Copyright
by
Tuyen Ngoc Huynh
2011

The Dissertation Committee for Tuyen Ngoc Huynh
certifies that this is the approved version of the following dissertation:

Improving the Accuracy and Scalability of Discriminative Learning Methods for Markov Logic Networks

Committee:

Raymond J. Mooney, Supervisor

Pedro Domingos

Joydeep Ghosh

Kristen Grauman

Pradeep Ravikumar

Improving the Accuracy and Scalability of Discriminative Learning Methods for Markov Logic Networks

by

Tuyen Ngoc Huynh, B.E., M.S.C.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2011

Dedicated to my parents.

Acknowledgments

First and foremost, I want to thank my advisor Raymond Mooney for his guidance, encouragement, and support during the past six years. I admire Ray for his tremendous passion for science and his broad knowledge about Artificial Intelligence, Machine Learning, Natural Language Processing and many more including academic genealogy. I enjoyed meeting with Ray every week. Our weekly discussions have formed the main contributions in this thesis. I deeply appreciate his great patience with my strong accent and the freedoom he gave me to pursue new ideas. I will definitely miss him and our weekly meetings.

I would like to thank my committee members—Pedro Domingos, Joydeep Ghosh, Kristen Grauman, and Pradeep Ravikumar—for their insightful comments and suggestions which have help strengthen the work in this thesis. I am especially grateful to Pedro Domingos for taking time to visit Austin for my defense. I am also indebted to Kristen Grauman for giving detailed comments on my manuscripts and her help on various things.

I want to give a special thank to Lilyana Mihalkova who collaborated with me on my first publication at UT. Lily also helped me organize the Statistical Relational Learning reading group. I would also like to thank past and present members of the Machine Learning group: Bisha Barman, Yinon

Bentor, David Chen, Dan Garrette, Sonal Gupta, Rohit Kate, Joohyun Kim, Lilyana Mihalkova, Sindhu Raghavan, Joe Reisinger, and Parag Singla for coming to my practice talks and for their feedback and comments on paper drafts and presentations.

I would also like to thank some of the members of the UTCS department staff for their help during the last six years. Katherine Utz, Stacy Miller, Gloria Ramirez, and Lydia Griffith have been incredibly resourceful and very prompt in helping me with numerous administrative issues.

I am also grateful to my Vietnamese friends in Austin for being my second family, and for giving me the love, care and joy that I much needed.

Finally, this thesis would not have been possible without the tremendous support of my family: my father Huynh Ngoc Duong, my mother Nguyen Thi Tien, my sister Huynh Ngoc Nguyet Diem, my younger brother Huynh Ngoc Tuan, and my lover Trang Lam. I could never thank them enough for always believing in me and making me feel loved. I dedicate this thesis to them.

The research in this thesis was supported by the Defense Advanced Research Projects Agency (DARPA) and managed by the Air Force Research Laboratory (AFRL) under contract FA8750-05-2-0283, a gift from Microsoft Research, and ARO MURI grant W911NF-08-1-0242. I also want to thank MOSEK ApS and IBM for the free academic licenses of Mosek and CPLEX respectively. I also thank Vietnam Education Foundation (VEF) for the financial support during my first 2 years at UT and its sponsorship.

Tuyen Ngoc Huynh

The University of Texas at Austin

May 2011

Improving the Accuracy and Scalability of Discriminative Learning Methods for Markov Logic Networks

Publication No. __________________

Tuyen Ngoc Huynh, Ph.D.
The University of Texas at Austin, 2011

Supervisor: Raymond J. Mooney

Many real-world problems involve data that both have complex structures and uncertainty. Statistical relational learning (SRL) is an emerging area of research that addresses the problem of learning from these noisy structured/relational data. Markov logic networks (MLNs), sets of weighted first-order logic formulae, are a simple but powerful SRL formalism that generalizes both first-order logic and Markov networks. MLNs have been successfully applied to a variety of real-world problems ranging from extraction knowledge from text to visual event recognition. Most of the existing learning algorithms for MLNs are in the generative setting: they try to learn a model that is equally capable of predicting the values of all variables given an arbitrary set of evidence; and they do not scale to problems with thousands of examples. However, many real-world problems in structured/relational data are discriminative—where the variables are divided into two disjoint sets input and output, and the goal is to correctly predict the values of the output variables given evidence data about the input variables. In addition, these problems usually involve data that have thousands of examples. Thus, it is important to develop new discriminative learning methods for MLNs that are more accurate and more scalable, which are the topics addressed in this thesis.

First, we present a new method that discriminatively learns both the structure and parameters for a special class of MLNs where all the clauses are non-recursive ones. Non-recursive clauses arise in many learning problems in Inductive Logic Programming. To further improve the predictive accuracy, we propose a max-margin approach to learning weights for MLNs. Then, to address the issue of scalability, we present CDA, an online max-margin weight learning algorithm for MLNs. Ater that, we present OSL, the first algorithm that performs both online structure learning *and* parameter learning. Finally, we address an issue arising in applying MLNs to many real-world problems: learning in the presence of many hard constraints. Including hard constraints during training greatly increases the computational complexity of the learning problem. Thus, we propose a simple heuristic for selecting which hard constraints to include during training.

Experimental results on several real-world problems show that the proposed methods are more accurate, more scalable (can handle problems with thousands of examples), or both more accurate and more scalable than existing learning methods for MLNs.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A lot of data in the real world are in the form of structured/relational data such as sequences, graphs, multi-relational data, etc. These structured data contain a lot of entities (or objects) and relationships among the entities. For example, biochemical data contain information about various atoms and their interactions, social network data contain information about people and relationships between them, and so on. Moreover, there are a lot of uncertainties in these data: uncertainty about the attributes of an object, the type of an object, as well as relationships between objects. Statistical relational learning (SRL) (Getoor & Taskar, 2007) which combines ideas from rich knowledge representations, such as first-order logic, with those from probabilistic graphical models is an emerging area of research that addresses the problem of learning from these noisy structured/relational data.

A variety of different SRL models have been proposed in the last two decades. Among them, Markov Logic Networks (MLNs) (Richardson & Domingos, 2006; Domingos & Lowd, 2009), sets of weighted first-order logic formulae, are an elegant but powerful formalism. It generalizes both first-order logic and Markov networks. MLNs are capable of representing all possible probability distributions over a finite number of objects (Richardson & Domingos, 2006). Moreover, MLNs also subsume other SRL representations such as Probabilistic Relational Models (Koller & Pfeffer, 1998) and Relational Markov Networks (Taskar, Abbeel, & Koller, 2002). MLNs have been successfully applied to a variety of real-world problems ranging from extraction knowledge from text (Kok & Domingos, 2008) to visual event recognition (Tran & Davis, 2008). Therefore, in this thesis, we have chosen MLNs as the model for doing research.

