, to estimate the probabilities to build an RBC, the following number of calls for each query described in Section 2.3 are required:

(Q1) one.

(Q2) r_C, once for each class label.

(Q3) $r_C \cdot n$, once for each class label and each attribute.

(Q4) $O(r_C \cdot n \cdot r_A)$, once for each class label, each attribute, and each value of the attribute.

(Q5) $O(r_C \cdot n \cdot r_A)$, same as $(Q4)$.

Therefore, the total complexity is $O(1) + O(l_C r_C) + O((l_C + l_A) r_C n) +$
$O((l_C + l_A) r_C \cdot n \cdot r_A) + O((l_C + l_A) r_C \cdot n \cdot r_A) = O((l_C + l_A) r_C \cdot n \cdot r_A)$. In Section 2.7.1 we provide results of experiments which show that $O((l_C + l_A) r_C \cdot n \cdot r_A)$ is usually less than $O(|\mathcal{G}|)$ in practice.

2.6 Selective Attribute Crawling

In previous sections we have considered the problem of learning RBCs given an RDF dataset $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C)$ in the setting where the learner has direct access to $\mathcal{T}, \mathcal{A}$, and C, but not $\mathcal{G}$. Here we consider a more general problem where the learner does not have a priori knowledge of $\mathcal{A}$. This requires the learner to interact with the RDF store containing $\mathcal{G}$ in order to determine $\mathcal{A}$ (e.g. by crawling and selecting attributes) that best optimizes a predetermined performance criterion P. Since the number of attributes in an RDF store can be arbitrarily large we specify an additional constraint Z to guarantee termination (e.g. number of attributes crawled, number of queries posed, time spent, etc.).

Problem 2.17. Given an RDF dataset without attributes, $D = (\mathcal{G}, \mathcal{T}, C)$, a hypothesis class H, a performance criterion P, and constraint Z, the learning algorithm L outputs the following while respecting Z: (i) The selected tuple of attributes $\mathcal{A}$, and (ii) a classifier $h \in H$ that optimizes P.

For simplicity, we focus the setting where the constraint Z specifies the maximum the number of attributes crawled. We consider the problem of identifying $\mathcal{A}$ of cardinality at most Z so as to optimize P. This problem is a variant of the well-studied feature subset selection problem [Liu and Motoda (1998); Guyon and Elisseeff (2003)], albeit in a setting where the set of features is a priori unknown. Identifying attributes one at a time to optimize P can be seen as a search over a tree rooted at $\mathcal{T}$, where the edges are IRIs of properties and the nodes are the domains/ranges of properties, and an attribute corresponds to a path from the root to an RDF literal (a leaf in this tree). To complete the specification of the search problem, we need to specify operations for expanding a node to generate its successors and define the scoring function for evaluating nodes. Expanding a node consists of querying (i) the set of distinct properties outgoing from a node, (ii) the range of each property, and (iii) the type of each range (e.g. numeric, string, non-literal), each of which can be expressed as s SPARQL query. We define the score of a node based on the degree of correlation of the node with the target attribute C. Specifically, for each attribute (represented by a leaf), we compute mutual information [Cover and Thomas (1991)] between it and the the target attribute C. The score of an internal node is defined (recursively)

as a function of its descendants, e.g. average of the scores of its children.

Formally, the score of an attribute A is:

$$Score(A) = \sum_{C=c, A=a} p(A=a, C=c) \log_2 \frac{p(A=a, C=c)}{p(A=a)p(C=c)} \tag{2.3}$$

These probabilities can also be estimated based on the queries described in Section 2.3. Given this framework, a variety of alternative search strategies can be considered, along with several alternative scoring functions.

2.7 Experiments

We conduct three experiments each with a different goal. The first measures the communication complexity using the LinkedMDB [Hassanzadeh and Consens (2009)] dataset. The second experiment combines the US Census dataset with a government dataset to evaluate the accuracy of models using different attribute crawling strategies. Finally we demonstrate learning of RBC from another government dataset through a live SPARQL endpoint.

2.7.1 Communication Complexity Experiment

The goal of this experiment is to measure the communication complexity under two different approaches described in Section 2.5.

2.7.1.1 Dataset and Experiment Setup

The IMDB dataset is a standard benchmark that has been used to evaluate probabilistic relational models including RBCs [Neville et al. (2003b)]. The task is to predict whether a movie receives more than $2M in its opening week. We used LinkedMDB [Hassanzadeh and Consens (2009)], which is an RDF store extracted from IMDB, with links to other datasets on the Linked Open Data cloud [Cyganiak and Jentzsch (2011)]. We used links to Freebase[1] which includes $foaf$ property to the $Person$ class and three properties of class $Person$. Figure 2.1 shows the RDF schema of the extracted dataset. Since LinkedMDB does not have openingReceipts, we

[1] http://www.freebase.com

add them by crawling the IMDB website[2]; also for the Freebase data, we parse the yearOfBirth property for each Person from the dateOfBirth property. We extract 20 movies which are released after 2006 such that each movie has at least one actor, one director, and one producer. The target class is $Movie$ and the target attribute is ($openingReceipts$). We consider a total of 10 attributes: ($runtime$), and $(h, foaf, a)$ where $h \in \{hasActor, hasDirector, hasProducer\}$ and $a \in \{yearOfBirth, gender, hasAward\}$.

To show the growth of data transfer, we prepared 20 subsets of the dataset by corresponding to 1 to 20 movies. A movie instance consists of the URI of the movie and all reachable linked data for it. For communication complexity of learning RBC from RDF stores using statistical queries, we used the proposed approach to build an RBC for each subset and logged the SPARQL queries sent, saved the log in a plain text format, and measured the size of the logs. We compared the results with the communication complexity of learning RBC by first retrieving the data from a remote store for local processing as measured by the size of the corresponding dataset in RDF/XML format on disk.

2.7.1.2 Results

Figure 2.2 shows that the size of the raw data exceeds that of the query when there are more than three movie instances in the dataset. We also considered the case where the RDF store compresses the raw data before transfer, and in this case the size of the compressed raw data exceeds that of the query when there are more than 90 movie instances.

2.7.2 Selective Attribute Crawling Experiment

The goal of this experiment is to evaluate the accuracy of RBC models built using different attribute crawling strategies. Recall that in this setting the learner is only given a SPARQL endpoint of the RDF store, the target class, and the target attribute.

[2] http://www.imdb.com

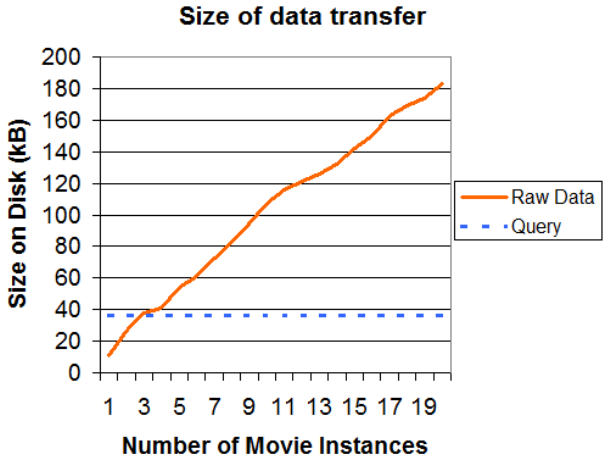

Figure 2.2 Comparison of size of data transfer for Experiment 2.7.1

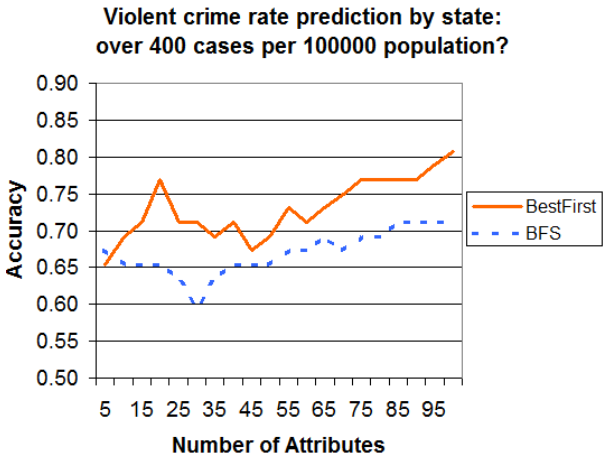

Figure 2.3 Comparison of two crawling strategies for Experiment 2.7.2

2.7.2.1 Dataset and Experiment Setup

In this experiment we use datasets from Data.gov and US census 2000. The target class is 52 US states and we wish to predict whether a state's violent crime rate is over 400 per 100,000 population, which is from dataset 311 of the Data-gov project [Ding et al. (2010a)]. We link this with the US Census 2000 dataset for the corresponding states. This dataset was converted to over 1 billion RDF triples by Tauberer (2011). Part of its RDF schema is shown in [Tauberer (2011)]. It uses a property as a way to sub-divide the population, and a number at a leaf represents the population that satisfies the conditions (properties) on the path from root. In our experiment we normalize by dividing every number of a state by the state's total population.

We vary the maximum number of attributes to be crawled. We set the constraint to be the number of attributes the learner is allowed to crawl. We apply two different attribute crawling strategies (described below) separately and build RBC models using the crawled attributes. To measure the accuracy of the built models, we randomly partition 52 states into 13 groups (of 4 states each) and perform cross validation. That is, for each group, the 4 states in the group are held out and used for prediction, and the remaining are used for training the model. The overall accuracy is the total number of correct predictions divided by the number of states (52).

We experiment with two crawling strategies: BreadthFirst (BFS) and BestFirst. BFS chooses the node with the least depth to expand and BestFirst chooses the node that has the highest score as defined in Section 2.6.

2.7.2.2 Results

As shown in Figure 2.3, BestFirst outperforms BFS with the exception of the case where the number of attributes is 5. We examined the crawled attributes for BestFirst from for choices of Z from 20 to 45, and found that the strategy focused on expanding the household property. This is because attribute selection is guided by mutual information between a candidate attribute and the target class. The sub-divisions of this property may provide very minimal additional information compared to the first one crawled in this group, and hence they may not contribute to the predictive accuracy. One way to circumvent this problem is to use a scoring function to that measures the amount of *information gain* resulting from a candidate attribute given all the attributes that have already been chosen. Another approach is to penalize the attributes based on the depth of search. A third approach is to use the marginal improvement in the accuracy of the RBC classifier resulting from inclusion of the attribute to decide whether to retain it. Other alternatives worth exploring include different search strategies such as Iterative Deepening Search (IDS, [Korf (1985)]).

2.7.3 Live Demonstration

The goal of this experiment is to demonstrate learning of RBC from a government dataset through a live SPARQL endpoint[3] hosted on Rensselaer Polytechnic Institute [Ding et al. (2010b)]. This endpoint supports aggregate and nested queries proposed in SPARQL 1.1.

2.7.3.1 Dataset and Experiment Setup

We used a Health Information National Trends Survey (HINTS, [Nelson et al. (2004)]) from NCI which has been converted into RDF as part of the Data-gov project [Ding et al. (2010a)]. The survey represents a cross-sectional study of health media use and cancer-related knowledge among adults in the United States, and it has been used by [Ackerson and Viswanath (2009)] to study associations of covariates with different smoking statuses. There are 12080 participants across two years (2003 and 2005), represented by 623544 total number of RDF triples, and the raw RDF data (as TTL dump) has a size 35.9MB on disk. The task in our setting is to predict the smoking status (never, former, or current) of a participant from 16 other attributes such as race, sex, household income, and education. The dataset is propositional in nature although represented in RDF format; that is, every attribute has exactly one value in terms of the reduced multiset attribute data, hence the task reduces to learning of a conventional Naive Bayes classifier. Nevertheless, the experiment demonstrates learning of RBC from large and remote RDF store by querying its SPARQL endpoint.

2.7.3.2 Results

A total of 159 queries were posed to the live SPARQL endpoint, and the model was learned in approximately 30 secs, using 2.8 GHz processor with 4 GB memory, and the network download and upload speed is approximately 3 Mbps.

[3] http://logd.tw.rpi.edu/sparql

2.8 Conclusion

2.8.1 Summary

The emergence of RDF as a basic data representation format for Semantic Web has led to increasing availability of all kinds of data in RDF. Transforming this data into knowledge calls for approaches to learning predictive models from massive RDF stores in settings where (i) the learning algorithm can interact with the data store only through a SPARQL endpoint; (ii) the model needs to be updated in response to updates to the underlying RDF store; and (iii) the attributes that can be used to build the predictive models are not known a priori and hence need to be identified by crawling the RDF store. We have introduced an approach to learning predictive models from RDF stores in such settings using Relational Bayesian Classifiers (RBCs) as an example. We have implemented our solutions in an open source system available as part of the INDUS toolkit for learning predictive models from massive data sets [Koul and Lin (2013)] and demonstrated the its feasibility using experiments with several RDF datasets.

2.8.2 Related Work

The work on SPARQL-ML [Kiefer et al. (2008)] extends SPARQL with data mining support to build classifiers, including statistical relational models such as RBC from RDF data. Other works on learning predictive models from RDF data include [Bicer et al. (2011)] and [Tresp et al. (2009)]. Bicer et al. (2011) defined kernel machines over RDF data where features are constructed by ILP-based dynamic propositionalization. Tresp et al. (2009) represented RDF triples as entries in a Boolean matrix, and matrix completion methods are used to train the model and predict unknown triples off-line. However, all the approaches assume that the learner has direct access to RDF data. In contrast, our approach does not require the learning algorithm to have direct access to RDF data, and relies only on the ability of the RDF store to answer aggregate SPARQL queries.

CHAPTER 3. LEARNING CLASSIFIERS FROM RDF DATA WITH SUBCLASS HIERARCHIES

As shown in Chapter 2, the massive size and distributed nature of LOD cloud present a challenging machine learning problem where the data can only be accessed *remotely*, i.e. through a query interface such as the SPARQL endpoint of the data store. In this chapter we further extend our work and consider incorporating other ontological information that may be available in an RDF store. Specifically, existing approaches to learning classifiers from RDF data in such a setting fail to take advantage of RDF schema (RDFS) associated with the data store that asserts subclass hierarchies which provide information that can potentially be exploited by the learner. Against this background, in this chapter we present ProbAVT, an algorithm for learning classifier from a remote RDF data store enriched with subclass hierarchies. ProbAVT encodes the constraints specified in a subclass hierarchy using latent variables in a directed graphical model, and adopts the Variational Bayesian EM approach to efficiently learn parameters. We conduct experiments with several real world datasets and show that ProbAVT achieves equal or better performance compared to other state-of-art models that incorporate subclass hierarchies, and is able to scale up to large hierarchies over few thousand nodes.

3.1 Introduction

RDF data stores often have associated RDF schema (RDFS) that provide additional information about RDF data. RDFS encode subclass hierarchies that provide background knowledge. There is a body of work which shows that background knowledge in the form of hierarchical groupings of attribute values can be exploited to produce compact yet accurate classifiers, and to minimize over-fitting from sparse data [Zhang et al. (2005); Caragea et al. (2009); Kiefer

et al. (2008); Rettinger et al. (2009)].

Against this background, we introduce ProbAVT, an algorithm for learning classifier from a remote RDF data store in settings where the RDF data (ABox in the Semantic Web parlance) is too large to be downloaded across the Internet by the learner and/or to fit in memory, but RDFS (TBox in the Semantic Web parlance) is not. ProbAVT encodes the constraints specified in a subclass hierarchy using latent variables in a directed graphical model, and adopts the Variational Bayesian EM approach to efficiently learn parameters. Our experiments with several real world datasets show that: (i) not surprisingly, ProbAVT outperforms its counterpart that does not incorporate background knowledge in the form of subclass hierarchies; (ii) ProbAVT achieves equal or better performance compared to other state-of-art models that incorporate subclass hierarchies, and is able to scale up to large hierarchies over few thousand nodes. We have implemented and incorporated ProbAVT into INDUS [Koul and Lin (2013)], an open source suite of statistical query based learning algorithms for learning predictive models from massive data.

The rest of the chapter is organized as follows: Section 3.2 formulates the problem of learning classifiers from RDF data and the associated RDFS subclass hierarchies. Section 3.3 presents a solution to incorporate subclass hierarchies using latent variables in a directed graphical model, and adapts the Variational Bayesian EM approach for parameter learning. Section 3.4 presents results of experiments on several real world. Finally Section 3.6 concludes with a summary of of the key results and a brief discussion of related work.

3.2 Problem Formulation

In this chapter, we assume that an RDF graph may include subclass hierarchies, i.e. it includes the following set of reserved predicates: `rdfs:subClassOf` (`sc`), `rdfs:domain` (`dom`), `rdfs:range` (`range`), and `rdf:type` (`type`).

Definition 3.1. Given an RDF graph $\mathcal{G}$, the TBox denoted by $\mathcal{G}_T$ is the subset of $\mathcal{G}$ defined by $\{(s,p,o) \in \mathcal{G} : p \in \{\texttt{sc}, \texttt{dom}, \texttt{range}\}\} \cup \{(s, \texttt{type}, o) \in \mathcal{G} : o \in \{\texttt{rdfs:Class}, \texttt{rdfs:Property}\}\}$.

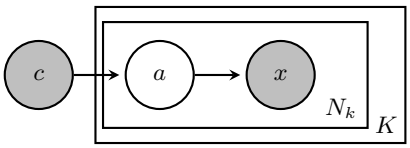

Figure 3.1 A graphical model representation of the proposed model to learn with subclass hierarchies.

Definition 3.2. Given an RDF graph $\mathcal{G}$, the ABox of $\mathcal{G}$ denoted by $\mathcal{G}_A$ is $\mathcal{G} \setminus \mathcal{G}_T$.