Currently, most of the existing learning algorithms for MLNs are in the generative setting: they try to learn a model that is equally capable of predicting the values of all variables given an arbitrary set of evidence. However, most of the learning problems in relational data are discriminative—where the variables are divided into two disjoint sets input and output, and the goal is to correctly predict the values of the output variables given evidence data about the input ones. For example, in many problems in biochemistry, the goal is to learn a model that discriminates the active chemical compounds from the inactive ones based on their molecular structures. This task is called the structure activity relationship prediction, and it is an important task in drug design and discovery (King, Sternberg, & Srinivasan, 1995). Another example is *structured prediction* problems (Bakir, Hoffman, Schölkopf, Smola, Taskar, & Vishwanathan, 2007) where the output variables are interdependent. For example, in field segmentation (Grenager, Klein, & Manning, 2005), one is given a text document represented as a sequence of tokens and the goal is to segment the document into fields (i.e. to label each token in the document with a field label). Note that, there are dependencies between tokens' labels such as consecutive tokens usually have the same field label. It is, therefore, an important research problem to develop discriminative learning methods for MLNs that have high predictive accuracies on these discriminative tasks.

On the other hand, all existing methods for learning the structure (i.e. logical clauses) of an MLN (Kok & Domingos, 2005; Mihalkova & Mooney, 2007; Biba, Ferilli, & Esposito, 2008; Kok & Domingos, 2009, 2010) are batch algorithms that are effectively designed for training data with relatively few *mega-examples* (Mihalkova, Huynh, & Mooney, 2007). A mega-example is a large set of connected facts, and mega-examples are disconnected and independent from each other. For instance, in WebKB (Slattery & Craven, 1998), there are four mega-examples, each of which contains data about a particular university's computer-science department's web pages of professors, students, courses, research projects and the hyperlinks between them. Previous work has found that there are a lot of repeated patterns in each mega-example and exploited this characteristic to develop efficient structure learning methods for MLNs on those problems (Kok & Domingos, 2009, 2010). However, there are many real-world problems with a different character — involving data with thousands of smaller structured examples. For example, a standard dataset for semantic role labeling consists of 90,750 training examples where each example is a verb and all of its semantic arguments in a sentence (Carreras & Màrquez, 2005). In addition, each example does not contain a lot of repeated patterns, but the patterns are repeated across examples. On the other hand, most existing methods for learning the parameters (i.e. clauses' weights) of an MLN employs batch training where the learner must repeatedly run inference over all training examples in each iteration, which becomes computationally expensive on datasets with thousands of training examples. Thus, it is necessary to develop new discriminative learning methods for MLNs that can handle data with a large number of examples.

1.1 Thesis Contributions

This thesis addresses two important issues in discriminative learning for MLNs: accuracy and scalability. We presents new discriminative learning methods that are more accurate, more scalable (can handle problems with thousands of examples), or both.

First, we describe a new method that discriminatively learns both the structure and parameters for a special class of MLNs where all the clauses are non-recursive ones which arise in many benchmark problems in Inductive Logic Programming (ILP). Most existing learning methods for non-recursive clauses in ILP are purely logical approaches, which cannot handle uncertainty. So, the idea is to use those ILP methods to construct a large number of potentially useful clauses, and then use l_1-regularized parameter learning methods to properly weight them, preferring to assign zero weights to clauses that do not contribute significantly to overall predictive accuracy, thereby eliminating them. The proposed approach outperforms existing ones in term of predictive accuracy and achieves state-of-the-art results on a benchmark problem in drug design.

Second, existing discriminative methods for learning the weights of an MLN attempt to maximize the conditional log likelihood, which is suitable when the goal is to predict accurate probabilities. However, in many applications, the actual goal is to maximize an alternative performance metric such as classification accuracy or F-measure. Max-margin methods are a competing approach to discriminative training that are well-founded in computational learning theory and have demonstrated empirical success in many applications (Cristianini & Shawe-Taylor, 2000). They also have the advantage that they can be adapted to maximize a variety of performance metrics in addition to classification accuracy (Joachims, 2005). Thus, we present a max-margin approach to learning weights for an MLN. We show how to formulate the weight learning problem for MLNs as a max-margin optimization problem. In order to solve the optimization problem, we develop a new approximate inference algorithm for MLNs based on Linear Programming relaxation. Experimental results on several problems show that our max-margin weight learner generally has better and more stable predictive accuracy than the previously best discriminative MLN weight learner.

However, like other existing weight learners for MLNs, the above max-margin weight learner does not scale to problems with thousands of training examples. To address this issue, we develop CDA, an online max-margin weight learning algorithm for structured prediction, and apply it to learn weights for an MLN. The algorithm is derived from the primal-dual framework (Kakade & Shalev-Shwartz, 2009), a general framework for deriving online algorithms that have low regret. Since CDA processes one example at a time, it can handle problems with thousands of training examples where existing batch learning methods for MLNs cannot. On the other hand, CDA generally achieves better accuracy than existing online methods for structured prediction.

The above CDA online algorithm only updates the parameters of an input MLN and assumes the structure of the input MLN is complete or perfect. Nevertheless, it is usually impossible or infeasible to specify a complete or perfect model's structure at the beginning. So it would be useful to have an algorithm that enhances the model's initial structure along with updating the model's parameters. Therefore, we present OSL, the first algorithm that performs both online structure and parameter learning. At each step, based on the model's wrong predictions, OSL finds new clauses that fix these errors, then uses an adaptive subgradient method with l_1-regularization to update weights for both old and new clauses. Experimental results on two real-world datasets show that OSL outperforms systems that only do online parameter learning. In addition, OSL also performs well when starting from scratch (i.e. no input structure).

Finally, we address an issue arising in applying MLNs to many real-world problems: learning in the presence of many hard constraints. Including hard constraints during training greatly increases the computational complexity of the learning problem. Thus, we propose a simple heuristic for selecting which hard constraints to include during training. Experimental results on the task of bibliographic citation segmentation show that the proposed approach achieves the best predictive accuracy while still allowing for efficient training.

1.2 Thesis Outline

The remainder of the thesis is organized as follows.

- Chapter 2 reviews our terminology and notation, and presents background on MLNs, max-margin structured prediction, the primal-dual framework, and some standard evaluation metrics.
- Chapter 3 describes our discriminative structure and parameter learning algorithm for MLNs with non-recursive clauses.
- Chapter 4 presents the max-margin approach to learning weights for MLNs. Chapter 5 discusses our online max-margin weight learning algorithm for MLNs.
- Chapter 6 describes OSL, an online algorithm that updates both the structure and parameters of an MLN.
- Chapter 7 presents our work on learning with hard constraints.
- Chapter 8 discusses future work and chapter 9 concludes the thesis.

We note that the material presented in Chapter 3 has appeared in our previous publication (Huynh & Mooney, 2008), the material in Chapter 4 has appeared in (Huynh & Mooney, 2009) and the material in Chapter 5 has appeared in (Huynh & Mooney, 2011).

Chapter 2

Background

2.1 Terminology and Notation

There are four types of symbols in first-order logic: constants, variables, predicates, and functions (Genesereth & Nilsson, 1987). Here, we assume that the domains contain no functions. Constants are objects in the domain and can have types. Variables range over objects in the domain. Predicates represent relations in the domain. Each predicate has a number of arguments. Each argument can have a type that specifies the type of constant that can be used to ground it. We denote constants by strings starting with upper-case letters, and variables by strings starting with lower-case letters. A term is a constant or a variable. An atom is a predicate applied to terms. A ground atom is an atom all of whose arguments are constants. A positive literal is an atom, and a negative literal is a negated atom. A ground literal is a literal containing only constants. A possible world is an assignment of truth values to all ground atoms in a domain. A formula consists of literals connected by logical connectives (i.e. $\vee$ and $\wedge$). A formula in clausal form, also called a clause, is a disjunction of literals. A ground clause is a clause containing only ground literals. A clause with at most one positive literal is called a Horn clause. A Horn clause with exactly one positive literal is a definite clause.

For mathematical terms, we use lower case letters (e.g. η, λ) to denote scalars, bold face letters (e.g. $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{w}$) to denote vectors, and upper case letters (e.g. $\mathcal{W}$, $\mathcal{X}$) to denote sets. The inner product between vectors $\boldsymbol{w}$ and $\boldsymbol{x}$ is denoted by $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$. The $[a]_+$ notation denotes a truncated function at 0, i.e. $[a]_+ = max(a, 0)$

2.2 Inductive Logic Programming and Aleph

Traditional Inductive Logic Programming (ILP) systems discriminatively learn logical Horn-clause rules (logic programs) for inferring a given target predicate given information provided by a set of background predicates. These purely logical definitions are induced from Horn-clause background knowledge and a set of positive and negative tuples of the target predicate. For more information about ILP, please see (Dzeroski, 2007).

Aleph is a popular and effective ILP system primarily based on Progol (Muggleton, 1995). The basic Aleph algorithm consists of four steps. First, it selects a positive example to serve as the “seed” example. Then, it constructs the most specific clause, the “bottom clause”, that entails that selected example. The bottom clause is formed by conjoining all known facts about the seed example. Next, Aleph finds generalizations of this bottom clause by performing a general to specific search. These generalized clauses are scored using a chosen evaluation metric, and the clause with the best score is added to the final theory. This process is repeated until it finds a set of clauses that covers all the positive examples. Aleph allows users to customize each of these steps, and thereby supports a variety of specific algorithms.

2.3 MLNs and Alchemy

An MLN consists of a set of weighted first-order logic formulae. It provides a way of softening first-order logic by making situations in which not all formulae are satisfied less likely but not impossible (Richardson & Domingos, 2006; Domingos & Lowd, 2009). More formally, let $\mathcal{X}$ be the set of all ground atoms, $\mathcal{C}$ be the set of all clauses in the MLN, w_i be the weight associated with clause $c_i \in \mathcal{C}$, $\mathcal{G}_{c_i}$ be the set of all possible groundings of clause c_i. Then the probability of a possible world $\mathbf{x}$ is defined as (Richardson & Domingos, 2006):

$$\begin{aligned} P(\mathcal{X} = \mathbf{x}) &= \frac{1}{\mathcal{Z}} \exp\left(\sum_{c_i \in \mathcal{C}} w_i \sum_{g \in \mathcal{G}_{c_i}} g(\mathbf{x})\right) \\ &= \frac{1}{\mathcal{Z}} \exp\left(\sum_{c_i \in \mathcal{C}} w_i n_i(\mathbf{x})\right) \end{aligned}$$

where $g(\mathbf{x})$ is 1 if g is satisfied and 0 otherwise, $n_i(\mathbf{x}) = \sum\limits_{g \in \mathcal{G}_{c_i}} g(\mathbf{x})$ is the number of true groundings of c_i in the possible world $\mathbf{x}$, and $\mathcal{Z} = \sum\limits_{\mathbf{x} \in \mathcal{X}} \exp\left(\sum_{c_i \in \mathcal{C}} w_i n_i(\mathbf{x})\right)$ is the normalization constant. In many applications, we know a priori which predicates are evidence predicates and which predicates are query ones, and the goal is to correctly predict query atoms given evidence atoms. If we partition the ground atoms in the domain into a set of evidence atoms $\mathcal{X}$ and a set of query atoms $\mathcal{Y}$, the conditional probability of $\mathbf{y}$ given $\mathbf{x}$ is:

$$P(\mathcal{Y} = \mathbf{y} | \mathcal{X} = \mathbf{x}) = \frac{1}{\mathcal{Z}_{\mathbf{x}}} \exp\left(\sum_{c_i \in \mathcal{C}} w_i n_i(\mathbf{x}, \mathbf{y})\right) \qquad (2.1)$$

where $n_i(\mathbf{x}, \mathbf{y})$ is the number of true groundings of c_i in the possible world $(\mathbf{x},\mathbf{y})$ and $\mathcal{Z}_{\mathbf{x}} = \sum\limits_{\mathbf{y} \in \mathcal{Y}} \exp\left(\sum_{c_i \in \mathcal{C}} w_i n_i(\mathbf{x}, \mathbf{y})\right)$ is the normalization constant.

There are two main inference tasks in MLNs. The first one is to infer the Most Probable Explanation (MPE) or the most probable truth values for a set of unknown literals $\mathbf{y}$ given a set of known literals $\mathbf{x}$, provided as evidence (also called MAP inference in some other work). This task is formally defined as follows:

$$\begin{aligned}\arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) &= \arg\max_{\mathbf{y}} \frac{1}{\mathcal{Z}_x} \exp\left(\sum_{c_i \in \mathcal{C}} w_i n_i(\mathbf{x}, \mathbf{y})\right) \\ &= \arg\max_{\mathbf{y}} \sum_{c_i \in \mathcal{C}} w_i n_i(\mathbf{x}, \mathbf{y})\end{aligned}$$

MPE inference in MLNs is therefore equivalent to finding the truth assignment that maximizes the sum of the weights of satisfied clauses, a Weighted MAX-SAT problem. This is an NP-hard problem for which a number of approximate solvers exist, of which the most commonly used is MaxWalkSAT (Kautz, Selman, & Jiang, 1997). Recently, Riedel (2008) proposed a more efficient method to solve the MPE inference problem called Cutting Plane Inference (CPI), which does not require grounding the whole MLN. The CPI is a meta inference algorithm that incrementally constructs some parts of a large and complex Markov network and then uses some MPE inference algorithm to find the MPE solution on the constructed network. The main idea is that we don't need to ground the whole Markov network to find the MPE solution since there is a lot of redundant information in the whole network. However, the CPI method only works well when the separation step returns a small set of constraints. In the worst case, it also constructs the whole ground MLN.