Let R be an `rdfs:Class`, we denote $Sub(R)$ to be the set of all subclasses of R (including R itself). In this work we further assume that the range of each attribute A_k can be asserted by a subclass hierarchy R_k, and we denote $\mathcal{R} = (R_1, \ldots, R_K)$. We introduce the problem of learning from RDF data with subclass hierarchies.

Problem 3.3. [Learning from RDF data with subclass hierarchies] Given a subclass hierarchy annotated RDF data set $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, C, \mathcal{R})$, a hypothesis class H, and a performance criterion P, the learning algorithm L outputs a classifier $h \in H$ that optimizes P.

3.3 Learning Classifiers with Subclass Hierarchies

We use the Leaf function (2.10) to reduce each attribute graph into a multi-set, and transform D into a *multiset attributed dataset* $\mathcal{M}(D) = \{c(x), (\mathcal{L}(A_1(x)), \ldots, \mathcal{L}(A_K(x))) : x \in \mathcal{T}(\mathcal{G})\}$, the multi-set of leaves of the attribute graph $A(x, \gamma)$ when it is viewed as a directed acyclic graph rooted in x.

We begin with modeling $\mathcal{M}(D)$ using a generative process that incorporates the abstraction provided by subclass hierarchies, this is achieved by introducing a latent variable to each observed value in a multi-set; its graphical model is depicted in Figure 3.1 and the full joint distribution is given by (3.1) where N_k is the cardinality of $\mathcal{L}(A_k(x))$, and both x_{kn} and a_{kn} take a value from all nodes in the subclass hierarchy R_k.

$$p(c, \mathbf{a}, \mathbf{x}) = p(c) \prod_{k=1}^{K} \prod_{n=1}^{N_k} p(a_{kn} \mid c)\, p(x_{kn} \mid a_{kn}) \tag{3.1}$$

We encode the subclass hierarchy of an attribute A_k by placing the following constraint: $p(x_{kn} \mid a_{kn}) = 0$ if x_{kn} is not a descendent of a_{kn} in the subclass hierarchy rooted at R_k. We

intend to interpret a_{kn} as the *abstraction* that represents the observed value x_{kn}, and then use $p\left(a_{kn} \mid c\right)$ to model the class conditional probabilities as in naive Bayes. Alternatively, we can write $p\left(a_{kn} \mid c, x_{kn}\right) \propto p\left(c\right) p\left(a_{kn} \mid c\right) p\left(x_{kn} \mid a_{kn}\right)$ to describe the *distribution* of abstractions that best represent a value x_{kn} given class, which depends on the parameters learned from the dataset. To learn those parameters, we take the Variational Bayesian approach as in [Beal and Ghahramani (2006)] where we wish to approximate the log marginal likelihoods given by:

$$\ln p\left(c, \mathbf{x} \mid m\right) = \ln \int d\boldsymbol{\theta}\, p\left(\boldsymbol{\theta} \mid m\right) \prod_{i=1}^{I} \sum_{\mathbf{a}_i} p\left(c_i, \mathbf{a}_i, \mathbf{x}_i \mid \boldsymbol{\theta}\right) \tag{3.2}$$

We derive the lower bound by applying Jensen's inequality via variational distributions $q_{\boldsymbol{\theta}}\left(\boldsymbol{\theta}\right)$ and $\{q_{\mathbf{a}_i}\left(\mathbf{a}_i\right)\}_{i=1}^{I}$.

$$\ln p\left(c, \mathbf{x} \mid m\right) \geq \int d\boldsymbol{\theta}\, q_{\boldsymbol{\theta}}\left(\boldsymbol{\theta}\right) \ln \frac{p\left(\boldsymbol{\theta} \mid m\right)}{q_{\boldsymbol{\theta}}\left(\boldsymbol{\theta}\right)} + \sum_{i=1}^{I} \int d\boldsymbol{\theta}\, q_{\boldsymbol{\theta}}\left(\boldsymbol{\theta}\right) \sum_{\mathbf{a}} q_{\mathbf{a}_i}\left(\mathbf{a}_i\right) \ln \frac{p\left(c_i, \mathbf{a}_i, \mathbf{x}_i \mid \boldsymbol{\theta}, m\right)}{q_{\mathbf{a}_i}\left(\mathbf{a}_i\right)} \tag{3.3}$$

Let N_{ca} be the expected total number of times variable $A = a$ when its parent $C = c$, similarly let N_{ax} be the expected total number of times variable $X = x$ when its parent $A = a$. We also let N_{cx} be the number of times $C = c$ and $X = x$. The corresponding variational E-step and M-step can be derived as follows [Beal and Ghahramani (2006)].

Variational E-step:

$$q_{\mathbf{a}_i}\left(\mathbf{a}_i\right) \propto p\left(c, \mathbf{a}, \mathbf{x} \mid \boldsymbol{\theta}\right) \tag{3.4}$$

Variational M-step:

$$\ln \tilde{\boldsymbol{\theta}}_{ca} = \psi\left(\lambda_{ca} + N_{ca}\right) - \psi\left(\sum_{a} \lambda_{ca} + N_{ca}\right) \tag{3.5}$$

$$\ln \tilde{\boldsymbol{\theta}}_{ax} = \psi\left(\lambda_{ax} + N_{ax}\right) - \psi\left(\sum_{x} \lambda_{ax} + N_{ax}\right) \tag{3.6}$$

where $\psi(\cdot)$ is the digamma function.

Table 3.1 Accuracy results of three UCI datasets, using 10-fold cross validation with 95% confidence interval.

Dataset	NB	NB-AVT	ProbAVT
Mushroom	95.83 (±0.43)	99.85 (±0.08)	99.45 (±0.16)
Soybean	92.09 (±2.02)	94.73 (±1.68)	94.44 (±1.72)
Nursery	90.32 (±0.51)	90.32 (±0.51)	90.32 (±0.51)

Since in the E-step, every a_{kn} is independent of each other, we can simply maintain N_{cax} that counts the number of times $C = a$, $A = a$, and $X = x$, which can be iteratively updated by using the parameters in M-step and N_{cx} obtained from data. In the M-step, both N_{ca} and N_{ax} can be derived from N_{cax}. Hence, N_{cax} can be initialized and maintained locally, and the only sufficient statistic required from data is N_{cx}.

3.4 Experiments

We present two sets of experimental results, one using datasets from UCI repository and another using RDF datasets.

3.4.1 UCI datasets with Subclass Hierarchy

We use three datasets (with only discrete attributes) from UCI repository (i.e., Mushroom, Soybean, and Nursery) where the subclass hierarchies are supplied by domain experts. These datasets correspond to a special case in our setting where $N_k = 1$ for all k, and the goal of this set of experiments is to compare the proposed method (denoted by ProbAVT) against an alternative model proposed by Zhang et al. (2006) that extends naive Bayes with subclass hierarchies (denoted by NB-AVT), as well as a plain naive Bayes as a baseline (denoted by NB).

Table 3.1 shows the accuracy results of 10-fold cross validation. We observe that ProbAVT compares favorably with NB-AVT, while they both outperform NB in Mushroom and Soybean dataset.

3.4.2 RDF datasets with Subclass Hierarchy

3.4.2.1 Public Contract Dataset

We use Task 2 dataset of the Linked Data Mining Challenge appeared in Data Mining on Linked Data Workshop 2013[1]. This RDF dataset contains heterogeneously integrated data of UK public contracts using various vocabularies and ontologies such as Public Contracts Ontology[2], Simple Knowledge Organization System (SKOS)[3], DBpedia[4], etc. The task of the challenge is to classify the contracts as multi-contract or not, and the training set have 48 multi-contracts out of a total of 216 contracts. A multi-contract is a contract that unifies two or more unrelated commodities. In this experiment we only extract four discrete attributes: (i) Kind, with range services/supplies/works; (ii) MainObject, with a range described by a large SKOS vocabulary, and we interpret skos:broaderTransitive as a subclass hierarchy of 4539 nodes; (iii) AdditionalObject, with the same range as mainObject; and (iv) Location, encoded using NUTS (Nomenclature of territorial units for statistics[5]) classification, with four levels of hierarchy consists of 197 nodes.

The extracted dataset also corresponds to a special case where $N_k = 1$ for all k. The dataset is relatively small in size however it is described by several large subclass hierarchies, and NB-AVT is no longer tractable hence omitted from the results.

Table 3.2 shows the results of this experiment under 10-fold cross validation. Despite the small size and difficulty of this dataset, ProbAVT still shows moderate improvements over NB: ProbAVT achieves similar accuracy with NB, while having a lower precision and a significant higher recall; in terms of ROC ProbAVT has about 7% higher area under curve.

3.4.2.2 Web Service Dataset

We use an RDF benchmark dataset from OWLS-TC v2.1[6] service retrieval test collection that contains 578 OWL-S Semantic Web service descriptions. The attributes are input and

[1] http://keg.vse.cz/dmold2013
[2] http://opendata.cz/public-contracts-ontology
[3] http://www.w3.org/TR/skos-reference/
[4] http://dbpedia.org
[5] http://epp.eurostat.ec.europa.eu/portal/page/portal/nuts_nomenclature/introduction
[6] http://projects.semwebcentral.org/projects/owls-tc/

Table 3.2 Classification results of Public Contract dataset, using 10-fold cross validation with 95% confidence interval.

Performance Measure	NB	ProbAVT
Accuracy	81.25 (±5.21)	81.73 (±5.15)
Precision	66.67 (±6.29)	55.00 (±6.63)
Recall	5.00 (±2.91)	27.50 (±5.95)
ROC	62.60 (±6.45)	69.90 (±6.12)

output concepts described by various subclass hierarchies (and even richer ontologies), and the dataset corresponds to the general case where $N_k \geq 1$, hence we compare our method against two other relational methods: a relational Bayesian classifier [Neville et al. (2003b)] (denoted RBC) and a modified relational probability tree [Neville et al. (2003a)] that incorporates subclass hierarchies (introduced as part of SPARQL-ML toolkit [Kiefer et al. (2008)] and hence we denote it by SML)

Table 3.3 shows four performance measures (FP rate, precision, recall, f-measure) for each class under 10-fold cross validation. We observe that ProbAVT outperforms SML in all measures, and outperform RBC in all measures except precision.

3.5 Discussion

We also implemented ML/MAP EM to compute a point estimate of the parameters, and as in [Beal and Ghahramani (2006)] we observe that the ML/MAP alternative can be easily trapped in a local minimum depending on the initial conditions. Variational Bayesian EM on the other hand computes a distribution over parameters and naturally incorporates a model complexity penalty, hence offers an explanation of superior and more stable performance.

Our approach of using latent variables to encode subclass hierarchies provides an alternative to the global cut framework proposed by Zhang et al. (2006). A cut on a hierarchy is a subset of nodes such that every leaf of the hierarchy is a descendant of some member in the cut. Loosely speaking, a cut defines a *hard* abstraction over a hierarchy, whereas our proposed method defines a *soft* and more general abstraction using latent variables, recall that we write $p\left(a_{kn} \mid c, x_{kn}\right)$ to describe the *distribution* of abstractions that best represent a value x_{kn} given class. One

Table 3.3 Classification results of Semantic Web Service dataset

Class	FP Rate			Precision			Recall			F-measure		
	RBC	SML	ProbAVT	RBC	SML	ProbAVT	RBC	SML	ProbAVT	RBC	SML	ProbAVT
communication	0.4	0.4	0.0	93.1	90.0	100.0	93.1	60.0	96.6	93.1	72.0	98.2
economy	12.9	1.8	8.1	81.1	96.4	87.0	100.0	88.9	97.6	89.6	92.5	92.0
education	0.2	9.0	1.8	99.2	71.6	93.7	88.1	86.9	87.4	93.3	78.6	90.4
food	0.0	0.2	0.2	100.0	96.0	95.5	72.0	80.0	84.0	83.7	87.3	89.4
medical	0.0	3.0	0.6	100.0	68.8	93.2	55.8	55.0	78.8	71.6	61.1	85.4
travel	0.2	6.9	0.0	99.0	74.4	100.0	95.3	87.3	94.3	97.1	80.3	97.1
weapon	0.2	0.2	0.7	96.2	96.4	85.2	100.0	90.0	92.0	98.0	93.1	88.5
average	2.0	3.1	**1.6**	**95.5**	84.8	93.5	86.3	78.3	**90.1**	89.5	80.7	**91.6**

disadvantage of learning a soft abstraction is that it can be less comprehensible, although it may be of interest to derive an optimal *closest* cut from a learned soft abstraction such that they give comparable performance. Another major difference between these two methods is that a cut is seen as a model structure where as latent variables introduce parameters within the model, and they present different learning challenges. Structure learning for the cut framework requires a search over the cut space, which Zhang et al. (2006) uses the conditional log likelihood to define the cut refinement criterion; however, introducing latent variables can be intractable to estimate the marginal likelihood of parameters. Despite these challenges, our experimental results show that adopting the Variational Bayesian EM approach [Beal and Ghahramani (2006)] is an efficient solution to our problem and can be scaled to large hierarchies over few thousand nodes. Furthermore with the proposed simple dependency structure (Figure 3.1), Variational Bayesian EM can be executed using only a sufficient statistic from data, which can be queried remotely without direct access to data.

3.6 Conclusion

3.6.1 Summary

Rapid growth of RDF data in the Linked Open Data (LOD) cloud offers unprecedented opportunities for analyzing such data using machine learning algorithms. The massive size and distributed nature of LOD cloud present a challenging machine learning problem where the data can only be accessed *remotely*, i.e. through a query interface such as the SPARQL endpoint of the data store. Existing approaches to learning classifiers from RDF data in such a setting fail to take advantage of RDF schema (RDFS) associated with the data store that asserts subclass hierarchies which provide information that can potentially be exploited by the learner. Against this background, we present ProbAVT, an algorithm for learning classifier from RDF data and the associated schema. ProbAVT encodes the constraints specified in a subclass hierarchy using latent variables in a directed graphical model, and adopts the Variational Bayesian EM approach to efficiently learn parameters. Our experiments with several real world datasets show that: (i) not surprisingly, ProbAVT outperforms its counterpart that does not incorporate

background knowledge in the form of subclass hierarchies; (ii) ProbAVT achieves equal or better performance compared to other state-of-art models that incorporate subclass hierarchies, and is able to scale up to large hierarchies over few thousand nodes. We have implemented and incorporated ProbAVT into INDUS [Koul and Lin (2013)], an open source suite of statistical query based learning algorithms for learning predictive models from massive data.

3.6.2 Related Work

ProbAVT extends Chapter 2 to exploit background knowledge in the form of RDFS subclass hierarchies, and yet still learns without direct access to RDF data.

Other works that incorporate subclass hierarchy include SPARQL-ML [Kiefer et al. (2008)] that extends SPARQL with data mining support to build statistical relational models from RDF and RDFS data, the global cut framework proposed by Zhang et al. (2006, 2005), and a link-based text data proposed by Caragea et al. (2009). However, these works assume that the learner has direct access to all RDF data (both TBox and ABox). In contrast, our approach only assumes that the learner has direct access to the ontology (TBox) associated with the data which is usually much smaller than ABox, and relies on SPARQL queries to access the instance data (ABox).

ProbAVT can be seen as a topic model [Blei and Lafferty (2009); Blei (2012)] or a directed graphical model with hidden variables [Beal and Ghahramani (2006)]. In fact it is also a special case of Infinite Semantic Hidden Models [Rettinger et al. (2009)] which incorporates expressive ontologies (i.e. $\mathcal{SHOIN}(D)$) to learning predictive models by encoding logic rules as constraints in the variables of Hidden Markov Models. In our case we only encode subclass hierarchies using the constraint $p\left(x_{kn} \mid a_{kn}\right) = 0$ if x_{kn} is not a descendent of a_{kn}, however this simplicity allows the model to be learned with sufficient statistics and scales to large hierarchies over few thousand nodes.

3.6.3 Future Work

It is of interest to visualize the soft abstraction learned from data such that it is more comprehensible, and identify conditions for which they can relate to the hard abstractions in

the global cut framework proposed by Zhang et al. (2006, 2005). Another direction of future work includes modeling of dependencies between multiple attributes, and one way towards this goal is to adapt a multi-modal topic model [Balasubramanyan and Cohen (2011); Chang et al. (2009); Nallapati et al. (2008)] in our setting such that the model can be learned or approximated with sufficient statistics so that direct access to data is not required. It is also interesting to enrich the expressivity of ontologies (e.g. subproperty hierarchy) within the same setting.

CHAPTER 4. LEARNING CLASSIFIERS FROM DISTRIBUTIONAL DATA

In this chapter we set aside our motivation of learning from RDF data (or linked data in general) and investigate a novel type of learning problem which we call *distributional instance classification*. We motivate, precisely formulate, and present solutions for this problem. We will resume our setting of linked data in Chapter 5 where we will solve a more challenging problem that utilizes the results presented in this chapter.

Many data mining applications give rise to *distributional data* wherein objects or individuals are naturally represented as K-tuples of *bags of feature values* where feature values in each bag are sampled from a *feature and object specific* distribution. We formulate and solve the problem of learning classifiers from distributional data. We consider three classes of methods for learning distributional classifiers: (i) those that rely on *aggregation* to encode distributional data into tuples of attribute values, i.e., instances that can be handled by traditional supervised machine learning algorithms; (ii) those that are based on *generative models* of distributional data; and (iii) the discriminative counterparts of the generative models considered in (ii) above. We compare the performance of the different algorithms on real-world as well as synthetic distributional data sets. The results of our experiments demonstrate that classifiers that take advantage of the information available in the distributional instance representation outperform or match the performance of those that fail to fully exploit such information.