The second inference task in MLNs is marginal inference whose goal is to compute the marginal probabilities of some unknown query literals $\mathbf{y}$. Computing these probabilities is also intractable, but there are good approximation algorithms such as MC-SAT (Poon & Domingos, 2006) and lifted belief propagation (Singla & Domingos, 2008).

Learning an MLN consists of two tasks: structure learning and weight learning. The weight learner can learn weights for clauses written by a human expert or automatically induced by a structure learner. There are two approaches to weight learning in MLNs: generative and discriminative. In discriminative learning, we know a priori which predicates will be used to supply evidence and which ones will be queried, and the goal is to correctly predict the latter given the former. Several discriminative weight learning methods have been proposed, all of which try to find weights that maximize the Conditional Log Likelihood (CLL) (equivalently, minimize the negative CLL). In MLNs, the derivative of the negative CLL with respect to a weight w_i is the difference of the expected number of true groundings $E_w[n_i]$ of the corresponding clause f_i and the actual number according to the data n_i. However, computing the expected count $E_w[n_i]$ is intractable. The first discriminative weight learner (Singla & Domingos, 2005) uses the structured perceptron algorithm (Collins, 2002) where it approximates the intractable expected counts by the counts in the MPE state computed by the MaxWalkSAT. Later, Lowd and Domingos

(2007) presented a number of first-order and second-order methods for optimizing the CLL. These methods use samples from MC-SAT to approximate the expected counts used to compute the gradient and Hessian of the CLL. Among them, the best performing is *preconditioner scaled conjugate gradient* (PSCG) (Lowd & Domingos, 2007). This method uses the inverse diagonal Hessian as the preconditioner.

Regarding structure learning, there are currently two main approaches for learning clauses for MLNs. The first one is a top-down approach (Kok & Domingos, 2005; Biba et al., 2008). These algorithms can start from an empty network or from an existing knowledge base. So they can be used for learning a new MLN or revising an existing MLN. The algorithms usually start from the set of unit clauses, and iteratively add new clauses to the model. In each step, they try to find the best clause to add to the current MLN by adding, deleting, or flipping the sign of a literal (Kok & Domingos, 2005) or performing a stochastic local search (Biba et al., 2008). The weight of each candidate clause is set to optimize the *weighted pseudo log-likelihood* (WPLL) (Kok & Domingos, 2005) through an optimization procedure. Then each candidate structure is scored by the WPLL (Kok & Domingos, 2005) or by the CLL (Biba et al., 2008), and the best candidate clause is add to the learnt MLN. The other approach is the bottom-up one (Mihalkova & Mooney, 2007; Kok & Domingos, 2009, 2010). Mihalkova and Mooney (2007) proposed the first bottom-up structure learner for MLNs called Busl. It first constructs Markov network templates from the data and then generates candidate clauses from these network templates. All candidate clauses are also evaluated using WPLL, and added to the final MLN in a greedy manner. Later, Kok and Domingos (2009) proposed a new bottom-up structure learner for MLNs called LHL. The main idea of this algorithm is based on the observation that a relational database can be viewed as a hypergraph with constants as nodes and relations as hyperedges. Then a clause can be constructed from a path in the hypergraph. However, a hypergraph usually contains an exponential number of paths. So to make it tractable, the algorithm first lifts the hypergraph by jointly clustering all the constants in the relational database to form higher-level concepts, then finds paths in the lifted hypergraph. Recently, Kok and Domingos (2010) proposed LSM, another bottom-up MLN structure learner that can learn long clauses (more than 5 literals). The key insight of LSM is that relational data usually contain repeated patterns of densely connected objects called structural motifs. By limiting the search to each unique motif, LSM is able to find good clauses in an efficient manner.

ALCHEMY (Kok, Singla, Richardson, & Domingos, 2005) is an open source software package for MLNs. It includes implementations for all of the major existing algorithms for structure learning, generative weight learning, discriminative weight learning, and inference. Our proposed algorithms are implemented using ALCHEMY.

2.4 Max-margin structured prediction

In this section, we briefly review the max-margin structured prediction problem and an algorithmic schema for solving it efficiently. For more detail, see Tsochantaridis, Joachims, Hofmann, and Altun (2005), Joachims, Finley, and Yu (2009). In structured prediction, we want to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ is the space of inputs and $\mathcal{Y}$ is the space of multivariate and structured outputs, from a set of training examples S:

$$S = ((\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)) \in (\mathcal{X} \times \mathcal{Y})^n$$

The goal is to find a function h that has low prediction error. This can be accomplished by learning a discriminant function $f : \mathcal{X} \times \mathcal{Y} \rightarrow R$, then maximizing f over all $y \in \mathcal{Y}$ for a given input x to get the prediction:

$$h_{\mathbf{w}}(x) = \arg\max_{\mathbf{y} \in \mathcal{Y}} f_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$$

The discriminant function $f_{\mathbf{w}}(\mathbf{x}, \mathbf{y})$ takes the form of a linear function:

$$f_{\mathbf{w}}(\mathbf{x}, \mathbf{y}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$$

where $\mathbf{w} \in R^n$ is a parameter vector and $\boldsymbol{\phi}$ is a feature vector relating an input $\mathbf{x}$ and output $\mathbf{y}$. The features need to be designed for a given problem so that they capture the dependency structure of $\mathbf{y}$ and $\mathbf{x}$ and the relations among the outputs $\mathbf{y}$. Then, the goal is to find a weight vector $\mathbf{w}$ that maximizes the margin which is the difference between the model's score for the correct label and the model's score for the closest incorrect one:

$$\gamma(\mathbf{x}_i, \mathbf{y}_i; w) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) - \max_{\mathbf{y}'_i \in \mathcal{Y} \backslash \mathbf{y}_i} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}'_i)$$

The max-margin problem above can be formulated as an optimization problem called structural SVM (Tsochantaridis, Joachims, Hofmann, & Altun, 2004; Tsochantaridis et al., 2005) as follows:

Optimization Problem 1 (OP1): Structural SVM

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$
$$s.t.\ \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \mathbf{w}^T[\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{y}_i) - \boldsymbol{\phi}(\mathbf{x}_i,\mathbf{y})] \geq 1 - \xi_i$$