4.1 Introduction

The standard classification problem entails assigning an instance x from an instance space $\mathcal{X}$ (that is typically modeled by a tuple of measurements or attribute values) a label from a set

of mutually exclusive classes $\mathcal{C}$. A classifier h is a mapping $h : \mathcal{X} \mapsto \mathcal{C}$. The goal of learning in such a setting is to identify a classifier from a space of classifiers $\mathcal{H}$, one that optimizes a desired performance measure, e.g., accuracy of the classifier. Consider for example, a clinical diagnosis scenario which calls for classifying a patient as healthy or suffering from a particular illness based on a set of tests or measurements. Suppose the k^{th} feature or test result takes values from the domain Δ_k. The standard approach is to represent each patient o_i by a K-tuple or a K-dimensional vector of feature values $x_i = (x_{i1}, \ldots, x_{iK}) \in \Delta_1 \times \Delta_2 \times \cdots \Delta_K$ (where each $x_{ik} \in \Delta_k$) that encode the results of specific medical tests or measurements [Bishop (2006)]. The goal is to predict the class label $c_i \in \{\mathit{Healthy}, \mathit{Ill}\}$ for each patient o_i. However, because of the variability associated with physiological measurements such as the heart rate, blood pressure, or body temperature of an individual, it is often necessary to repeat the tests or measurements in order to arrive at a reliable diagnosis. If the measurements are synchronous, then it is possible to represent each patient by a collection or bag of instances (K-tuples of feature values) and model the problem of predicting the class label for each patient as a multiple instance learning problem [Dietterich et al. (1997); Zhou (2004)]. However, because the different tests or measurements have different sources of variability associated with them, it is not uncommon to carry out the tests in an asynchronous fashion, with each test repeated different number of times. Hence, as illustrated in Table 4.1, it is far more meaningful to model the input to the classifier, in this case, an individual o_i, by a K-tuple of bags (multi-sets) of feature values $(B_1^i, \ldots, B_K^i)$ where each B_k^i represents a bag of values of the k^{th} feature of object o_i, sampled from the specific feature and individual specific distribution. Note that, in general, the size of the bag B_k^i can differ from feature to feature and for a given feature, from one object to another.

Many big data applications give rise to *distributional data* wherein objects or individuals are naturally represented as K-tuples of *bags of feature values* where feature values in each bag are sampled from a *feature and object specific* distribution. We refer to the resulting representation $x_i = (B_1^i, \ldots, B_K^i)$ of an object o_i as the *distributional instance* (DI) representation of o_i. We refer to the problem of learning classifiers that predict the class labels of distributional instances as the *distributional instance learning* (DIL) problem. One way to apply traditional approaches to classification in this setting is to simply replace each bag of feature values B_k^i by

Table 4.1 An example of a distributional data set of three patients with their four attributes where status represents the class label.

Status	Body Temperature in Fž	Heart Rate in BPM	Blood Pressure Systolic in mmHg	Blood Pressure Diastolic in mmHg
Healthy	{98.5, 98.8, 99.0}	{70, 68}	{100, 106}	{75, 77}
Ill	{99.2, 100.3, 98.6, 99.0, 98.4}	{80, 75, 83, 76}	{120, 116, 126}	{76, 76, 83}
Healthy	{98.6}	{61, 69}	{95}	{65}

an aggregate value, e.g., the mean or mode computed from the observed values of the feature for a given individual. However, such an aggregation process can result in significant loss of useful information. It is much more natural to view the input to the classifier as a K-tuple of bags of attribute values. Against this background, we formulate and solve the DIL problem.

We consider representative algorithms from three classes of methods for DIL: (i) those that rely on *aggregation* to encode distributional data into tuples of attribute values, i.e., instances that can be handled by traditional supervised machine learning algorithms; (ii) those that are based on *generative models* of distributional data; and (iii) the discriminative counterparts of the generative models considered in (ii) above. We compare the performance of the different algorithms on two real-world as well as one synthetic distributional data sets. The results of our experiments demonstrate that DIL algorithms that take advantage of the information available in the distributional instance representation outperform or match the performance of their counterparts that fail to fully exploit such information. We conclude with a brief summary of the main results, discussion of related work, and some directions for further research.

4.2 Distributional Instance Classification Problem

We introduce some key definitions before proceeding to formulate the problem of learning classifiers from distributional data. For brevity subscripts are omitted when they are clear from context.

Definition 4.1 (Distributional Instance Representation)**.** Let $\Delta_1, \ldots, \Delta_K$ be K sets (discrete of continuous) that correspond to the domains of a finite number (K) of measurable attributes of objects to be classified, and $\mathcal{C}$ a finite set of class labels. A Distributional Instance representation x_i of an object or individual o_i is a K-tuple of bags (multi-sets) of feature values $x_i = (B_1^i, \ldots, B_K^i)$ where each B_k^i represents a bag of values of the k^{th} feature of object o_i, sampled from the specific feature and individual specific distribution. We denote by s_k the size of k^{th} domain, $s_k = |\Delta_k|$.

Note that the widely used bag of words representation of text is a special case of distributional representation where $K = 1$ and Δ_1 is simply the vocabulary of the document collection.

Definition 4.2. Let $c_i \in \mathcal{C}$ be the class label of x_i. A *distributional data set* $D = \{(x_1, c_1), \ldots, (x_n, c_n)\}$ is a multi-set of labeled *distributional instances*.

Example 4.3. Table 4.1 shows an example of a distributional data set consisting of three objects (patients).

Definition 4.4 (Distributional Classifier)**.** Each classifier h accepts as input, a distributional instance x, and outputs a predicted class label $h(x) \in \mathcal{C}$.

Definition 4.5 (Distributional Classifier Learning Problem)**.** Given a distributional data set D, a hypothesis class $\mathcal{H}$ of classifiers, and a performance criterion f, a distributional classifier learning algorithm L outputs a *distributional classifier* $h \in \mathcal{H}$ that optimizes f.

4.3 Distributional Instance Learning Algorithms

We consider three basic approaches to DIL: (i) those that rely on *aggregation* to encode distributional data into tuples of attribute values, i.e., instances that can be handled by traditional supervised machine learning algorithms; (ii) those that are based on *generative models* of distributional data; and (iii) the discriminative counterparts of the generative models considered in (ii) above.

4.3.1 Aggregation

Here we represent each bag of features in the DI representation of an instance by a single value, by applying a suitable aggregation function, e.g., *min*, *max*, *average* for continuous Δ and *mode* for discrete Δ. Hence we reduce the data set into a traditional attribute-value data set where each instance is represented by a finite number of attributes each of which takes a single value from the set of possible values for the corresponding attribute. This approach reduces the problem of learning from distributional data to the standard supervised learning problem which can be solved using a variety of existing supervised learning algorithms [Bishop (2006); Murphy (2012)].

Within this framework, we consider a variety of sophisticated aggregation schemes proposed by Perlich and Provost (2006). Without loss of generality, consider a distributional data set D

in which the distributional instances are encoded using DI representation and class labels are binary, i.e., $\mathcal{C} = \{+, -\}$. Suppose that B_k^i is the bag of values of k^{th} attribute of an instance x_i. After Perlich and Provost (2006), we define $V_k^i = \left(v_{k1}^i, \ldots, v_{ks_k}^i\right)$ to be a vector of counts (or histogram) of values in B_k^i where v_{kt}^i is the number of occurrences of the t^{th} value $d_{kt} \in \Delta_k$. Next we define an unconditional reference vector as $V_k^{(*)} = \sum_i V_k^i$, and also a class-conditional reference vector for $c \in \mathcal{C}$ as $V_k^{(c)} = \sum_i \delta_{c,c_i} V_k^i$ where δ is a Kronecker delta function. A number of aggregation schemes can be defined using various measures of distance between V_k^i and the reference vectors [Perlich and Provost (2006)] (see below). Let $DIST$ be a set of M distance functions between two vectors such as cosine or Euclidean, then we describe three aggregation schemes as follows.

1. Unconditional vector distances (UCVD): We compute an M-element vector $(dist_m(V_k^{(*)}, V_k^i))_{m=1}^M$ where $dist_m \in DIST$. We concatenate the feature vector representations from each of the K bags of features of x_i to obtain a single feature vector of length MK.

2. Class-conditional vector distances (CCVD): We compute a $|\mathcal{C}|M$-sized vector, e.g., $(dist_m(V_k^{(+)}, V_k^i),\ dist_m(V_k^{(-)}, V_k^i))_{m=1}^M$. The rest follows the scheme of UCVD and reduces into a traditional attribute-value data set where each instance is a vector of length $|\mathcal{C}|MK$.

3. Differences of class-conditional vector distances (DCCVD): We compute the pair-wise difference between every two class-conditional vector distances, resulting with a vector of size $M|\mathcal{C}|(|\mathcal{C}| - 1)/2$, e.g., $(dist_m(V_k^{(+)}, V_k^i) - dist_m(V_k^{(-)}, V_k^i))_{m=1}^M$.

By applying this process to each of the distributional instances in the data set D, we can effectively reduce the problem of learning distributional classifiers to the well-studied problem of supervised learning of classifiers in the traditional setting where each object to be classified is represented by a tuple of attribute values.

4.3.2 Generative Models

We consider a joint distribution $p(B_1, \ldots, B_K, c)$. For simplicity, under the naive Bayes assumption that bags of features are conditionally independent given the class label c the most

probable class label is given by:

$$\begin{aligned} h_{NB}(x) &\triangleq \arg\max_{c\in\mathcal{C}} p(c \mid B_1, \ldots, B_K) \\ &= \arg\max_{c\in\mathcal{C}} p(c) \prod_{k=1}^{K} p(B_k \mid c). \end{aligned}$$

We can now consider a variety of models for $p(B_k \mid c)$ including those based on Bernoulli or multinomial event models [McCallum and Nigam (1998)], Dirichlet distribution [Ferguson (1973); Minka (2012)] or Dirichlet-multinomial (Polya) distribution [Madsen et al. (2005); Minka (2012)]. We denote these models by NB(Ber), NB(Mul), NB(Dir), and NB(Pol) respectively, and outline each of them below.

Let $b_{kt} \in \{1, 0\}$ denote the presence or absence of $d_{kt} \in \Delta_k$ in an attribute bag B_k and, similarly, let v_{kt} denote the number of occurrences of d_{kt}. A class-conditional bag probability, $p(B_k \mid c)$, can be modeled by event models such as Bernoulli (4.1) or multinomial (4.2):

$$p(B_k \mid c; \boldsymbol{\theta}) \triangleq \prod_{t=1}^{s_k} \theta_{ckt}^{b_{kt}} (1 - \theta_{ckt})^{1-b_{kt}} \tag{4.1}$$

$$p(B_k \mid c; \boldsymbol{\theta}) \triangleq p(|B_k|) \frac{(\sum_{t=1}^{s_k} v_{kt})!}{\prod_{t=1}^{s_k} v_{kt}!} \prod_{t=1}^{s_k} \theta_{ckt}^{v_{kt}} \tag{4.2}$$

where $\theta_{ckt} = p(d_{kt} \mid c)$.

Next, the Dirichlet distribution (4.3) allows us to treat B_k as a sample from a distribution which, in turn, is drawn from another distribution as follows:

$$\begin{aligned} p(B_k \mid c; \boldsymbol{\alpha}) &\triangleq p\left(\bar{V}_k \mid c\right) \\ &\triangleq \mathcal{D}(\boldsymbol{\alpha}_{ck}) \\ &= \frac{\Gamma(\sum_{t=1}^{s_k} \alpha_{ckt})}{\prod_{t=1}^{s_k} \Gamma(\alpha_{ckt})} \prod_{t=1}^{s_k} \bar{v}_{kt}^{\alpha_{ckt}-1} \end{aligned} \tag{4.3}$$

where $\boldsymbol{\alpha}_{ck} = (\alpha_{ck1}, \ldots, \alpha_{cks_k})$ is a vector parameter of Dirichlet distribution for class $c \in \mathcal{C}$ and $\bar{V}_k = (\bar{v}_{k1} \cdots \bar{v}_{ks_k})$ is the *normalized* vector of counts of values in B_k with $\bar{v}_{kt} = v_{kt} / \sum_t v_{kt}$. Finally, we describe the Dirichlet-multinomial (Polya) distribution (4.4) that compounds a

Dirichlet distribution with a multinomial distribution:

$$\begin{aligned} p\left(B_k \mid c; \boldsymbol{\alpha}\right) &\triangleq p\left(V_k \mid c\right) \\ &\triangleq \int p\left(V_k; \boldsymbol{\theta}_{ck}\right) p\left(\boldsymbol{\theta}_{ck}; \boldsymbol{\alpha}_{ck}\right) d\boldsymbol{\theta}_{ck} \\ &= \frac{\Gamma\left(\sum_t \alpha_{ckt}\right)}{\Gamma\left(\sum_t v_{kt} + \alpha_{ckt}\right)} \prod_{t=1}^{s_k} \frac{\Gamma\left(v_{kt} + \alpha_{ckt}\right)}{\Gamma\left(\alpha_{ckt}\right)} \end{aligned} \tag{4.4}$$

where $\boldsymbol{\theta}_{ck} = (\theta_{ck1}, \ldots, \theta_{cks_k})$ is a vector of multinomial parameters.

For all four models, their parameters, which is a set of parameters for each class and for each attribute, are estimated by maximum likelihood employing the Laplace correction.

4.3.3 Discriminative Models

We consider the discriminative counterparts of the generative models described in Section 4.3.2 using standard techniques for transforming a generative model into its discriminative counterpart (e.g., a naive Bayes model into a logistic regression model [Bouchard and Triggs (2004)]). The details of this derivation is presented in Appendix A. Discriminative models [Minka (2005)] can be acquired by plugging in four different distributions (shown in Section 4.3.2) for $p\left(B \mid c\right)$ in the following equation.

$$p\left(c = 1 \mid x\right) \triangleq \frac{1}{1 + \exp\left(\ln \frac{p(c=0)}{p(c=1)} + \sum_{k=1}^{K} \ln \frac{p(B_k|c=0)}{p(B_k|c=1)}\right)} \tag{4.5}$$

There exists an equivalent parametric form for posterior distribution, $p\left(c \mid x; \mathbf{w}\right)$. We estimate a vector of parameters $\mathbf{w}$ as

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{i=1}^{n} \ln p\left(c_i \mid x_i; \mathbf{w}\right) - \lambda \left|\left|\mathbf{w}\right|\right|_2^2 . \tag{4.6}$$

It is possible to adopt ℓ^2–regularization by setting $\lambda > 0$ to reduce over-fitting to training data. Given the estimated parameter $\mathbf{w}^*$, prediction on a distributional data instance is given by $h\left(x\right) = \arg\max_{c \in \mathcal{C}} p\left(c \mid x; \mathbf{w}^*\right)$. Discriminative models for Bernoulli and multinomial distributions define posterior distribution as below (respectively):

$$p\left(c = 1 \mid x; \mathbf{w}\right) \triangleq \frac{1}{1 + \exp\left(w_0 + \sum_{k,t} b_{kt} w_{kt}\right)}$$

$$p\left(c = 1 \mid x; \mathbf{w}\right) \triangleq \frac{1}{1 + \exp\left(\ln \frac{p(c=0)}{p(c=1)} + \sum_{k,t} v_{kt} w_{kt}\right)}$$

which are logistic regression models. Parameters for these two models can be estimated with optimization tools specialized in logistic regression (e.g., Fan et al. (2008)). For Dirichlet and Polya distributions, we first formulate $p(c \mid x; \boldsymbol{\alpha})$ by substituting $p(B \mid c)$ in (4.5) with (4.3) and (4.4), respectively. By setting $\mathbf{w} = \ln(\boldsymbol{\alpha})$, we drop the constraint $\boldsymbol{\alpha} > 0$ and employ an unconstrained gradient ascent method[1].

4.4 Experimental Results

We report results of experiments designed to answer the following questions.

(i) How do representative DIL algorithms within each of the three classes of DIL methods outlined in Section 4.3 compare with each other?

(ii) How do the three classes of DIL methods compare with each other?

(iii) How do classifiers that take advantage of the information available in the distributional instance representation compare with their counterparts that reduce DIL to traditional supervised learning (by transforming distributional instances into tuples of attribute values)?

We use two real world data sets to address questions (i) and (ii) above, and we use a synthetic data set to address question (iii).

4.4.1 Experiment I

4.4.1.1 Data Sets and Experimental Setup

Due to lack of publicly available distributional data benchmarks, we turn to available data sets that can be modeled as distributional data.

The first data set, the Last.fm data set, is crawled from a social music network Last.fm[2] using its API[3] (an example is shown in Figure 4.1). We select two disjoint groups that contain

[1] In our experiments, we used Hessian-free Newton method implemented by Mark Schmidt. See http://www.di.ens.fr/~mschmidt/Software/minFunc.html

[2] http://www.last.fm/

[3] http://www.last.fm/api

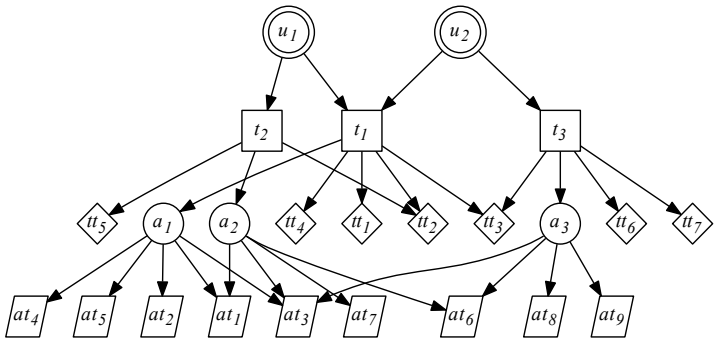

Figure 4.1 A portion of Last.fm data set with entities and links among them. Entities are users u, tracks t, track's tags tt, artists a, and artist's tags at. Corresponding distributional data instances are $x_1 = \{\{t_1, t_2\}, \{a_1, a_2\}, \{tt_1, \ldots tt_5\}, \{at_1, \ldots, at_7\}\}$ and $x_2 = \{\{t_1, t_3\}, \{a_1, a_3\}, \{tt_1, \ldots, tt_7\}, \{at_1, \ldots at_6, at_8, at_9\}\}$.

approximately equal number of users (2098/2081). We collect the *track*, *artist*, *track's tags*, and *artist's tags* favored by each user and represent them as bags. All collections of tags are processed with stop-word removing and stemming, using Apache Lucene[4]. We use only tracks and artists whose number of occurrences greater or equal than 45 and 100, correspondingly. The result is a distributional data set of 8340 tracks attributed to one or more of the 3753 artists. Likewise, we eliminate the track tags and artist tags that occur fewer than 350 times and 120 times respectively.

The second data set is obtained from the Splog (spam blogs) data set which contains 700 authentic and 700 spam blogs in HTML format [Kolari et al. (2006)]. For each blog, we extracted four attributes: a bag of words, a bag of anchors (words marked up with hyperlinks), a bag of URLs, and a bag of HTML tags. Then, similarly we remove infrequent elements for each attribute and remove instances with missing values. The statistics of the two data sets are shown in Table 4.2.

We compare the three classes of DIL described in Section 4.3. Specifically, for the aggregation models we select: (i) *mode* aggregation combined with a simple naive Bayes classifier (denoted Mode+NB); (ii) CCVD combined with a Gaussian naive Bayes (denoted CCVD+NB); and (iii) CCVD combined with a logistic regression (denoted CCVD+LR). We only selected CCVD among the complex aggregation schemes since it is reported to yield more accurate classifier

[4] http://www.luccene.apache.org/

Table 4.2 Data set statistics. Since bags contain duplicate elements, the size of a bag may exceed the domain size.

Data set (+/− count)	Attribute	Domain Size	Average Bag Size
Last.fm (2098/2081)	track	8340	120
	track's tag	5077	15874
	artist	3753	58
	artist's tag	3640	10535
Splog (695/693)	word	7968	2627
	anchor	7819	316
	URL	7833	937
	HTML tag	152	538

by Perlich and Provost (2006). For ℓ^2-regularized discriminative models, we set $\lambda = 1$ in (4.6) without optimization. We evaluate both data sets using 10-fold cross-validation and draw their ROC (Receiver Operating Characteristic) curves with AUC (Area Under Curve).

4.4.1.2 Results

Among the models based on aggregation, CCVD+LR outperforms the rest in both accuracy and AUC measures, thus confirming the conclusions reported by Perlich and Provost (2006). Among generative models, NB(Pol) outperforms the rest on both accuracy and AUC, while NB(Dir) is also competitive for the Last.fm data set. For discriminative models, DM_{ℓ^2}(Pol), DM_{ℓ^2}(Dir), and DM_{ℓ^2}(Ber) are equally competitive on the Last.fm data set; and DM_{ℓ^2}(Pol) outperforms the rest for the Splog data set. The ℓ^2-regularized discriminative models generally outperform their un-regularized counterparts (with the exception of DM_{ℓ^2}(Mul) on the accuracy measure for the Splog data set). Indeed, in this case un-regularized models are special cases of regularized models with the hyperparameter $\lambda = 0$; and in principle, the results of our regularized models could be improved further by optimizing the hyperparameter.

To answer the second question, we compare the best model from each approach. Considering the accuracy measure alone, NB(Pol) and DM_{ℓ^2}(Pol) consistently outperforms the rest for both data sets. We observe that the best models under the AUC measure is a subset of the best models under the accuracy measure, and this is possibly because in general AUC is statistically more

Table 4.3 Results for Experiment I. Each number in parentheses is standard deviation from 10-fold cross-validation. Bolded numbers represent best results for each column based on paired t-test on 10-fold cross validation with alpha = 0.05.

Hypothesis Class	Model	Last.fm		Splog	
		Accuracy	AUC	Accuracy	AUC
Aggregation	Mode+NB	73.01 (2.17)	79.72 (2.20)	69.09 (2.70)	85.62 (1.91)
	CCVD+NB	76.21 (1.76)	82.42 (1.72)	63.49 (4.39)	79.35 (4.54)
	CCVD+LR	**81.55 (2.15)**	**89.64 (1.64)**	86.46 (4.26)	93.00 (3.60)
Generative	NB(Ber)	72.70 (2.76)	82.43 (3.04)	79.04 (3.55)	87.83 (3.57)
	NB(Mul)	**81.98 (2.17)**	86.38 (1.81)	88.69 (2.53)	93.51 (2.97)
	NB(Dir)	**81.65 (2.11)**	**89.52 (1.62)**	78.53 (2.36)	84.07 (3.06)
	NB(Pol)	**82.12 (2.05)**	**88.97 (1.33)**	**88.98 (3.68)**	93.87 (2.52)
Discriminative	DM(Ber)	79.59 (2.69)	87.84 (1.76)	89.34 (2.60)	95.57 (1.41)
	DM(Mul)	76.29 (2.78)	82.60 (2.55)	86.67 (3.46)	92.55 (2.64)
	DM(Dir)	79.47 (2.74)	87.92 (1.81)	89.63 (2.69)	95.87 (1.19)
	DM(Pol)	79.97 (2.46)	87.80 (1.92)	90.56 (2.27)	96.10 (1.36)
	DM_{ℓ^2}(Ber)	**81.36 (1.88)**	89.15 (1.61)	89.70 (2.33)	95.85 (1.37)
	DM_{ℓ^2}(Mul)	80.35 (1.98)	86.54 (1.37)	86.38 (3.19)	95.51 (2.56)
	DM_{ℓ^2}(Dir)	**80.21 (2.82)**	88.36 (1.75)	89.70 (2.49)	95.94 (1.17)
	DM_{ℓ^2}(Pol)	**80.98 (2.36)**	88.78 (1.79)	**91.07 (2.02)**	**96.25 (1.32)**

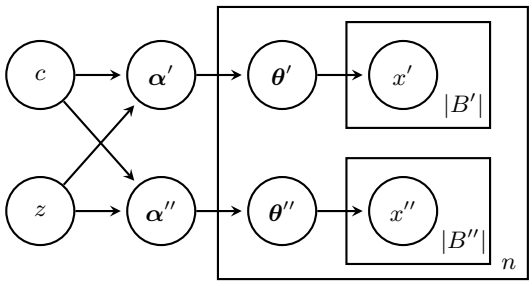

Figure 4.2 Generation process for the synthetic data. We control c and z to generate n data with bags of size $|B| = |B'| + |B''|$. Dirichlet parameter $\boldsymbol{\alpha}'$ and $\boldsymbol{\alpha}''$ is deterministically generated given c and z.

discriminant than accuracy [Huang and Ling (2005)]. If we now consider the AUC measure alone, we observe that NB(Pol), NB(Dir), and CCVD+LR are equally competitive for the Last.fm data set, while NB(Pol) outperforms the rest for the Splog data set. In summary, NB(Pol) and DM$_{\ell^2}$(Pol) show similar performance.

4.4.2 Experiment II

The second experiment is designed to examine as to how the classifiers that take advantage of the information available in the distributional instance representation compare with those that do not fully exploit such information. Recall that among the models described in Section 4.3, only NB(Dir), NB(Pol), DM(Dir), and DM(Pol) model distributions of distributional instances. Since naive Bayes models and discriminative models have the same likelihood $p\,(B \mid c)$, we only consider naive Bayes models in this experiment.

We deliberately crafted scenarios in which some of the models that do not take advantage of the information available in the would fail to discriminate between two classes. For example, Mode+NB would fail if $p(c \mid mod(B))$ is close to 0.5; NB(Mul) would fail if the parameters for both classes (i.e. its sufficient statistics $\bar{V}_k^{(+)}$ and $\bar{V}_k^{(-)}$) are similar; and CCVD would fail if two reference vectors are similar and their distances from the DIL representation of the object to be classified are also similar.

4.4.2.1 Data Set and Experimental Setup

We generated a synthetic data set with binary class ($\mathcal{C} = \{+, -\}$) and a single attribute ($K = 1$) by combining samples from two Polya distributions. The two Polya distributions have domains $\Delta' = \{0, 1\}$ and $\Delta'' = \{2, 3\}$ whose Dirichlet parameters are $\boldsymbol{\alpha}'$ and $\boldsymbol{\alpha}''$ respectively. The generation process is shown in Figure 4.2, where Dirichlet parameters are defined deterministically as follows,

$$\boldsymbol{\alpha}' = \begin{cases} (z, z) & c = + \\ \left(z^{-1}, z^{-1}\right) & c = - \end{cases}, \quad \boldsymbol{\alpha}'' = \begin{cases} \left(z^{-1}, z^{-1}\right) & c = + \\ (z, z) & c = - \end{cases}$$

and we draw a bag B' from Polya distribution as follows,

$$\begin{aligned} \theta' &\sim \text{Dir}\left(\boldsymbol{\alpha}'\right) \\ x' &\sim \text{Mult}\left(\boldsymbol{\theta}'\right) \\ B' &\triangleq \left\{x'_1, \ldots, x'_{|B'|}\right\}. \end{aligned}$$

Similarly, B'' are drawn from Polya distribution with $\boldsymbol{\alpha}''$. Finally the bag B for an instance is defined as $B' \cup B''$. The process generates a set of n instances with a label c where each instance is characterized by z, $|B'|$, and $|B''|$ which we will denote by $f\left(n, c, z, |B'|, |B''|\right)$. A balanced data set with $2n$ instances can be described as $f\left(n, +, z, |B'|, |B''|\right) \cup f\left(n, -, z, |B'|, |B''|\right)$.

We defined three groups of balanced data sets where generations of data sets in a group only differ in z. The first group is balanced data sets with $n = 500$ and $|B'|, |B''| = 40$, and the second group differs by $|B''| = 80$. A data set in the third group is defined as the *union* of two balanced data sets of 1000 instances where their $(|B'|, |B''|)$ are $(40, 80)$ and $(80, 40)$, respectively. Given a fixed z, a data set in the first two groups can be represented as

$$\bigcup_{c \in \{-,+\}} f\left(500, c, z, 40, \left|B''\right|\right)$$

where $|B''|$ is 40 or 80 respectively, while a data set in the third group is

$$\bigcup_{c \in \{-,+\}} f\left(500, c, z, 40, 80\right) \cup f\left(500, c, z, 80, 40\right).$$

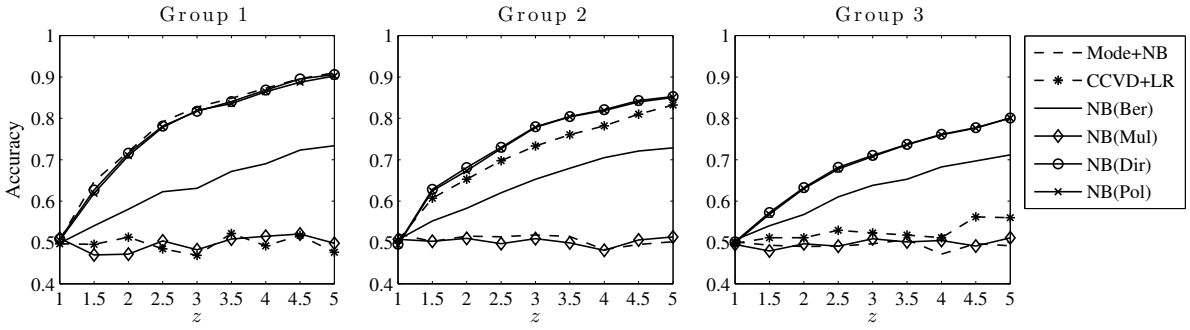

Figure 4.3 Accuracies of six classifiers on different groups of data sets where each synthetic data set consists of samples drawn from a stochastic process which is a composition of two Polya processes.

We repeat this process five times with different random seeds to obtain five different distributional data sets. We estimate the accuracy of the classifiers using 10-fold cross-validation. We report the average accuracy of the DIL methods over the five data sets, in each case, estimated using 10-fold cross-validation.

4.4.2.2 Results

Figure 4.3 shows the results of this experiment. We observe that all naive Bayes models behave similarly in all three groups. In particular, accuracies for NB(Dir), NB(Pol), and NB(Ber) increase as z increases, while NB(Mul) is unable to discriminate between the two classes. NB(Mul) fails because $\bar{V}_k^{(+)}$ and $\bar{V}_k^{(-)}$ are designed to be similar by setting the Dirichlet parameters of both classes to be complement of each other.[5] Performance of NB(Dir) matches with NB(Pol), because the bag length is constant for all instances. As z increases, the good performance of NB(Ber) can be explained by observing it is more likely that the bag of feature values for the negative class contain all 0's or all 1's.

Interestingly, the behaviors of Mode+NB and CCVD+LR vary across three groups of experiments. In the case of Mode+NB, it fails in Group 2 because it is most likely that one of two values of the second bag B'' is chosen as mode for an instance *independent* to the label of the instance[6]; whereas it fails in Group 3 because all values are equally likely to be the mode.

[5]Take B' for example, for the positive class, the bags are likely to contain approximately equal numbers of 0's and 1's whereas for the negative class, the bags are likely to contain either a majority of 0's or a majority of 1's; hence this ensures that the expected multinomial parameters for both classes are $(0.5, 0.5)$.

[6]With a rare chance, mode can be a value of the bag B' if an instance contains three values with 40 counts.

In the case of CCVD+LR, in all three groups, the expected class-conditional reference vectors are identical for both classes. The first and the third group guarantee the expected distance of positive instances and that of negative instances are the same. For the second group, the expected distances differ due to the asymmetry in $|B'|$ and $|B''|$.

Overall, these experiments clearly demonstrate that DIL methods that take advantage of the information available in the distributional instance representation can potentially outperform those that do not fully exploit such information.

4.5 Conclusion

4.5.1 Summary

Many data mining applications naturally give rise to *distributional data* wherein objects or individuals to be labeled are naturally represented as K-tuples of bags of feature values sampled from a *feature and object specific* distribution. We have introduced distributional instance learning problem, i.e., the problem of learning classifiers from distributional data. We have considered three classes of methods for learning such classifiers: (i) those that rely on *aggregation* to encode distributional data into tuples of attribute values, i.e., instances that can be handled by traditional supervised machine learning algorithms; (ii) those that are based on *generative models* of distributional data; and (iii) the discriminative counterparts of the generative models considered in (ii) above. We have compared the performance of the different algorithms on real-world as well as synthetic distributional data sets. The results of our experiments demonstrate that DIL algorithms that take advantage of the information available in the distributional instance representation outperform or match the performance of their counterparts that make use of aggregation schemes that discard such information.

4.5.2 Related Work

The DIL problem is a generalization of (i) the traditional supervised learning problem where an object to be classified is represented by a tuple of attribute values [Bishop (2006)], and (ii)

In other words, B' consists of only one value and B'' consists of even number of two values.

the problem of learning document classifiers and image classifiers using a bag of words representation of documents [McCallum and Nigam (1998)], and bag of visual words representation of images [Tirilly et al. (2008)] (which represent special cases of learning distributional classifiers with $K = 1$).

The problem of learning distributional classifiers is related to the multiple instance learning (MIL, Dietterich et al. (1997); Zhou (2004)), where an object to be classified is represented as a *bag of instances* and each instance is represented by a tuple of feature values. Only the label of the bag is specified in the training set, and MIL assumes that a bag is labeled negative if and only if all of its instances are negative, and a bag is labeled positive if and only if at least one of its instances is positive [Dietterich et al. (1997)]. Since then, recent work on MIL has relaxed the standard MIL assumption to allow all the instances in a bag to contribute to the bag's label. However, unlike MIL which models an object to be classified by a *bag of tuples of feature values*, DIL models an object to be classified as a *tuple of bags of feature values*. Both MIL and DIL reduce to the same problem when the number of features (K) is one.

DIL bears some resemblance to Latent Dirichlet Allocation (LDA, Blei et al. (2003)) or more generally probabilistic topic models [Blei (2012)], which are generative probabilistic models for documents that model each word in a document by a mixture of latent topics (i.e., distributions over a fixed vocabulary). While the topic models are typically learned in an unsupervised setting, a document's topic distribution can be effectively used for document classification as demonstrated by Blei et al. (2003). Supervised topic models such as sLDA proposed by Blei and McAuliffe (2008) can be seen as a special case of DIL where the number of bags (K) in the DI representation of the objects to be classified (in this case, documents) is equal to one. However, recent work on topic models, e.g., Block-LDA [Balasubramanyan and Cohen (2011)], Nubbi [Chang et al. (2009)], and Link-PLSA-LDA [Nallapati et al. (2008)] has begun to explore topic models for objects with multiple features ($K > 1$). Block-LDA models documents where each document contains collections of entities of different types (which can be modeled by different bags that make up a distributional instance in DIL). DIL can be seen as a supervised variant of such topic models, i.e., supervised topic models defined over multiple features ($K > 1$) with discrete domain.

DIL can be used to model learning from relational data [Getoor and Taskar (2007)] and RDF data [Manola and Miller (2004)]. For example, relational Bayesian classifiers [Neville et al. (2003b); Lin et al. (2011)] model an object (nodes in a network) to be classified using bags of values of features from those objects that are related to it via relational links.