The slack variables are used to allow some errors in the training data, and the scalar C $\geq$ 0 is a hyper-parameter that controls the trade-off between minimizing the training error and maximizing the margin. This formulation implicitly imposes a zero-one loss on each constraint which is inappropriate for most kinds of structured output since it treats a prediction that is very close to the correct one as the same as a prediction that is completely different from the right one. To take into account this problem, Taskar, Guestrin, and Koller (2004) proposed to re-scale the margin by the Hamming loss of the wrong label. This margin-rescaling approach also works for other loss functions as well (Tsochantaridis et al., 2005). The resulting optimization problem is as follows:

Optimization Problem 2 (OP2): Structural SVM with Margin-Rescaling

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i$$
$$s.t.\ \forall i, \forall \mathbf{y} \in \mathcal{Y} : \mathbf{w}^T[\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{y}_i) - \boldsymbol{\phi}(\mathbf{x}_i,\mathbf{y})] \geq \Delta(\mathbf{y}_i,\mathbf{y}) - \xi_i$$

Note that, the OP1 is the OP2 with zero-one label loss. Recently, Joachims et al. (2009) proposed a reformulation of the above optimization, called "1-slack" structural SVMs which combines all training examples into one big training example and has only slack variable for the new mega example:

Optimization Problem 3 (OP3): 1-Slack Structural SVM with Margin-Rescaling

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$$

$$s.t.\ \forall(\bar{\mathbf{y}}_1,...,\bar{\mathbf{y}}_n) \in \mathcal{Y}^n : \frac{1}{n}\mathbf{w}^T\sum_{i=1}^{n}[\boldsymbol{\phi}(\mathbf{x}_i,\mathbf{y}_i) - \boldsymbol{\phi}(\mathbf{x}_i,\bar{\mathbf{y}}_i)] \geq \frac{1}{n}\sum_{i=1}^{n}\Delta(\mathbf{y}_i,\bar{\mathbf{y}}_i) - \xi$$

The 1-slack reformulation leads to a faster and more scalable training algorithm whose running time is provably linear in the number of training examples (Joachims et al., 2009).

In each iteration, the algorithm 2.1 solves a Quadratic Programming (QP) problem (line 4) to find the optimal weights corresponding to the current set of constraints $\mathcal{W}$ and a separation oracle (line 6), also called a loss-augmented inference problem (Taskar, Chatalbashev, Koller, & Guestrin, 2005), to find the most violated constraint to add to $\mathcal{W}$. The QP problem in line 4 can be solved by any general QP solver. In contrast, for each representation (such as Markov networks or weighted context free grammars) a specific algorithm is needed for solving the loss-augmented inference problem.

To enforce a sparse solution on the learned weights, we can replace the square 2-norm, $\mathbf{w}^T\mathbf{w}$, on these formulations by the 1-norm, $||\mathbf{w}||_1 = \sum_{i=1}^{n}|w_i|$,

Algorithm 2.1 Cutting-plane method for solving the "1-slack structural SVMs" (Joachims et al., 2009)

```
1: Input: $S = ((\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_n, \mathbf{y}_n)), C, \varepsilon$
2: $\mathcal{W} \leftarrow \emptyset$
3: repeat
4:
       $$(\mathbf{w}, \xi) \leftarrow \min_{\mathbf{w}, \xi \geq 0} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$$
       $$s.t. \quad \forall(\bar{\mathbf{y}_1}, ..., \bar{\mathbf{y}_n}) \in \mathcal{W} : \frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n}[\boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\phi}(\mathbf{x}_i, \bar{\mathbf{y}}_i)] \geq \frac{1}{n}\sum_{i=1}^{n}\Delta(\mathbf{y}_i, \bar{\mathbf{y}}_i) - \xi$$
5:     for $i = 1$ to $n$ do
6:         $\hat{\mathbf{y}}_i \leftarrow \arg\max_{\hat{\mathbf{y}} \in \mathcal{Y}}\{\Delta(\mathbf{y}_i, \hat{\mathbf{y}}) + \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_i, \hat{\mathbf{y}})\}$
7:     end for
8:     $\mathcal{W} \leftarrow \mathcal{W} \cup \{(\hat{\mathbf{y}_1}, ..., \hat{\mathbf{y}_n})\}$
9: until $\frac{1}{n}\sum_{i=1}^{n}\Delta(\mathbf{y}_i, \hat{\mathbf{y}}_i) - \frac{1}{n}\mathbf{w}^T\sum_{i=1}^{n}[\boldsymbol{\phi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\phi}(\mathbf{x}_i, \hat{\mathbf{y}}_i)] \leq \xi + \varepsilon$
10: return $(\mathbf{w}, \xi)$
```

like previous work on 1-norm SVMs (Bradley & Mangasarian, 1998; Zhu, Rosset, Hastie, & Tibshirani, 2003) for binary classification. Using the substitution $w_i = w_i^+ - w_i^-$ and $|w_i| = w_i^+ + w_i^-$ with $w_i^+, w_i^- \geq 0$ (Fung & Mangasarian, 2004), we can cast the 1-norm minimization problem as a Linear Programming (LP) problem and use the algorithm 2.1 to solve the LP problem by replacing the QP problem in line 4 by the transformed LP problem. A special case of the 1-norm structural SVM for the case of Markov Networks is presented in Zhu and Xing (2009).

In summary, to apply structural SVMs to a new problem, one needs to choose a representation for model, design a corresponding feature vector function $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$, select a label loss function $\Delta(\mathbf{y}, \bar{\mathbf{y}})$, and design algorithms to solve the two argmax problems:

Prediction: $\arg\max_{\mathbf{y}\in\mathcal{Y}} \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$

Separation Oracle: $\arg\max_{\bar{\mathbf{y}}\in\mathcal{Y}} \{\Delta(\mathbf{y}, \bar{\mathbf{y}}) + \mathbf{w}^T \boldsymbol{\phi}(x, \bar{y})\}$

2.5 The Primal-Dual Algorithmic Framework for Online Learning

In this section, we briefly review the primal-dual framework for strongly convex loss functions (Kakade & Shalev-Shwartz, 2009) which is the latest framework for deriving online algorithms that have low regret, the difference between the cumulative loss of the online algorithm and the cumulative loss of the optimal offline solution. Considering the following primal optimization problem:

$$\inf_{\mathbf{w}\in\mathcal{W}} P_{t+1}(w) = \inf_{\mathbf{w}\in\mathcal{W}} \left((\sigma t) f(\mathbf{w}) + \sum_{i=1}^{t} g_i(\mathbf{w}) \right) \tag{2.1}$$

where $f : W \to R_+$ is a function that measures the complexity of the weight vectors in W, $g_i : W \to R$ is a loss function, and σ is non-negative scalar. For example, if $W = R^d$, $f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2$,
and $g_i(\mathbf{w}) = \max_{y\in\mathcal{Y}} [\Delta(\mathbf{y}_t, \mathbf{y}) - \langle \mathbf{w}, (\boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y})\rangle]_+$ then the above optimization problem is the max-margin structured prediction problem described in previous section. We can rewrite the optimization problem in Eq. 2.1 as follows:

$$\inf_{\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_t} \left((\sigma t) f(\mathbf{w}_0) + \sum_{i=1}^{t} g_i(\mathbf{w}_i) \right)$$
$$\text{s.t.} \quad \mathbf{w}_0 \in \mathcal{W}, \quad \forall i \in 1 \ldots t, \mathbf{w}_i = \mathbf{w}_0$$

where we introduce t new vectors $\mathbf{w}_1, ..., \mathbf{w}_t$ and constrain them to all be equal to $\mathbf{w}_0$. The dual of this problem is:

$$\begin{aligned}
&\sup_{\boldsymbol{\lambda}_1,...,\boldsymbol{\lambda}_t} D_{t+1}(\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_t) \\
&= \sup_{\boldsymbol{\lambda}_1,...,\boldsymbol{\lambda}_t} \left[-(\sigma t) f^* \left(-\frac{1}{(\sigma t)} \sum_{i=1}^{t} \boldsymbol{\lambda}_i \right) - \sum_{i=1}^{t} g_i^*(\boldsymbol{\lambda}_i) \right]
\end{aligned}$$

where each $\boldsymbol{\lambda}_t$ is a vector of Lagrange multipliers for the equality constraint $\mathbf{w}_t = \mathbf{w}_0$, and $f^*, g_1^*, ..., g_t^*$ are the Fenchel conjugate functions of $f, g_1, ..., g_t$. A Fenchel conjugate function of a function $f : W \to R$ is defined as $f^*(\boldsymbol{\theta}) = sup_{\mathbf{w} \in W}(\langle \mathbf{w}, \boldsymbol{\theta} \rangle - f(\mathbf{w}))$. See (Kakade & Shalev-Shwartz, 2009) for details on the steps to derive the dual problem.

From the weak duality theorem (Boyd & Vandenberghe, 2004), we know that the dual objective is upper bounded by the optimal value of the primal problem. Thus, if an online algorithm can incrementally ascend the dual objective function in each step, then its performance is close to the performance of the best fixed weight vector that minimizes the primal objective function (the best offline learner), since by increasing the dual objective, the algorithm moves closer to the optimal primal value.

Based on this observation, Kakade et. al. (Kakade & Shalev-Shwartz, 2009) proposed the general online incremental dual ascent algorithm (Algorithm 2.2), where $\partial g_t(\mathbf{w}_t) = \{\boldsymbol{\lambda} : \forall \mathbf{w} \in W, g_t(\mathbf{w}) - g_t(\mathbf{w}_t) \geq \langle \boldsymbol{\lambda}, (\mathbf{w} - \mathbf{w}_t) \rangle\}$ is the set of subgradients of g_t at w_t. The condition $\exists \boldsymbol{\lambda}' \in \partial g_t$ s.t. $D_{t+1}(\boldsymbol{\lambda}_1^{t+1}, ..., \boldsymbol{\lambda}_t^{t+1}) \geq D_{t+1}(\boldsymbol{\lambda}_1^{t}, ..., \boldsymbol{\lambda}_{t-1}^{t}, \boldsymbol{\lambda}')$ ensures the dual objective is increased in each step. The

Algorithm 2.2 A general incremental dual ascent algorithm for σ-strongly convex loss function (Kakade & Shalev-Shwartz, 2009)

```
**Input:** A strongly convex function $f$, a positive scalar $\sigma$
**for** $t = 1$ **to** $T$ **do**
    Set: $\mathbf{w}_t = \nabla f^* \left( -\frac{1}{\sigma t} \sum_{i=1}^{t-1} \boldsymbol{\lambda}_i^t \right)$
    Receive: $l_t(\mathbf{w}_t) = \sigma f(\mathbf{w}_t) + g_t(\mathbf{w}_t)$
    Choose $(\boldsymbol{\lambda}_1^{t+1}, ..., \boldsymbol{\lambda}_t^{t+1})$ that satisfy the condition:
    $\exists \boldsymbol{\lambda}' \in \partial g_t(\mathbf{w}_t)$ s.t. $D_{t+1}(\boldsymbol{\lambda}_1^{t+1}, ..., \boldsymbol{\lambda}_t^{t+1}) \geq D_{t+1}(\boldsymbol{\lambda}_1^t, ..., \boldsymbol{\lambda}_{t-1}^t, \boldsymbol{\lambda}')$
**end for**
```

regret of any algorithm derived from Algorithm 2.2 is $O(\log T)$ (Kakade & Shalev-Shwartz, 2009), where T is the number of examples seen so far.

A simple update rule that satisfies the condition in Algorithm 2.2 is to find a subgradient $\boldsymbol{\lambda}' \in \partial g_t(\mathbf{w}_t)$ and set $\boldsymbol{\lambda}_t^{t+1} = \boldsymbol{\lambda}'$ and keep all other $\boldsymbol{\lambda}_i$'s unchanged (i.e. $\boldsymbol{\lambda}_i^{t+1} = \boldsymbol{\lambda}_i^t, \ \forall i < t$). However, the gain in the dual objective for this simple update rule is minimal. To achieve the largest gain in the dual objective, one can optimize all the $\boldsymbol{\lambda}_i$'s at each step. But this approach is usually computationally prohibitive to use since at each step, we need to solve a large optimization problem:

$$(\boldsymbol{\lambda}_1^{t+1}, ..., \boldsymbol{\lambda}_t^{t+1}) \in \underset{\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_t}{\arg\max}\, D_{t+1}(\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_t)$$

A compromise approach is to fully optimize the dual objective function at each time step t but only with respect to the last variable $\boldsymbol{\lambda}_t$:

$$\boldsymbol{\lambda}_i^{t+1} = \begin{cases} \boldsymbol{\lambda}_i^t & \text{if } i < t \\ \arg\max_{\boldsymbol{\lambda}_t} D_{t+1}(\boldsymbol{\lambda}_1^t, ..., \boldsymbol{\lambda}_{t-1}^t, \boldsymbol{\lambda}_t) & \text{if } i = t \end{cases}$$

This is called the Coordinate-Dual-Ascent (CDA) update rule. If we can find a closed-form solution of the optimization problem with respect to the last variable $\boldsymbol{\lambda}_t$, then the computational complexity of the CDA update is similar to the simple update but the gain in the dual objective function is larger. Previous work (Shalev-Shwartz & Singer, 2007b) showed that algorithms which more aggressively ascend the dual function have better performance.