4.5.3 Future Work

We have explored only some of the simplest approaches to DIL. It would be interesting to explore DIL models that account for dependencies between feature values within a bag as well as between bags. It would be useful to consider variants of kernel methods (e.g., SVM) that use kernel functions to compute similarity between distributional instances, e.g., adaptations of kernel functions for distributions [Jaakkola and Haussler (1998); Jebara et al. (2004)] to the setting with $K > 1$. Of particular interest in this context are support measure machines (SMM) introduced by Muandet et al. (2012) which extend SVMs by representing distributions as mean embeddings in the reproducing kernel Hilbert space, which allows the application of standard kernel methods for classifying probability distributions. One subtle difference between the DIL formulation in this paper and that of SMMs is that the input to an SMM classifier is a probability distribution whereas the input to a distributional classifier is a K-tuple of bags where each bag is a finite sample drawn from a feature and object specific (albeit unknown) distribution. It would be interesting to consider DIL variants of decision trees, random forests, support vector machines, nearest neighbor classifiers, etc. as well as variants of DIL models and algorithms that can handle ordinal or continuous valued features. Of particular interest are DIL algorithms that can effectively handle massive data sets with billions or trillions of objects and millions or billions of attributes each represented using bags ranging in size from tens of thousands to millions of values.

CHAPTER 5. LEARNING CLASSIFIERS FROM CHAINS OF MULTIPLE INTERLINKED RDF DATA STORES

We extend based on previous chapters and consider the problem of learning predictive models from *multiple interlinked* RDF stores. Specifically we: (i) introduce statistical query based formulations of several representative algorithms for learning classifiers from RDF data; (ii) introduce a distributed learning framework to learn classifiers from multiple interlinked RDF stores that form a chain; (iii) identify three special cases of RDF data fragmentation and describe effective strategies for learning predictive models in each case; (iv) consider a novel application of a matrix reconstruction technique from the field of Computerized Tomography [Herman (2009)] to approximate the statistics needed by the learning algorithm from projections using count queries, thus dramatically reducing the amount of information transmitted from the remote data sources to the learner; and (v) report results of experiments with a real-world social network data set (Last.fm), which demonstrate the feasibility of the proposed approach.

5.1 Introduction

We motivate the problem of learning predictive models from multiple interlinked RDF stores using the scenario shown in Figure 5.1. In this case, one might want to use data from Facebook and New York Times to predict the interest of a user in belonging to a Facebook group, based on the distribution of tags associated with the New York Times news stories that the user has shared with her social network on Facebook. This is an instance of the *node prediction problem* [Bhagat et al. (2011)]. In general, building such predictive models entails using information from multiple interlinked, physically distributed, autonomously maintained RDF stores. In such a setting, it is neither desirable nor feasible to gather all of the data in a centralized location for analysis,

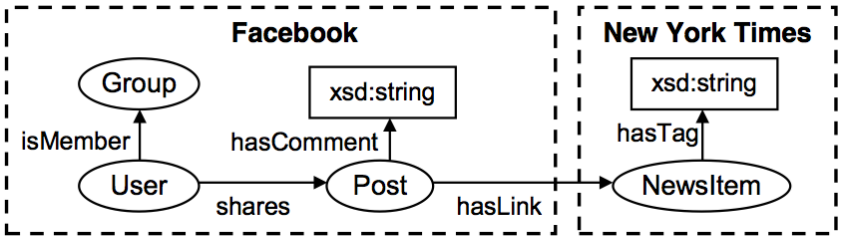

Figure 5.1 A motivating scenario of two RDF stores that are linked to form a *chain* of RDF stores: Facebook users share posts about news items published in New York Times.

because of access, memory, bandwidth, and computational restrictions. In other settings, access to data may be limited due to privacy and confidentiality constraints [Aggarwal and Yu (2008); Wu et al. (2010)]. This calls for techniques for learning predictive models (e.g. classifiers) from multiple interlinked RDF stores that support only *indirect* access to data (e.g. via a query interface such as SPARQL). Barring Lin et al. (2011) who proposed an approach to learning relational Bayesian classifiers [Neville et al. (2003b)] from a *single* remote RDF store using statistical queries against its SPARQL endpoint, to the best of our knowledge, there has been very little work on this problem.

Against this background, we consider the problem of learning predictive models from *multiple interlinked* RDF stores. Specifically we: (i) introduce statistical query based formulations of several representative algorithms for learning classifiers from RDF data; (ii) introduce a distributed learning framework to learn classifiers from multiple interlinked RDF stores that form a chain; (iii) identify three special cases of RDF data fragmentation and describe effective strategies for learning predictive models in each case; (iv) consider a novel application of a matrix reconstruction technique from the field of Computerized Tomography [Herman (2009)] to approximate the statistics needed by the learning algorithm from projections using count queries, thus dramatically reducing the amount of information transmitted from the remote data sources to the learner; and (v) report results of experiments with a real-world social network data set (Last.fm), which demonstrate the feasibility of the proposed approach.

The chapter is organized as follows: Section 5.2 begins with learning classifiers from a single remote RDF store. Section 5.3 extends learning to multiple interlinked RDF stores. Section 5.4 describes the experiments and the results. Finally Section 5.5 concludes with a summary, related

work, and future work.

5.2 Learning Classifiers from RDF Data

5.2.1 RDF Learner Defined

We begin by defining the problem of learning classifiers from a single RDF data source. We recall the formulation from Section 2.2 and adapt these definitions for the setting required by this chapter.

Recall that an RDF triple is $(s, p, o) \in (I \cup B) \times I \times (I \cup B \cup L)$ where s is the subject, p the predicate, and o the object of the triple and I, B, and L are pairwise disjoint infinite sets of URIs, Blank nodes, and Literals respectively. An RDF graph is a set of RDF triples. Given an RDF graph $\mathcal{G}$, and a *target class* $\mathcal{T}$ which is a distinguished URI of type rdfs:Class in $\mathcal{G}$, we denote the set of instances of the target class as $\mathcal{T}(\mathcal{G}) = \{x : \ (x, \text{rdf:type}, \mathcal{T}) \in \mathcal{G}\}$. An *attribute* A (of a target class $\mathcal{T}$) is a tuple of predicates $(p_1, \ldots, p_J)$ such that the domain of p_1 is $\mathcal{T}$, the range of p_j is the domain of p_{j+1}, and the range of p_J is a literal. Given an instance x_i of the target class $\mathcal{T}$ and an attribute $A_k = (p_1^k, \ldots, p_J^k)$, we define B_k^i to be the bag (multi-set) of literals matched by the variable $?v_J$ in the Basic Graph Pattern [W3C SPARQL Working Group] $((x, p_1^k, ?v_1)$ AND $(?v_1, p_2^k, ?v_2) \ldots (?v_{J-1}, p_J^k, ?v_J))$ where $v_j \in V$ are variables. For convenience we denote the range of p_J^k by Δ_k, and let $|\Delta_k| = s_k$. A *target attribute* is a distinguished attribute denoted by A_c, which describes the *class label* of an instance, hence we assume that each instance has exactly one class label, i.e., $\left|B_c^i\right| = 1$ for every $x_i \in \mathcal{T}(\mathcal{G})$; for brevity the class label is denoted by c_i and the set of all possible values of c_i is denoted by $\mathcal{C}$. An *RDF data set* D is a tuple $(\mathcal{G}, \mathcal{T}, \mathcal{A}, A_c)$ where $\mathcal{G}$ is an RDF graph, $\mathcal{T}$ a target class in $\mathcal{G}$, $\mathcal{A} = (A_1, \ldots, A_K)$ a tuple of attributes, and A_c is a target attribute. Given an RDF data set $D = (\mathcal{G}, \mathcal{T}, \mathcal{A}, A_c)$, its induced *multiset attributed data set* [Lin et al. (2011)] is defined as $\mathcal{M}(D) = \{((B_1^i, \ldots, B_K^i), c_i) : \ x_i \in \mathcal{T}(\mathcal{G})\}$.

Definition 5.1. The input to an RDF node classifier h is $(B_1^i, \ldots, B_K^i)$ where x_i is an instance of a target class $\mathcal{T}$, and the output $h(x_i) \in \mathcal{C}$ is a class label.

An RDF Learner L [Lin et al. (2011)] is an algorithm that, given an RDF data set $D =$

$(\mathcal{G}, \mathcal{T}, \mathcal{A}, A_c)$, its induced multiset attributed data set $\mathcal{M}(D)$, a hypothesis class H, and a performance criterion P, outputs a classifier $h \in H$ that optimizes P.

5.2.2 Representative Classes of RDF Learners

We consider two basic approaches to learn from RDF data: (i) those that rely on *aggregation* to encode nodes to be classified as tuples of attribute values, i.e., instances that can be handled by traditional supervised machine learning algorithms; and (ii) those that are based on *generative models* of data. Specifically, we reduce the problem into a Distributional Instance Classification problem (Section 4.2) and adapt the algorithms described in Section 4.3.1 and Section 4.3.2. Note that the discriminative models in Section 4.3.3 were not considered in this chapter because these models generally do not have sufficient statistics that are easily computed, hence they are less suitable in the setting of remote and distributed data sources assumed in this chapter.

5.2.2.1 Aggregation

Here we represent each bag of attributes in $\mathcal{M}(D)$ by a single value, by applying a suitable aggregation function, e.g., *min*, *max*, *average* for continuous values and *mode* for discrete values. Hence we reduce the data set into a traditional attribute-value data set where each instance is represented by a finite number of attributes, each of which takes a single value from the set of possible values for the corresponding attribute. We also consider a more sophisticated aggregation scheme called Class-Conditional Vector Distances (CCVD) proposed by Perlich and Provost (2006) (see Section 4.3.1 for details).

Regardless of which aggregation scheme described above, by applying an aggregation scheme to each of the instances in $\mathcal{M}(D)$, we can effectively reduce the problem of learning from an RDF data set to the well-studied problem of supervised learning in the traditional setting where each instance to be classified is represented by a tuple of attribute values.

5.2.2.2 Generative Models

We consider a joint distribution $p(B_1, \ldots, B_K, c)$. For simplicity, under the naive Bayes (NB) assumption that bags of attributes are conditionally independent given the class label c the most probable class label is given by:

$$\begin{aligned} h_{NB}(x) &\triangleq \arg\max_{c \in \mathcal{C}} p(c \mid B_1, \ldots, B_K) \\ &= \arg\max_{c \in \mathcal{C}} p(c) \prod_{k=1}^{K} p(B_k \mid c). \end{aligned}$$

As described in Section 4.3.2, we can consider a variety of models for $p(B_k \mid c)$ including those based on Bernoulli or multinomial event models [McCallum and Nigam (1998)], Dirichlet distribution [Ferguson (1973); Minka (2012)] or Dirichlet-multinomial (Polya) distribution [Madsen et al. (2005); Minka (2012)] (denoted by NB(Ber), NB(Mul), NB(Dir), and NB(Pol) respectively).

5.2.3 Sufficient Statistics

We describe the sufficient statistics to estimate the parameters (via maximum likelihood) for each attribute A_k and for each of the models in Section 5.2.2, and provide the corresponding SPARQL queries to obtain these statistics.

- Aggregation function: $agg(B_k^i)$ and the class label for each instance x_i where agg is some aggregation function. If naive Bayes is learned on the reduced data set then the following is sufficient: number of instances with the class label c and $d = agg(B_k^i)$ for every combination of c and d. The former can be expressed by an aggregation query and the later is equivalent to $S(\mathcal{G}, \mathcal{T}, C = c, A_k, agg, d)$ by Lin et al. (2011).

- CCVD and NB(Pol): for each $c \in \mathcal{C}$, V_k^i for each instance x_i such that $c_i = c$. Its SPARQL query can be expressed by:

```
SELECT ?x ?vj COUNT(?vj) WHERE {
  ?x rdf:type <T> .
  ?x <classLabel> c .
```

```
  ?x <p1> ?v1 .  ... ?vj-1 <pj> ?vj .
} GROUP BY ?x ?vj
```

- NB(Ber) and NB(Mul): $V_k^{(c)}$ for each $c \in \mathcal{C}$. Its SPARQL query can be expressed by:

```
SELECT ?vj COUNT(?vj) WHERE {
  ?x rdf:type <T> .
  ?x <classLabel> c .
  ?x <p1> ?v1 .  ... ?vj-1 <pj> ?vj .
} GROUP BY ?vj
```

- NB(Dir): for each $c \in \mathcal{C}$, $log(\bar{v}_{kt}) = \sum_i log(v_{kt}^i) - log(\sum_t v_{kt}^i)$ for the subset of instances x_i such that $c_i = c$ (see Minka (2012))[1]. An alternative statistic though not minimal is V_k^i for each instance x_i such that $c_i = c$.

5.2.4 Approximating V_k^i

In Section 5.2.3 we observe that V_k^i (conditioned on some class $c \in \mathcal{C}$) is required for a number of models (CCVD, NB(Dir), and NB(Pol)). However, as shown by the experiment in Section 5.4.4, obtaining V_k^i for each x_i is quite expensive. Instead, it is much cheaper to obtain approximate summaries of each V_k^i. For example, define $V_{kt}^* = \sum_i v_{kt}^i$ and let $V_k^* = \left(v_{k1}^*, \ldots, v_{ks_k}^*\right)$; similarly define $V_{k*}^i = \sum_t v_{kt}^i$ and let $V_{k*} = \left(v_{k*}^1, \ldots, v_{k*}^I\right)$ where I is the total number of instances. In other words, if V_k^i for all $i \in [1, I]$ are represented as an I-by-s_k matrix where v_{kt}^i is the value of row-i and column-t, then V_k^* and V_{k*} are its column and row projections respectively. We say that V_k^* is the projection towards the *leaf* of the attribute chain A_k and hence we refer V_k^* as the *leaf projection*[2]; similarly we refer V_{k*} as the *root projection*. Here we propose to approximate V_k^i from the leaf and root projections in order to save the size of communication.

The problem of reconstructing a matrix from its projections is closely related to the problem of reconstructing a 3D representation of an object from images of its slices, that has been widely

[1] This requires log function which is currently not supported by SPARQL.

[2] Equivalently, $V_k^{(c)}$ is the leaf projection for those x_i where $c_i = c$.

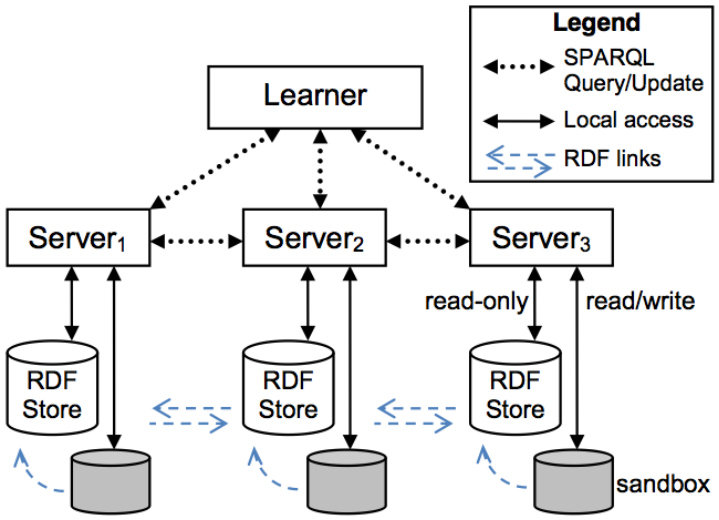

Figure 5.2 Distributed learning framework from multiple interlinked RDF stores. In practice there can be interactions (queries and RDF links) between any two data sources, in the figure only the adjacent interactions are drawn for simplicity.

studied in the field of Computerized Tomography (CT) (see Herman (2009) for a review). The problem of reconstructing V_k^i from its root and leaf projections is a special case of the matrix reconstruction problem. Hence, we can adapt existing approaches from CT, and one of the simplest such methods is Algebraic Reconstruction Technique (ART, Gordon et al. (1970)). ART is an iterative algorithm for solving a system of linear equations where each equation encodes the projection angle and its projected value from a matrix. Here we describe the update equation in our simplified case of column and row projections. Let x_t^i be the element of row-i and column-t of an I-by-T matrix, representing our reconstructed matrix; and let X^* and X_* be its column and row projections respectively. Let V^* and V_* be the true column and row projections respectively (i.e., those computed from the original matrix). Then the update equation is $x_t^i := x_t^i + \lambda \left(\frac{V_*^i - X_*^i}{T} + \frac{V_t^* - X_t^*}{I} \right)$ where λ is a relaxation parameter between 0 and 1.

5.3 Learning Classifiers from Multiple Interlinked RDF Data Stores

We now turn to the problem of learning predictive models from *multiple*, interlinked data sources. Consider the scenario shown in Figure 5.2. We assume that each data source corresponds to an RDF store that can be queried through an access interface (e.g. SPARQL query server), and (optionally) a sandbox that is set up with write access for each user (i.e. learner).

We can use SPARQL 1.1 update queries [W3C SPARQL Working Group] to store intermediate results of queries in the sandbox for use by the learner. Also we can use SPARQL 1.1 federated queries [W3C SPARQL Working Group] to retrieve query results from other remote servers as needed and store them in the sandbox for use by the learner. Thus, an RDF data set D is fragmented across sites $[1, N]$ into data set fragments $D_1, \ldots, D_N$ such that $\bigcup_{n=1}^{N} D_n = D$; we further assume that the learner may be subject to access constraints $Z_1, \ldots, Z_N$ associated with $D_1, \ldots, D_N$ such that $\bigcup_{n=1}^{N} Z_n = Z$. An access constraint may restrict the class of queries that can be answered by a data source, e.g. due to privacy considerations or the query answering capabilities of the data source. The task of learning from multiple interlinked RDF stores can be stated as follows.

Definition 5.2. A Distributed RDF Learner L_d is an algorithm that, given the fragments $D_1, \ldots, D_N$ of a training data set D distributed across the sites $[1, N]$ through a set of access interfaces $A_1, \ldots, A_N$ with access constraints $Z = \bigcup_{n=1}^{N} Z_n$, a hypothesis class H, and a performance criterion P, outputs a classifier $h \in H$ that optimizes P using only the interactions against D that are allowed by Z.

It is useful to consider three generic ways in which an RDF data set can be fragmented across multiple interlinked RDF stores.