2.6 Evaluation Metrics

In this section, we briefly review some standard metrics for evaluating the predictions produced by a model. For MLNs, all the query literals are binary (i.e either true or false). So, there are only four outcomes which are shown in Table 2.1:

Table 2.1: Confusion matrix

		Actual Values	
		True	False
Predicted Values	True	True Positive (TP)	False Positive (FP)
	False	False Negative (FN)	True Negative (TN)

Below are definitions of some standard evaluation metrics:

- Accuracy: the proportion of corrected predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

- Precision: the proportion of correted true predictions.

$$\text{Precision} = \frac{TP}{TP + FP}$$

- Recall: the proportion of true literals that are correcly predicted.

$$\text{Precision} = \frac{TP}{TP + FN}$$

- F_1: the harmonic mean of precision and recall.

$$F_1 = 2\frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Chapter 3

Discriminative Structure and Weight Learning for MLNs with Non-recursive Clauses

3.1 Introduction

In this chapter, we look at a special class of MLNs where all the clauses are non-recursive clauses which contain only one non-evidence literal. Non-recursive clauses arise in many learning problems in ILP such as the structure activity relationship prediction (SAR) problem mentioned in chapter 1. For those problems, there is a specific *target predicate* that must be inferred given evidence data about other *background predicates* used to describe the input data. We have found that existing structure learning algorithms for MLNs (Kok & Domingos, 2005; Mihalkova & Mooney, 2007) perform very poorly when tested on several benchmark ILP problems since they are non-discriminative.

Thus, we present a new method that discriminatively learns both the structure and parameter for MLN with non-recursive clauses. The proposed approach first uses an off-the-shelf ILP system to generate a large set of good candidate clauses, then utilizes l_1-regularization with exact inference to learn weights for those candidate clauses.

The remainder of the chapter is organized as follows. Section 3.2 presents the proposed method. Section 3.3 reports experimental evaluation. Section 3.4 discusses related work and section 3.5 summarizes the chapter.

3.2 The Proposed Method

3.2.1 Discriminative Structure Learning

Ideally, the search for discriminative MLN clauses would be directly guided by the goal of maximizing their contribution to the predictive accuracy of a complete MLN. However, this would require evaluating every proposed refinement to the existing set of learned clauses by relearning weights for all of the clauses and performing full probabilistic inference to determine the score of the revised model. This process is computationally expensive and would have to be repeated for each of the combinatorially large number of potential clause refinements. Evaluating clauses in standard ILP is quicker since each clause can be evaluated in isolation based on the accuracy of its logical inferences about the target predicate. Consequently, we take the heuristic approach of using a standard ILP method to generate clauses; however, since the logical accuracy of a clause is only a rough approximation of its value in a final MLN, we generate a large number of candidates whose accuracy is at least markedly greater than random guessing and allow subsequent weight learning to determine their value to an overall MLN.

In order to find a set of potentially good clauses for an MLN, we use a particular configuration of ALEPH. Specifically, we use the **induce_cover** command and **m-estimate** evaluation function. The **induce_cover** command implements a variant of PROGOL's MDIE greedy covering algorithm (Muggleton, 1995) which does *not* remove previously covered examples when scoring a new clause. The normal ALEPH **induce** command scores a clause based only on its coverage of currently uncovered positive examples. However, this scoring is not reflective of its use in a final MLN, and we found that the **induce_cover** approach produces a larger set of more useful clauses that significantly increases the accuracy of our final learned MLN. The m-estimate (Džeroski, 1991) is a Bayesian estimation of the accuracy of a clause (Cussens, 2007). The m parameter defining the underlying prior distribution is automatically set to the maximum likelihood estimate of its best value. The output of **induce_cover** is a theory, a set of high-scoring clauses that cover all the positive examples. However, these clauses were selected based on an m-estimate of their accuracy under a purely logical interpretation, and may not be the best ones for an MLN. Therefore, in addition to these clauses, we also save all generated clauses whose m-estimate is greater than a predefined threshold (set to 0.6 in our experiments). This provides a large set of clauses of potential utility for an MLN. We use the name ALEPH++ to refer to this version of ALEPH.

3.2.2 Discriminative Weight Learning

Compared to ALCHEMY's previously best discriminative weight learning method (Lowd & Domingos, 2007), our method embodies two important modifications: *exact inference* and l_1-*regularization*. This section describes these two modifications.

First, given the restricted nature of the clauses constructed by ALEPH, we can use an efficient exact probabilistic inference method when learning the weights instead of the approximate inference algorithm that is used to handle the general case. Since these clauses are non-recursive clauses in which the target predicate only appears once, a grounding of any clause will contain only one grounding of the target predicate. For MLNs, this means that the Markov blanket of a query atom only contains evidence atoms. Consequently, the query atoms are independent given the evidence. Let Y be the set of query atoms and X be the set of evidence atoms, the conditional log likelihood of Y given X in this case is:

$$\begin{aligned}\log P(Y = y|X = x) &= \log \prod_{j=1}^{n} P(Y_j = y_j|X = x) \\ &= \sum_{j=1}^{n} \log P(Y_j = y_j|X = x)\end{aligned}$$

and,

$$P(Y_j = y_j|X = x) =$$
$$\frac{exp(\sum_{i \in \mathcal{C}_{Y_j}} w_i n_i(x, y_{[Y_j=y_j]}))}{exp(\sum\limits_{i \in \mathcal{C}_{Y_j}} w_i n_i(x, y_{[Y_j=0]})) + exp(\sum\limits_{i \in \mathcal{C}_{Y_j}} w_i n_i(x, y_{[Y_j=1]}))}$$

where $\mathcal{C}_{Y_j}$ is the set of all MLN clauses with at least one grounding containing the query atom Y_j, $n_i(x, y_{[Y_j=y_j]})$ is the number groundings of the ith clause that evaluate to true when all the evidence atoms in X and the query atom Y_j

are set to their truth values, and similarly for $n_i(x, y_{[Y_j=0]})$ and $n_i(x, y_{[Y_j=1]})$ when Y_j is set to 0 and 1 respectively. Then the gradient of the CLL is:

$$\frac{\partial}{\partial w_i} \log P(Y = y|X = x) =$$
$$\sum_{j=1}^{n} [n_i(x, y_{[Y_j=y_j]}) - P(Y_j = 0|X = x) n_i(x, y_{[Y_j=0]})$$
$$-P(Y_j = 1|X = x) n_i(x, y_{[Y_j=1]})]$$

Notice that the sum of the last two terms in the gradient is the expected count of the number of true groundings of the i'th formula. In general, computing this expected count requires performing approximate inference under the model. For example, Singla and Domingos (2005) ran MPE inference and used the counts in the MPE state to approximate the expected counts. However, in our case, using the standard closed world assumption for evidence predicates, all the n_i's can be computed without approximate inference since there is no ground atom whose truth value is unknown. This is a result of restricting the structure learner to non-recursive clauses. In fact, this result still holds even when the clauses are not Horn clauses. The only restriction is that the target predicates appear only once in every clause. Note that given a set of weights, computing the conditional probability $P(y|x)$, the CLL, and its gradient requires only the n_i counts. So, in our case, the conditional probability $P(Y_j = y_j|X = x)$, the CLL, and its gradient can be computed exactly. In addition, these counts only need to be computed once, and ALCHEMY provides an efficient method for computing them. ALCHEMY also provides an efficient way to construct the Markov blanket of a query atom, in particular it ignores all ground formulae whose truth values are unaffected by the value of the query atom. In our case, this helps reduce the size of the Markov blanket of a query atom significantly since many ground clauses are satisfied by the evidence. As a result, our exact inference is very fast even when the MLN contains thousands of clauses.