5.3.1 Characterizing RDF Data Fragmentation

The simplest case of RDF data set fragmentation corresponds to the setting where there are no links between individual data stores. However, in general, the data stores may contain triples (edges) that link two or more data stores. E.g., a triple (i, c, j) could be in D_1 while (j, c, k) could be in D_2. We refer to the set of all resources that play the role of either the *subject* or the *object* of an RDF triple in a data set as the *resources* of a data set; we use the term *subject resources* to refer to the set of all resources that appear *only* as the subject of an RDF triple in the data set; we use the term *object resources* to refer to the set of all resources that appear *only* as the object of an RDF triple in the data set. We can now identify three special cases of data fragmentation across multiple interlinked RDF stores (Figure 5.3):

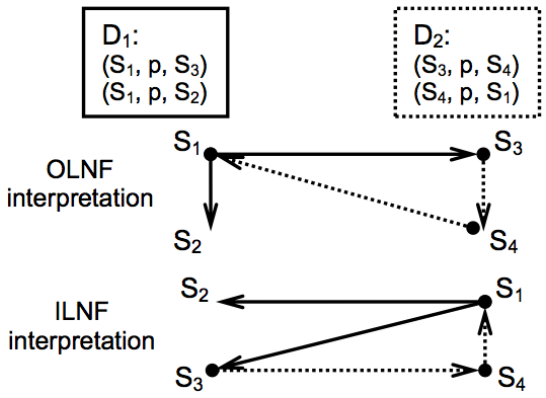

Figure 5.3 Two fragmented data sources D_1 and D_2 showing an example of both OLNF and ILNF, because both subject resources $S_1 \cup \{S_3, S_4\}$ and object resources $\{S_2, S_3\} \cup \{S_1, S_4\}$ for each fragment are disjoint. However it is not LFNF because S_1 and S_3 are shared.

1. The *link-free normal form* (LFNF) where different fragments do not share any resources.

2. The *out-link normal form* (OLNF) where different fragments do not share any *subject* resources.

3. The *in-link normal form* (ILNF) where different fragments do not share any *object* resources.

The scenario in Figure 5.1 is an example of OLNF. Note that formally an RDF store holds a set of triples (edges), and in general the resources (nodes) are not necessarily owned by any data store; thus, it is possible that a set of data sources can simultaneously conform to both OLNF and ILNF as in Figure 5.3. However in practice, the domain name of a resource often indicates its ownership; hence if a set of data sources satisfy both OLNF and ILNF we can use the domain name of the resources to determine which normal form is more appropriate to use. We further note that the three normal forms described above are not exhaustive, i.e., an RDF data set can, in general, be fragmented in ways that do not conform to any of the three normal forms considered here.

We observe that obtaining the statistics needed for learning classifiers from multiple RDF stores when the data fragmentation corresponds to LFNF reduces to combining the results of the statistical queries from the individual sources, e.g., the root and leaf projections (see Figure 5.4, top). Because we can decompose the statistical queries in this fashion, the communication

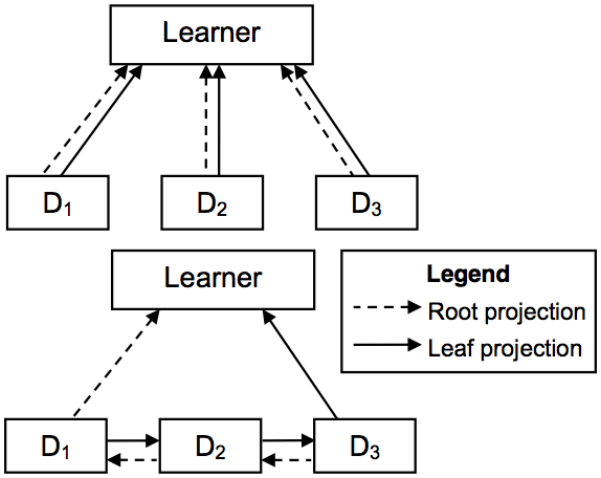

Figure 5.4 Computation of root projection and leaf projection under different data fragmentations: (i) LFNF (top); and (ii) OLNF and ILNF (bottom).

complexity of learning classifiers from multiple RDF data sources in LFNF is equivalent to that of learning classifiers from a single RDF store obtained by taking the union of the RDF triples from the respective sources.

5.3.2 Learning Classifiers under OLNF and ILNF

WLOG we focus on only OLNF. We consider a chain of interlinked RDF stores (e.g. Figure 5.1). Specifically, we address the problem of obtaining the sufficient statistics in Section 5.2.3 without having to gather the data from multiple RDF stores into a central location. First we describe how to obtain the leaf and root projections in such a setting. Consider an attribute A_k, and let J_n be the number of resources shared between D_n and D_{n+1},. We define a J_n-by-J_{n+1} matrix M_n where the value at row-r_n and column-c_n is the total number of paths that connect the shared resources indexed by r_n and c_n respectively. We set J_0 to be I (number of instances), and set J_N to be s_k (number of possible values for attribute A_k). Now, we have $V_k^* = \mathbf{1}^T M_1 \cdots M_N$ and $V_{k*} = M_1 \cdots M_N \mathbf{1}$. We note that it is more efficient to multiply the matrices from the left to right for V_k^*, and from right to left for V_{k*} (see Figure 5.4, bottom). Thus, the leaf projection $V_k^* = \left(v_{k1}^*, \ldots, v_{ks_k}^*\right)$ is computed starting at D_1 by transferring $\mathbf{1}^T M_1$ to D_2, and so on ending up with V_k^* at D_N which is then transferred to the learner. The root projection $V_{k*} = \left(v_{k*}^1, \ldots, v_{k*}^I\right)$ is computed in a similar fashion starting at D_N and working

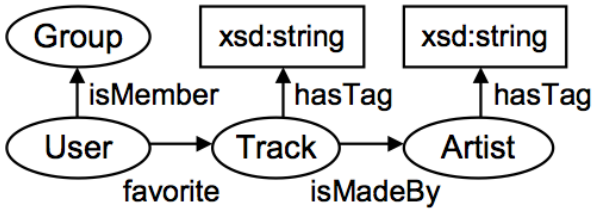

Figure 5.5 RDF Schema of Last.fm data set.

towards D_1. It is easy to see that the communication costs associated with the computation of the leaf and root projections respectively are given by $s_k + \sum_{n=1}^{N-1} J_n$ and $I + \sum_{n=1}^{N-1} J_n$.

In the case of V_k^i required by CCVD, NB(Dir), and NB(Pol), we first gather the leaf projection V_k^* and the root projection V_{k*} as described above, and use ART to reconstruct the corresponding V_k^i (see Section 5.2.4), which is used to construct the predictive model. In the case of aggregation, we use the approximated V_k^i to compute the aggregation function $agg(B_k^i)$.

We note that NB(Ber) and NB(Mul) classifiers can be learned from the leaf projections (for each class $c \in \mathcal{C}$) alone, which guarantees that the classifiers learned from an OLNF fragmented RDF data set are identical to their centralized counterparts (that are learned from the data set obtained by combining the fragments).

5.4 Experiments and Results

5.4.1 Data Sets

We used a real world data set crawled from a social music network Last.fm[3] using its API (its schema is shown in Figure 5.5). We selected two disjoint groups that contain approximately equal number of users (2098/2081), and include those tracks and artists whose number of occurrences are greater than or equal to 45 and 100, respectively. Likewise, we eliminated all the track's tags and artist's tags that occurred fewer than 350 and 120 times. All collections of tags are preprocessed by removing stop words and stemming, using Apache Lucene. The resulting data set is converted to RDF format which includes 8340 tracks attributed to one or more of the 3753 artists. From this data, we extracted two subsets: (i) Dataset-Track, which includes only the tags associated with the tracks; and (ii) Dataset-Artist, which includes only

[3] http://www.last.fm/

Table 5.1 Results for Experiment 5.4.2 that report accuracy (%) and standard deviation (in parentheses) from 10-fold cross validation. Starred (*) indicates the OLNF model is provably exact with respect to its centralized counterpart. Bolded indicates best results for each column based on paired t-test on 10-fold cross validation with alpha = 0.05.

Model	Dataset-Track		Dataset-Artist	
	Centralized	OLNF	Centralized	OLNF
Mode+NB	71.4(3.2)	53.2(1.2)	70.8(2.5)	59.8(1.4)
CCVD+LR	**81.1(2.3)**	75.7(3.5)	**81.7(1.9)**	68.9(6.3)
NB(Ber)	71.3(2.5)	71.3(2.5)*	69.5(1.8)	69.5(1.8)*
NB(Mul)	**82.0(2.4)**	**82.0(2.4)***	**81.7(2.5)**	**81.7(2.5)***
NB(Dir)	**81.4(2.7)**	78.0(3.3)	79.9(1.9)	74.1(4.2)
NB(Pol)	**82.2(2.1)**	**81.6(2.4)**	**82.2(2.3)**	**81.8(2.5)**

the tags associated with the artists. In both cases, the task is to predict the group of the user. We simulate the OLNF setting by suitably fragmenting the datasets. For example in the case of Dataset-Track we store the triples of $isMember$ and $favorite$ in D_1 and the triples of $hasTag$ in D_2 such that $Track$ resources are shared between D_1 and D_2.

5.4.2 Learning Classifiers from OLNF RDF Data Fragments

The first set of experiments was designed to compare the performance of the proposed approaches to learning classifiers from an RDF data set that is fragmented (in OLNF) across multiple RDF stores with their centralized counterparts that have access to the entire data set in a single location. We trained two aggregation models and four generative models described in Section 5.2.2: the *mode* aggregation coupled with a naive Bayes classifier (Mode+NB), the CCVD aggregation coupled with a logistic regression classifier (CCVD+LR), and the four naive Bayes generative models NB(Ber), NB(Mul), NB(Dir), and NB(Pol). Note that NB(Ber) and NB(Mul) need only leaf projections and therefore their models under OLNF is provably exact with respect to its centralized counterparts. The rest of the classifiers in the OLNF setting rely on the ART approximations of V_k^i and hence their performance is a function of the quality of the approximation. In this experiment, the termination threshold (difference between the true projection and its ART reconstruction) is set to 5% of the size of the matrix, and λ is set to 0.25.

The results in Table 5.1 show that: (i) not surprisingly, the performance of Mode+NB, CCVD+LR, NB(Dir), and NB(Pol) that rely on ART approximation of the needed statistics in the OLNF setting is always no better than that of their centralized counterparts which do not have to rely on the ART approximation and can instead use the statistics obtained directly from the entire data set; (ii) NB(Pol), despite its reliance on the ART approximation in the OLNF setting, shows performance that is competitive with its centralized counterpart although the latter has the advantage of using statistics obtained directly from the entire data set; and (iii) NB(Mul) surprisingly, is quite competitive with NB(Pol) in both centralized and OLNF settings despite using less information than NB(Pol).

5.4.3 Sensitivity of ART

The previous experiment used a fixed termination threshold for the iterative ART approximation procedure. Because the performance of the classifiers that rely on ART approximations of V_k^i is a function of the quality of the approximation which in turn depends on the the number of ART iterations, we designed an experiment to explore this dependence. In this set of experiments, we used 80% of the data for training, and 20% of the data for testing. First we measure the error of the reconstruction as estimated by (i) projection error which is the sum of absolute differences of each element between the true and reconstructed leaf and root projections; and (ii) matrix error which is the sum of absolute difference of each element between the true and reconstructed matrices. We also measure the classification accuracies of the trained models in the case of Mode+NB, CCVD+LR, NB(Dir), and NB(Pol) which make use of the ART approximation. The results summarized in Figure 5.6 show that the projection error approaches zero after a sufficiently large number of iterations; however, the matrix error remains relatively high even after 1000 iterations. In the case of classification accuracies, we note three clear trends shown in Figure 5.7: (i) Mode+NB using the ART approximation does not quite approach Mode+NB that uses statistics obtained directly from the data regardless of the number of ART iterations; (ii) the performance of NB(Dir) lags that of NB(Pol) during the first few iterations of ART but both achieve comparable performance with increasing number of ART iterations; and (iii) CCVD+LR starts off with the worst performance but shows steady improvement with

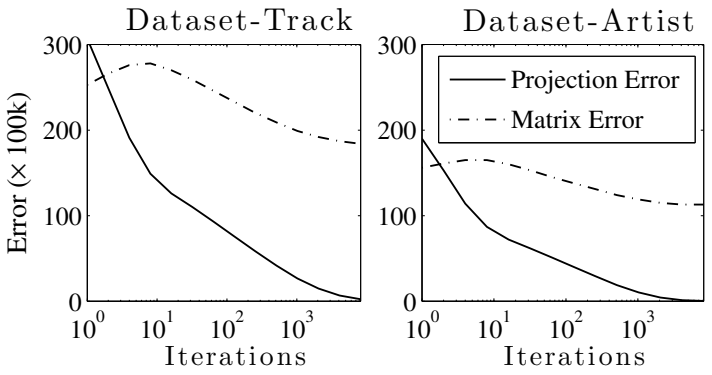

Figure 5.6 Projection and matrix errors at various stages of ART approximation.

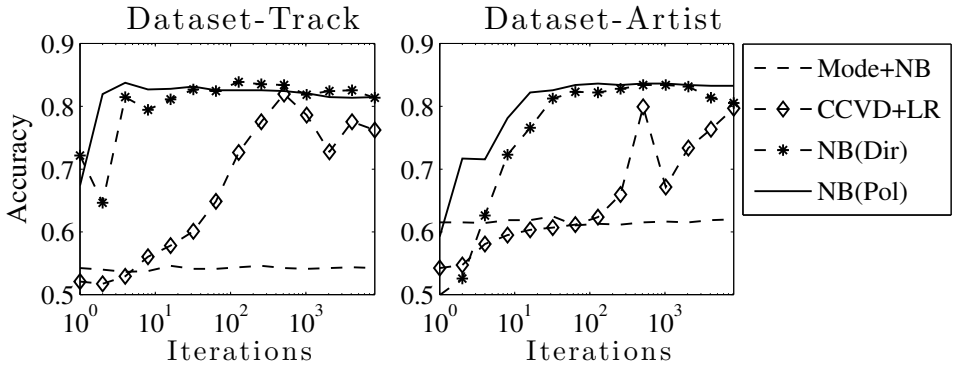

Figure 5.7 Classification performance using data reconstructed at various stages of ART approximation.

increasing number of ART iterations. However, theoretical underpinnings of these observations remain to be investigated.

5.4.4 Communication Complexity

The third experiment was designed to measure the communication cost of obtaining the projections required by the ART approximation. Since the size of query is negligible compared to the query results in our setting, we measure only the size of query results transferred, as the size of the underlying data set is varied. We used Dataset-Track and Dataset-Artist described in Section 5.4.1 considering subsets of users ranging from 400 to 4000 in steps of 400, retaining in each case only the resources (tracks, artists, tags) that are connected to the subset of users. We recorded the size of raw RDF data in TTL format, the size of leaf projection, the size of root projection, and finally the size of matrix (stored as an adjacency list).

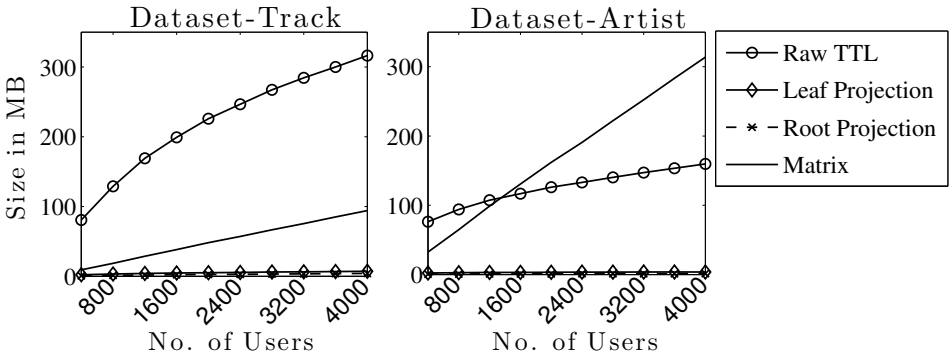

Figure 5.8 Communication complexities over the size of data sets (measured by number of Users).

The results of this experiment summarized in Figure 5.8 show that, not surprisingly, the size of raw data as well as matrices are significantly larger than the leaf and root projections, demonstrating the advantages of ART approximation in learning classifiers from large, OLNF-fragmented RDF data sets over alternative approaches that transmit the data or the matrix (as opposed to only the leaf and root projections) from the data source(s) to the learner. In the case of Dataset-Artist, we observe that the size of matrix even exceeds that of raw data, and this can be explained by the fact that a majority of artists are shared among (indirectly connected to) different users which blows up the size of the matrix, whereas in the RDF representation the artist resources and their tags only appear once in the data set.

5.5 Conclusion

5.5.1 Summary

The emergence of many interlinked, physically distributed, and autonomously maintained RDF stores such as the LOD cloud offers unprecedented opportunities for predictive modeling and knowledge discovery from such data. However existing machine learning approaches are limited in their applicability because it is neither desirable nor feasible to gather all of the data in a centralized location for analysis due to access, memory, bandwidth, computational restrictions, and sometimes privacy or confidentiality constraints. Against this background we propose to learn classifiers from multiple interlinked RDF stores via their SPARQL query interfaces. Specifically we have: (i) introduced statistical query based formulations of several

representative algorithms for learning classifiers from RDF data; (ii) introduced a distributed learning framework to learn classifiers from multiple interlinked RDF stores; (iii) identified three special cases of RDF data fragmentation and describe effective strategies for learning in each case; (iv) considered a novel application of a matrix reconstruction technique from the field of Computerized Tomography [Herman (2009)] to approximate the statistics needed by the learning algorithm from projections using count queries, thus dramatically reducing the amount of information transmitted from the remote data sources to the learner; and (v) reported results of experiments with a real-world social network data set, which demonstrate the feasibility of the proposed approach.

5.5.2 Related Work

Most of the existing work on learning predictive models from RDF data (e.g. Nickel et al. (2012); Lösch et al. (2012)) assume that the learner has direct access to data. Lin et al. (2011) proposed an approach to learning relational Bayesian classifiers [Neville et al. (2003b)] from a remote RDF store in a setting where the learner can only query the RDF data store through a restricted class of statistical queries. This paper extends the work by Lin et al. (2011) to the setting of multiple interlinked RDF stores using a larger class of predictive models including Mode+NB, CCVD+LR, NB(Dir) and NB(Pol) where, for practical reasons, we have to approximate the relevant statistics. Our approach takes advantage of SPARQL 1.1 update queries [W3C SPARQL Working Group] and federated queries [W3C SPARQL Working Group], which extends the remote access framework first introduced in Qi et al. (2013) to multiple RDF stores. As opposed to federated query processing approaches for RDF data [Buil-Aranda et al. (2013); Schwarte et al. (2011); Karnstedt et al. (2012)], which focus on the problem of answering queries formulated in a general purpose query language from multiple RDF data sources, our focus in this paper is on answering restricted classes of statistical queries needed for learning classifiers from RDF data. Restricting the classes of queries to those that useful in the learning predictive models from RDF data allows us to take advantage of optimizations such as the efficient accumulation of projections (Section 5.3.2).

The work by Lin et al. (2011) was inspired by a general learning framework proposed by

Caragea et al. (2005) for learning classifiers from distributed *tabular* data [Caragea et al. (2004, 2005)]. However, to the best of our knowledge the approaches described in this paper are among the first of their kind for learning classifiers from an RDF data set that is fragmented across multiple interlinked RDF stores.

5.5.3 Future Work

ART, the method for reconstructing an approximation of a matrix from its projections is among the simplest such technique originally developed in the field of CT. Other promising matrix reconstruction methods worth exploring in our setting include filtered backprojection [Herman (2009)] and quadratic optimization [Herman (2009)]. The kinds of RDF data fragmentation considered in this paper are relatively simple, albeit interesting special cases. It would be interesting to consider learning classifiers in a setting where an RDF data set is fragmented across multiple RDF stores that are connected through more complex linkage patterns including in particular, trees and DAGs as opposed to the linear chains considered in this study. Lastly it is of interest to consider richer classes of predictive models and the corresponding learning problems, including those that model dependency between attributes (e.g. adaptations of statistical relational learning [Getoor and Taskar (2007)]), or feature construction strategies for linked data [Rossi et al. (2012)].

CHAPTER 6. GENERAL CONCLUSIONS

6.1 Summary and Contributions

The growing adoption of Linked Data has made it possible to link and share many disparate, previously isolated, distributed, autonomously generated and managed data across virtually every domain of human endeavor, including government, life sciences, geography, social media, and commerce. This offers unprecedented opportunities for using disparate data sources in predictive modeling and decision making in a broad range of applications ranging from scientific discovery to public policy [Hert et al. (2011); Sahoo et al. (2009)]. However, there has been very limited work on effective approaches to learning predictive models from Linked Data. Existing machine learning approaches are limited in their applicability in this setting:

- Many data sets that are part of LOD are so large that it is not feasible to retrieve the entire data set from an RDF store for local analysis due to main memory, bandwidth, or computational constraints. In other settings, access to data may be limited due to the privacy or confidentiality considerations [Aggarwal and Yu (2008); Wu et al. (2010)].

  Aimed with overcoming this limitation, we introduce an approach (Chapter 2 [Lin et al. (2011)]) to learning Relational Bayesian Classifiers (RBCs, [Neville et al. (2003b)]) from a single but remote RDF data store, using statistical queries through the SPARQL endpoint of the RDF store. In Chapter 4 [Lin et al. (2013)] we generalize the representation of a linked data instance into K-tuples of *bags of feature values*, and introduced a novel type of learning problem which we call distributional instance classification. We offer an alternative motivation, precisely formulate, and present solutions for this problem.

- RDF triples in an RDF store have often associated with them, RDF Schema (RDFS, Brickley and Guha (2004)) that specify set of classes that organize RDF objects (subjects and

objects of predicates) and predicates into type hierarchies as well as domain and range restrictions on RDF predicates (i.e., the type of RDF objects that can appear as subjects or objects of a predicate respectively).

Aimed with overcoming this limitation, we introduce in Chapter 3 an algorithm for learning classifier from a remote RDF data store enriched with subclass hierarchies. Our algorithm encodes the constraints specified in a subclass hierarchy using latent variables in a directed graphical model, and adopts the Variational Bayesian EM approach to efficiently learn parameters. We show with several real world datasets that our solution achieves equal or better performance compared to other state-of-art models that incorporate subclass hierarchies, and is able to scale up to large hierarchies over few thousand nodes.

- LOD includes many loosely linked, physically distributed, autonomously maintained RDF stores where it is neither desirable nor feasible to gather all of the data in a centralized location for analysis, because of access, memory, bandwidth, and computational restrictions.

  Aimed with overcoming this limitation, in Chapter 5 [Lin and Honavar (2013)] we formulate the problem of, and provide solutions to, learning predictive models from *multiple interlinked* RDF stores. Specifically we extend Chapter 2 by incorporating the set of classifiers introduced in Chapter 4, and further extend the problem by considering a chain of multiple interlinked RDF stores. In particular we consider a novel application of a matrix reconstruction technique from the field of Computerized Tomography [Herman (2009)] to approximate the statistics needed by the learning algorithm from projections using count queries, thus dramatically reducing the amount of information transmitted from the remote data sources to the learner.

6.2 Future Work

In addition to the future work outlined at the end of each chapter, there are some potential broader directions that may be fruitful for future studies. These are summarized as follows.

- Chapter 5 applies a matrix reconstruction technique to approximate the statistics needed by the learning algorithm. However there are other approaches that may be used in the same setting, e.g. matrix sketching [Liberty (2013)] and matrix sampling [Achlioptas et al. (2013)] techniques. Instead of reconstructing the data matrices, these approaches could potentially be applied to obtain a summary of the data matrices (either a sketch or a sample) and then learn a model directly from the summary. The major challenge would be to develop a systematic and scalable algorithm to compute such summaries in various settings of multiple interlinked RDF stores. In certain cases these summaries could themselves be approximated again in order to maintain tractability due to the complex interlinking of data.

- Although Chapter 2 establishes the conditions under which the RBC models can be incrementally updated in response to addition or deletion of RDF data, there is still much work to be studied in the setting where RDF stores are subject to frequent updates. In particular, it is of interest to explore interesting cases to incrementally update models from *multiple interlinked* RDF stores.

- Chapter 2 provides a relatively straightforward solution to the setting where the relevant attributes for prediction are not known a priori, by selectively crawling the RDF data for attributes of interest in the case of a single remote RDF store. A natural extension is to extend selective crawling to the case of *multiple interlinked* RDF stores. Furthermore, by following RDF links, other RDF stores may be incrementally discovered and crawled on the fly.

APPENDIX A. DERIVATION OF DISCRIMINATIVE MODELS IN SECTION 4.3.3

Conditional Likelihood

Conditional likelihood for two-class classification problem with naive Bayes assumption

$$
\begin{aligned}
P\left(c=1 \mid x_i\right) &= \frac{P\left(x_i, c=1\right)}{P\left(x_i, c=1\right)+P\left(x_i, c=0\right)} \\
&= \frac{1}{1+\frac{P(x_i, c=0)}{P(x_i, c=1)}} \\
&= \frac{1}{1+\exp\left(\ln\left(\frac{P(x_i, c=0)}{P(x_i, c=1)}\right)\right)} \\
&= \frac{1}{1+\exp\left(\ln\left(\frac{P(x_i \mid c=0) P(c=0)}{P(x_i \mid c=1) P(c=1)}\right)\right)} \\
&= \frac{1}{1+\exp\left(\ln\frac{P(c=0)}{P(c=1)}+\ln\frac{P(x_i \mid c=0)}{P(x_i \mid c=1)}\right)} \\
&= \frac{1}{1+\exp\left(\ln\frac{P(c=0)}{P(c=1)}+\ln\frac{P\left(B_1^i, \ldots, B_K^i \mid c=0\right)}{P\left(B_1^i, \ldots, B_K^i \mid c=1\right)}\right)} \\
&= \frac{1}{1+\exp\left(\ln\frac{P(c=0)}{P(c=1)}+\ln\prod_{j=1}^{K}\frac{P\left(B_j^i \mid c=0\right)}{P\left(B_j^i \mid c=1\right)}\right)} \\
&= \frac{1}{1+\exp\left(\ln\frac{P(c=0)}{P(c=1)}+\sum_{j=1}^{K}\ln\frac{P\left(B_j^i \mid c=0\right)}{P\left(B_j^i \mid c=1\right)}\right)} \\
&= \frac{1}{1+\exp\left(\ln\frac{P(c=0)}{P(c=1)}+\sum_{j=1}^{K}\left(\ln P\left(B_j^i \mid c=0\right)-\ln P\left(B_j^i \mid c=1\right)\right)\right)}
\end{aligned}
$$

Logistic Regression

Basic logistic regression formulation with parameter $\mathbf{w}$,

$$P\left(c=1 \mid x_i; \mathbf{w}\right) = \frac{1}{1+\exp\left(\ln\frac{P(c=0)}{P(c=1)} + \sum_{j=1}^{K}\left(\ln P\left(B_j \mid c=0; \mathbf{w}\right) - \ln P\left(B_j \mid c=1; \mathbf{w}\right)\right)\right)}$$

Logistic Regression for Bernoulli

Let $P_{cjt} = \theta_{cjt}$ is a parameter for Bernoulli for class c, j^{th} attribute, and t^{th} value of j^{th} attribute. B_{jt} is presence of t.

$$\begin{aligned}
\ln P\left(B_j \mid c\right) &= \sum_{t=1}^{s_j} \ln\left(\theta_{cjt}^{B_{jt}}\left(1-\theta_{cjt}\right)^{1-B_{jt}}\right) \\
&= \sum_{t=1}^{s_j}\left(\ln\theta_{cjt}^{B_{jt}} + \ln\left(1-\theta_{cjt}\right)^{1-B_{jt}}\right) \\
&= \sum_{t=1}^{s_j}\left(B_{jt}\ln\theta_{cjt} + \left(1-B_{jt}\right)\ln\left(1-\theta_{cjt}\right)\right) \\
&= \sum_{t=1}^{s_j}\left(B_{jt}\left(\ln\theta_{cjt} - \ln\left(1-\theta_{cjt}\right)\right) + \ln\left(1-\theta_{cjt}\right)\right)
\end{aligned}$$

We parametrize w.r.t. B_{jt}. Hence,

$$\begin{aligned}
P\left(c=1 \mid x_i; \mathbf{w}\right) &= \frac{1}{1+\exp\left(\ln\frac{P(c=0)}{P(c=1)} + w_0 + \sum_{j,t} B_{jt}^{i} w_{jt}\right)} \\
&= \frac{1}{1+\exp\left(w_0 + \sum_{j,t} B_{jt}^{i} w_{jt}\right)}
\end{aligned}$$

Multinomial

$$\begin{aligned}
\ln P\left(B_j \mid c\right) &= \ln P\left(\left|B_j\right|\right) + \ln f\left(V_j\right) + \sum_{t=1}^{s_j} \ln\theta_{cjt}^{v_{jt}} \\
&= \ln P\left(\sum_{t=1}^{s_j} v_{jt}\right) + \ln f\left(V_j\right) + \sum_{t=1}^{s_j} v_{jt}\ln\theta_{cjt}
\end{aligned}$$

Incorporating two classes together.

$$P\left(c=1 \mid x_i; \mathbf{w}\right) = \frac{1}{1+\exp\left(\ln\frac{P(c=0)}{P(c=1)} + \sum_{j,t} v_{jt}^{i} w_{jt}\right)}$$

$\ln \frac{P(c=0)}{P(c=1)}$ can be viewed as fixed w_0. For a balanced data, $\ln \frac{P(c=0)}{P(c=1)} = 0$.

Dirichlet

$$\begin{aligned}\ln P\left(B_j \mid c\right) &= \ln \frac{\Gamma\left(\sum_{t=1}^{s_j} \alpha_{cjt}\right)}{\prod_{t=1}^{s_j} \left(\Gamma \alpha_{cjt}\right)} + \sum_{t=1}^{s_j} \left(\alpha_{cjt} - 1\right) \ln \bar{v}_{jt} \\ &= \ln \Gamma\left(\sum_{t=1}^{s_j} \alpha_{cjt}\right) - \sum_{t=1}^{s_j} \ln\left(\Gamma \alpha_{cjt}\right) + \sum_{t=1}^{s_j} \alpha_{cjt} \ln \bar{v}_{jt} - \sum_{t=1}^{s_j} \ln \bar{v}_{jt}\end{aligned}$$

In the form of logistic regression we write:

$$P\left(c = 1 \mid x; \alpha\right) = \left(1 + \exp\left(\ln \frac{P(c=0)}{P(c=1)} + \sum_{j=1}^{K} \left(\ln \frac{\Gamma\left(\sum_{t=1}^{s_j} \alpha_{0jt}\right)}{\Gamma\left(\sum_{t=1}^{s_j} \alpha_{1jt}\right)} - \sum_{t=1}^{s_j} \ln \frac{\left(\Gamma \alpha_{0jt}\right)}{\left(\Gamma \alpha_{1jt}\right)} + \sum_{t=1}^{s_j} \left(\alpha_{0jt} - \alpha_{1jt}\right) \ln \bar{v}_{jt}\right)\right)\right)^{-1}$$

Gradient

Let $h_j\left(\bar{V}_j; \alpha\right)$ be $\left(\ln \frac{\Gamma\left(\sum_{t=1}^{s_j} \alpha_{0jt}\right)}{\Gamma\left(\sum_{t=1}^{s_j} \alpha_{1jt}\right)} - \sum_{t=1}^{s_j} \ln \frac{\left(\Gamma \alpha_{0jt}\right)}{\left(\Gamma \alpha_{1jt}\right)} + \sum_{t=1}^{s_j} \left(\alpha_{0jt} - \alpha_{1jt}\right) \ln \bar{v}_{jt}\right)$.

Define a function h_j,

$$h_j\left(\bar{V}_j; \alpha\right) \equiv h_j^0\left(\bar{V}_j; \alpha_0\right) - h_j^1\left(\bar{V}_j; \alpha_1\right)$$

Given,

$$h_j^0\left(\bar{V}_j; \alpha_0\right) = \ln \Gamma\left(\sum_{t=1}^{s_j} \alpha_{0jt}\right) - \sum_{t=1}^{s_j} \ln\left(\Gamma \alpha_{0jt}\right) + \sum_{t=1}^{s_j} \alpha_{0jt} \ln \bar{v}_{jt}$$

Its partial derivative is

$$\frac{\partial h_j^0\left(\bar{V}_j; \alpha_0\right)}{\partial \alpha_{0ab}} = \psi_0\left(\sum_{t=1}^{s_j} \alpha_{0jt}\right) - \sum_{t=1}^{s_j} \psi_0\left(\alpha_{0jt}\right) + \ln \bar{v}_{ab}$$

Hence,

$$\frac{\partial h_j\left(\bar{V}_j; \alpha\right)}{\partial \alpha_{cab}} = (-1)^{\delta_{c,1}} \left(\psi_0\left(\sum_{t=1}^{s_j} \alpha_{cjt}\right) - \sum_{t=1}^{s_j} \psi_0\left(\alpha_{cjt}\right) + \ln \bar{v}_{ab}\right)$$

Let g be a function inside of exponential.

$$g\left(\bar{V}^i; \alpha\right) = \ln \frac{P\left(c=0\right)}{P\left(c=1\right)} + \sum_{j=1}^{K} h_j\left(\bar{V}_j^i; \alpha\right)$$

We want to maximize conditional log likelihood $\ell(\alpha)$

$$\begin{aligned}\arg\max_{\alpha} \ell(\alpha) &= \arg\max_{\alpha} \sum_{i=1}^{n} \ln P\left(c_i \mid \bar{V}^i; \alpha\right) \\ &= \arg\max_{\alpha} \sum_{i=1}^{n} c_i \ln P\left(c_i = 1 \mid \bar{V}^i; \alpha\right) + (1 - c_i) \ln P\left(c_i = 0 \mid \bar{V}^i; \alpha\right)\end{aligned}$$

We get

$$\ell(\alpha) = \sum_{i=1}^{n} c_i g\left(\bar{V}^i; \alpha\right) - \ln\left(1 + \exp g\left(\bar{V}^i; \alpha\right)\right)$$

its partial derivative is

$$\frac{\partial g\left(\bar{V}^i; \alpha\right)}{\partial \alpha_{cab}} = \sum_{j=1}^{K} \frac{\partial h_j\left(\bar{V}_j^i; \alpha\right)}{\partial \alpha_{cab}} = \frac{\partial h_a\left(\bar{V}_a^i; \alpha\right)}{\partial \alpha_{cab}}$$

Finally, partial derivative of $\ell(\alpha)$ is

$$\begin{aligned}\frac{\partial \ell(\alpha)}{\partial \alpha_{cab}} &= \sum_{i=1}^{n} \left(\frac{\partial c_i g\left(\bar{V}^i; \alpha\right)}{\partial \alpha_{cab}} - \frac{\partial \ln\left(1 + \exp g\left(\bar{V}^i; \alpha\right)\right)}{\partial \alpha_{cab}} \right) \\ &= \sum_{i=1}^{n} \left(c_i \frac{\partial g\left(\bar{V}^i; \alpha\right)}{\partial \alpha_{cab}} - \frac{\frac{\partial \exp g\left(\bar{V}^i; \alpha\right)}{\partial \alpha_{cab}}}{\left(1 + \exp g\left(\bar{V}^i; \alpha\right)\right)} \right) \\ &= \sum_{i=1}^{n} \left(c_i \frac{\partial g\left(\bar{V}^i; \alpha\right)}{\partial \alpha_{cab}} - \frac{\exp g\left(\bar{V}^i; \alpha\right) \frac{\partial g\left(\bar{V}^i; \alpha\right)}{\partial \alpha_{cab}}}{\left(1 + \exp g\left(\bar{V}^i; \alpha\right)\right)} \right) \\ &= \sum_{i=1}^{n} \left(c_i \frac{\partial h_a\left(\bar{V}_a^i; \alpha\right)}{\partial \alpha_{cab}} - \frac{\exp g\left(\bar{V}^i; \alpha\right)}{\left(1 + \exp g\left(\bar{V}^i; \alpha\right)\right)} \frac{\partial h_a\left(\bar{V}_a^i; \alpha\right)}{\partial \alpha_{cab}} \right) \\ &= \sum_{i=1}^{n} \left(c_i - P\left(c_i = 1 \mid \bar{V}^i; \alpha\right)\right) \frac{\partial h_a\left(\bar{V}_a^i; \alpha\right)}{\partial \alpha_{cab}} \\ &= \sum_{i=1}^{n} \left(c_i - P\left(c_i = 1 \mid \bar{V}^i; \alpha\right)\right) (-1)^{\delta_{c,1}} \left(\psi_0\left(\sum_{t=1}^{s_a} \alpha_{cat}\right) - \sum_{t=1}^{s_a} \psi_0\left(\alpha_{cat}\right) + \ln \bar{v}_{ab}^i \right)\end{aligned}$$

Polya

Probability is defined as

$$P(V_j \mid c) = \frac{\Gamma\left(\sum_{t=1}^{s_j} \alpha_{cjt}\right)}{\Gamma\left(\sum_{t=1}^{s_j} v_{jt} + \alpha_{cjt}\right)} \prod_{t=1}^{s_j} \frac{\Gamma\left(v_{jt} + \alpha_{cjt}\right)}{\Gamma\left(\alpha_{cjt}\right)}$$

Log probability is

$$\begin{aligned}\ln P\left(V_j \mid c\right) &= \ln \frac{\Gamma\left(\sum_{t=1}^{s_j} \alpha_{cjt}\right)}{\Gamma\left(\sum_{t=1}^{s_j} v_{jt} + \alpha_{cjt}\right)} \prod_{t=1}^{s_j} \frac{\Gamma\left(v_{jt} + \alpha_{cjt}\right)}{\Gamma\left(\alpha_{cjt}\right)} \\ &= \ln \Gamma\left(\sum_{t=1}^{s_j} \alpha_{cjt}\right) - \ln \Gamma\left(\sum_{t=1}^{s_j} v_{jt} + \alpha_{cjt}\right) + \sum_{t=1}^{s_j} \ln \Gamma\left(v_{jt} + \alpha_{cjt}\right) - \sum_{t=1}^{s_j} \ln \Gamma\left(\alpha_{cjt}\right)\end{aligned}$$

Gradient

Partial derivative of log probability w.r.t α_{cab} is

$$\psi_0\left(\sum_{t=1}^{s_a} \alpha_{cat}\right) - \psi_0\left(\sum_{t=1}^{s_a} v_{at} + \alpha_{cat}\right) + \psi_0\left(V_{ab} + \alpha_{cab}\right) - \psi_0\left(\alpha_{cab}\right)$$

For each α and c we get

$$\frac{\partial \ell\left(\alpha\right)}{\partial \alpha_{cab}} = \sum_{i=1}^{n} \left(\left(c_i - P\left(c_i = 1 \mid V^i; \alpha\right)\right)\right) \left(-1\right)^{\delta_{c,1}}$$
$$\left(\psi_0\left(\sum_{t=1}^{s_a} \alpha_{cat}\right) - \psi_0\left(\sum_{t=1}^{s_a} v_{at}^i + \alpha_{cat}\right) + \psi_0\left(v_{ab}^i + \alpha_{cab}\right) - \psi_0\left(\alpha_{cab}\right)\right)$$

BIBLIOGRAPHY

Achlioptas, D., Karnin, Z., and Liberty, E. (2013). Near-optimal distributions for data matrix sampling. In *Advances in Neural Information Processing Systems*. 73

Ackerson, L. K. and Viswanath, K. (2009). Communication inequalities, social determinants, and intermittent smoking in the 2003 health information national trends survey. *Prev Chronic Dis*, 6(2). 23

Aggarwal, C. C. and Yu, P. S. (2008). *Privacy-Preserving Data Mining: Models and Algorithms*. Springer Publishing Company, Incorporated, 1 edition. 2, 55, 71

Balasubramanyan, R. and Cohen, W. W. (2011). Block-lda: Jointly modeling entity-annotated text and entity-entity links. In *Proceedings of the Eleventh SIAM International Conference on Data Mining*, pages 450–461. 35, 52

Beal, M. J. and Ghahramani, Z. (2006). Variational bayesian learning of directed graphical models with hidden variables. *Bayesian Analysis*, 1(4):793–832. 28, 31, 33, 34

Bhagat, S., Cormode, G., and Muthukrishnan, S. (2011). Node classification in social networks. In Aggarwal, C. C., editor, *Social Network Data Analytics*, pages 115–148. Springer US. 54

Bicer, V., Tran, T., and Gossen, A. (2011). Relational kernel machines for learning from graph-structured RDF data. In *Proceedings of the 8th Extended Semantic Web Conference*. 7, 24

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. 37, 40, 51

Blei, D. and Lafferty, J. (2009). Topic models. *Text mining: classification, clustering, and applications*, 10:71. 34

Blei, D. and McAuliffe, J. (2008). Supervised topic models. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 121–128. MIT Press, Cambridge, MA. 52

Blei, D. M. (2012). Probabilistic topic models. *Commun. ACM*, 55(4):77–84. 34, 52

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022. 52

Bouchard, G. and Triggs, B. (2004). The Tradeoff Between Generative and Discriminative Classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT '04)*, pages 721–728, Prague, Czech Republic. 43

Brickley, D. and Guha, R. (Accessed 2004). RDF vocabulary description language 1.0: RDF Schema. Online. http://www.w3.org/TR/2004/REC-rdf-schema-20040210/. 2, 71

Buil-Aranda, C., Arenas, M., Corcho, O., and Polleres, A. (2013). Federating queries in SPARQL 1.1: Syntax, semantics and evaluation. *Web Semantics: Science, Services and Agents on the World Wide Web*, 18(1):1–17. 69

Caragea, C., Caragea, D., and Honavar, V. (2009). Learning link-based classifiers from ontology-extended textual data. In *Proceedings of the 2009 21st IEEE International Conference on Tools with Artificial Intelligence*, ICTAI '09, pages 354–361, Washington, DC, USA. IEEE Computer Society. 25, 34

Caragea, D., Silvescu, A., and Honavar, V. (2004). A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. *International Journal of Hybrid Intelligent Systems*, 1(1-2):80–89. 70

Caragea, D., Zhang, J., Bao, J., Pathak, J., and Honavar, V. (2005). Algorithms and software for collaborative discovery from autonomous, semantically heterogeneous, distributed information sources. In Hoffmann, A., Motoda, H., and Scheffer, T., editors, *Discovery Science*,

volume 3735 of *Lecture Notes in Computer Science*, pages 13–44. Springer Berlin / Heidelberg. 7, 70

Chang, J., Boyd-Graber, J., and Blei, D. M. (2009). Connections between the lines: augmenting social networks with text. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 169–178, New York, NY, USA. ACM. 35, 52

Cover, T. M. and Thomas, J. A. (1991). *Elements of information theory*. Wiley-Interscience, New York, NY, USA. 18

Cyganiak, R. and Jentzsch, A. (Accessed 2011). Linking open data cloud diagram. Online. http://lod-cloud.net/. 1, 19

Dietterich, T. G., Lathrop, R. H., and Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31 – 71. 37, 52

Ding, L., DiFranzo, D., Graves, A., Michaelis, J. R., Li, X., McGuinness, D. L., and Hendler, J. (2010a). Data-gov wiki: Towards linking government data. In *AAAI Spring Symposium on Linked Data Meets Artificial Intelligence*. 21, 23

Ding, L., DiFranzo, D., Graves, A., Michaelis, J. R., Li, X., McGuinness, D. L., and Hendler, J. A. (2010b). TWC data-gov corpus: incrementally generating linked government data from data. gov. In *Proceedings of the 19th international conference on World Wide Web*, pages 1383–1386. 23

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874. 44

Ferguson, T. (1973). A bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230. 42, 58

Getoor, L. and Taskar, B. (2007). *Introduction to Statistical Relational Learning*. The MIT Press. 7, 53, 70

Gordon, R., Bender, R., and Herman, G. T. (1970). Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and x-ray photography. *Journal of Theoretical Biology*, 29(3):471–481. 60

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182. 18

Hassanzadeh, O. and Consens, M. (2009). Linked movie data base. In *WWW 2009 Workshop: Linked Data on the Web*. 19

Heath, T. and Bizer, C. (2011). Linked data: Evolving the web into a global data space. *Synthesis Lectures on the Semantic Web: Theory and Technology*, 1(1):1–136. 1

Herman, G. T. (2009). *Fundamentals of Computerized Tomography: Image Reconstruction from Projections*. Springer Publishing Company, Incorporated, 2nd edition. 4, 54, 55, 60, 69, 70, 72

Hert, M., Reif, G., and Gall, H. C. (2011). A comparison of RDB-to-RDF mapping languages. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 25–32, New York, NY, USA. ACM. 1, 71

Huang, J. and Ling, C. (2005). Using auc and accuracy in evaluating learning algorithms. *Knowledge and Data Engineering, IEEE Transactions on*, 17(3):299 – 310. 48

Hung, E., Deng, Y., and Subrahmanian, V. S. (2005). RDF aggregate queries and views. In *21st International Conference on Data Engineering*, page 717–728. 16

Jaakkola, T. and Haussler, D. (1998). Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems 11*, pages 487–493. The MIT Press. 53

Jebara, T., Kondor, R. I., and Howard, A. (2004). Probability product kernels. *Journal of Machine Learning Research*, 5:819–844. 53

Karnstedt, M., Sattler, K.-U., and Hauswirth, M. (2012). Scalable distributed indexing and query processing over linked data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 10(0):3–32. 69

Kiefer, C., Bernstein, A., and Locher, A. (2008). Adding data mining support to SPARQL via statistical relational learning methods. In *Proceedings of the 5th European semantic web conference on The semantic web: research and applications*, page 478–492. 7, 24, 25, 31, 34

Kolari, P., Finin, T., and Joshi, A. (2006). Svms for the blogosphere: Blog identification and splog detection. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 92–99. 45

Korf, R. E. (1985). Depth-first iterative-deepening: an optimal admissible tree search. *Artif. Intell.*, 27:97–109. 22

Koul, N., Bui, N., and Honavar, V. (2010). Scalable, updatable predictive models for sequence data. In *BIBM*, pages 681–685. 7, 14

Koul, N., Caragea, C., Honavar, V., Bahirwani, V., and Caragea, D. (2008). Learning classifiers from large databases using statistical queries. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 923–926. 7

Koul, N. and Lin, H. T. (Accessed 2013). Indus learning framework. Google Code - http://code.google.com/p/induslearningframework/. 8, 24, 26, 34

Liberty, E. (2013). Simple and deterministic matrix sketching. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '13, pages 581–588, New York, NY, USA. ACM. 73

Lin, H. T. and Honavar, V. (2013). Learning classifiers from chains of multiple interlinked RDF data stores. In *Proceedings of the IEEE 2nd International Congress on Big Data*. 4, 72

Lin, H. T., Koul, N., and Honavar, V. (2011). Learning relational bayesian classifiers from RDF data. In *Proceedings of the 10th international conference on The semantic web*, volume 1, pages 389–404. 3, 53, 55, 56, 58, 69, 71

Lin, H. T., Lee, S., Bui, N., and Honavar, V. (2013). Learning classifiers from distributional data. In *Proceedings of the IEEE 2nd International Congress on Big Data*. 4, 71

Liu, H. and Motoda, H. (1998). *Feature Extraction, Construction and Selection: A Data Mining Perspective*. Kluwer Academic Publishers, Norwell, MA, USA. 18

Lösch, U., Bloehdorn, S., and Rettinger, A. (2012). Graph kernels for RDF data. In Simperl, E., Cimiano, P., Polleres, A., Corcho, O., and Presutti, V., editors, *The Semantic Web: Research and Applications*, volume 7295 of *Lecture Notes in Computer Science*, pages 134–148. Springer Berlin Heidelberg. 69

Madsen, R. E., Kauchak, D., and Elkan, C. (2005). Modeling word burstiness using the dirichlet distribution. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 545–552, New York, NY, USA. ACM. 42, 58

Manola, F. and Miller, E., editors (2004). *RDF Primer*. W3C Recommendation. World Wide Web Consortium. 1, 53

McCallum, A. and Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 41–48. AAAI Press. 42, 52, 58

Minka, T. P. (2005). Discriminative models, not discriminative training. Technical report, Microsoft Research, Cambridge, UK. 43

Minka, T. P. (2012). Estimating a dirichlet distribution. Technical report. 42, 58, 59

Muandet, K., Schölkopf, B., Fukumizu, K., and Dinuzzo, F. (2012). Learning from distributions via support measure machines. *CoRR*, abs/1202.6504. 53

Murphy, K. P. (2012). *Machine Learning: a Probabilistic Perspective*. MIT Press. 40

Nallapati, R. M., Ahmed, A., Xing, E. P., and Cohen, W. W. (2008). Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 542–550, New York, NY, USA. ACM. 35, 52

Nelson, D., Kreps, G., Hesse, B., Croyle, R., Willis, G., Arora, N., Rimer, B., Viswanath, K. V., Weinstein, N., and Alden, S. (2004). The health information national trends survey (hints): Development, design, and dissemination. *Journal of Health Communication: International Perspectives*, 9(5):443–460. 23

Neville, J., Jensen, D., Friedland, L., and Hay, M. (2003a). Learning relational probability trees. In *Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, page 625–630. 31

Neville, J., Jensen, D., and Gallagher, B. (2003b). Simple estimators for relational bayesian classifiers. In *Proceedings of the Third IEEE International Conference on Data Mining*, pages 609–612. 3, 11, 19, 31, 53, 55, 69, 71

Nickel, M., Tresp, V., and Kriegel, H.-P. (2012). Factorizing YAGO: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 271–280, New York, NY, USA. ACM. 69

Perlich, C. and Provost, F. (2006). Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning*, 62(1-2):65–105. 40, 41, 46, 57

Prud'ommeaux, E. and Seaborne, A. (2008). SPARQL query language for RDF. 1, 2, 10

Qi, L., Lin, H. T., and Honavar, V. (2013). Clustering remote RDF data using SPARQL update queries. In *In Proceedings of the ICDE Workshop on Graph Data Management: Techniques and Applications*. 69

Rettinger, A., Nickles, M., and Tresp, V. (2009). Statistical relational learning with formal ontologies. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 286–301, Berlin, Heidelberg. Springer-Verlag. 26, 34

Rossi, R. A., McDowell, L. K., Aha, D. W., and Neville, J. (2012). Transforming graph data for statistical relational learning. *J. Artif. Int. Res.*, 45(1):363–441. 70

Sahoo, S. S., Halb, W., Hellmann, S., Idehen, K., Jr, T. T., Auer, S., Sequeda, J., and Ezzat, A. (2009). A survey of current approaches for mapping of relational databases to RDF. Online. http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SurveyReport.pdf. Accessed 2012. 1, 71

Schwarte, A., Haase, P., Hose, K., Schenkel, R., and Schmidt, M. (2011). Fedx: optimization techniques for federated query processing on linked data. In *Proceedings of the 10th international conference on The semantic web*, volume 1, pages 601–616. 69

Tauberer, J. (Accessed 2011). The 2000 U.S. census: 1 billion RDF triples. Online. http://www.rdfabout.com/demo/census/. 21

Tirilly, P., Claveau, V., and Gros, P. (2008). Language modeling for bag-of-visual words image categorization. In *Proceedings of the 7th ACM International Conference on Image and Video Retrieval*, CIVR '08, pages 249–258, New York, NY, USA. ACM. 52

Tresp, V., Huang, Y., Bundschus, M., and Rettinger, A. (2009). Materializing and querying learned knowledge. In *Proceedings of the First ESWC Workshop on Inductive Reasoning and Machine Learning on the Semantic Web*. 7, 24

W3C SPARQL Working Group. SPARQL 1.1 overview. 56, 61, 69

Wu, X., Ying, X., Liu, K., and Chen, L. (2010). A survey of privacy-preservation of graphs and social networks. In Aggarwal, C. C., Wang, H., and Elmagarmid, A. K., editors, *Managing and Mining Graph Data*, volume 40 of *The Kluwer International Series on Advances in Database Systems*, pages 421–453. Springer US. 2, 55, 71

Zhang, J., Caragea, D., and Honavar, V. (2005). Learning ontology-aware classifiers. In Hoffmann, A., Motoda, H., and Scheffer, T., editors, *Discovery Science*, volume 3735 of *Lecture Notes in Computer Science*, pages 308–321. Springer Berlin / Heidelberg. 25, 34, 35

Zhang, J., Kang, D.-K., Silvescu, A., and Honavar, V. (2006). Learning accurate and concise naïve bayes classifiers from attribute value taxonomies and data. *Knowl. Inf. Syst.*, 9(2):157–179. 29, 31, 33, 34, 35

Zhou, Z.-H. (2004). Multi-instance learning: A survey. Technical report, Department of Computer Science and Technology, Nanjing University. 37, 52