Given a procedure for computing the CLL and its gradient, standard gradient-based optimization methods can be used to find a set of weights that optimizes the CLL. However, to prevent overfitting and select only the best clauses, we follow the approach suggested by Lee, Ganapathi, and Koller (2007) and introduce a *Laplacian* prior with zero mean, $P(w_i) = (\beta/2) \cdot exp(-\beta|w_i|)$, on each weight, and then optimize the posterior conditional log likehood instead of the CLL. The final objective function is:

$$\begin{aligned}\log P(Y|X)P(w) &= \log P(Y|X) + \log P(w)\\ &= \log P(Y|X) + \log(\prod_i P(w_i))\\ &= CLL + \sum_i \log\left(\frac{\beta}{2} \cdot exp(-\beta|w_i|)\right)\\ &= CLL - \beta \sum_i |w_i| + constant\end{aligned}$$

There is now an additional term $\beta \sum_i |w_i|$ in the objective function, which penalizes each non-zero weight w_i by $\beta|w_i|$. So, the larger β is (corresponding to a smaller variance of the prior distribution), the more we penalize non-zero weights. Therefore, placing a *Laplacian* prior with zero mean on each weight is equivalent to performing an l_1-regularization of the parameters. An important property of l_1-regularization is its tendency to force parameters to zero by strongly penalizing small terms (Lee et al., 2007). In order to learn weights that optimize the l_1-regularized CLL, we use the OWL-QN package which implements the Orthant-Wise Limited-memory Quasi-Newton algorithm (Andrew & Gao, 2007).

This approach to preventing over-fitting contrasts with the standard l_2-regularization used in previous work on learning weights for MLNs, which is equivalent to assuming a Guassian prior with zero mean on each weight and does not penalize non-zero weights as severely. Since ALEPH++ generates a very large number of potential clauses, l_1-regularization encourages eliminating the less useful ones by setting their weights to zero. In agreement with prior results on l_1-regularization (Ng, 2004; Dudík, Phillips, & Schapire, 2007), our experiments confirm that it results in simpler and more accurate learned models compared to l_2-regularization.

3.3 Experimental Evaluation

In this section, we present experiments that were designed to answer the following questions:

1. How does our method compare to existing methods, specifically:
   (a) Extant learning methods for MLNs.
   (b) Traditional ILP methods, viz. ALEPH.

   (c) "Advanced" ILP methods, viz. kFOIL (Landwehr, Passerini, Raedt, & Frasconi, 2006), TFOIL (Landwehr, Kersting, & Raedt, 2007), and RUMBLE (Rückert & Kramer, 2007).

2. How does each of our system's major novel components below contribute to its performance:

   (a) Generation of a larger set of potential clauses by using ALEPH++ instead of ALEPH.

   (b) Exact MLN inference for non-recursive clauses instead of general approximate inference.

   (c) l_1-regularization instead of l_2.

3.3.1 Data

We employed four benchmark data sets previously used to evaluate a variety of ILP and relational learning algorithms. They concern predicting the relative biochemical activity of variants of Tacrine, a drug for Alzheimer's disease (King et al., 1995). The data contain background knowledge about the physical and chemical properties of substituents such as their hydrophobicity and polarity, the relations between various physical and chemical constants, and other relevant information. The goal is to compare various drugs on four important biochemical properties: low **toxicity**, high **acetyl** cholinesterase inhibition, good reversal of scopolamine-induced **memory** impairment, and inhibition of **amine** re-uptake. For each property, the positive and negative

Table 3.1: Some background evidence and examples from the Alzheimer toxic dataset.

Background evidence	**Examples**
r_subst_1(A1,H), r_subst_1(B1,H), r_subst_1(D1,H),	less_toxic(B1,A1)
x_subst(B1,7,CL), polar(CL,POLAR3), size(CL,SIZE1),	less_toxic(A1,D1)
alk_groups(A1,0), alk_groups(B1,0), alk_groups(D1,0)	less_toxic(B1,D1)

examples are pairwise comparisons of drugs. For example, $less\_toxic(d_1, d_2)$ means that drug d_1's toxicity is less than d_2's. These ordering relations are transitive but not complete (i.e. for some pairs of drugs it is unknown which one is better). Therefore, this is a structured (a.k.a. collective) prediction problem since the output labels should form a partial order. However, previous work has ignored this structure and just predicted the examples separately as distinct binary classification problems. In this work, in addition to treating the problem as independent classification, we also use an MLN to perform structured prediction by explicitly imposing the transitive constraint on the target predicate. Table 3.1 shows some background facts and examples from one of the datasets, and Table 3.2 summarizes information about all four datasets.

Table 3.2: Summary statistics for Alzheimer's data sets.

Data set	**#Examples**	**% Positive**	**# Predicates**
Alzheimer acetyl	1326	50%	30
Alzheimer amine	686	50%	30
Alzheimer memory	642	50%	30
Alzheimer toxic	886	50%	30

3.3.2 Methodology

To answer the above questions, we ran experiments with the following systems:

Alchemy: Uses the structure learning algorithm MSL (Kok & Domingos, 2005) in Alchemy and the most accurate existing discriminative weight learning PSCG (Lowd & Domingos, 2007) with the "ne" (non-evidence) parameter set to the target predicate.

Busl: Uses Busl (Mihalkova & Mooney, 2007) and PSCG discriminative weight learning with the "ne" (non-evidence) parameter set to the target predicate.

Aleph: Uses Aleph's standard settings with a few modifications. The maximum number of literals in an acceptable clause was set to 5. The minimum number of positive examples covered by an acceptable clause was set to 2. The upper bound on the number of negative examples covered by an acceptable clause was set to 300. The evaluation function was set to **auto_m**, and the minimum score of an acceptable clause was set to 0.6. The **induce_cover** command was used to learn the clauses. We found that this configuration gave somewhat better overall accuracy compared to those reported in previous work.

AlephPSCG: Uses the discriminative weight learner PSCG to learn MLN weights for the clauses in the final theory returned by Aleph. Note that PSCG also uses l_2-regularization.

Aleph**ExactL2** : Uses the limited-memory BFGS algorithm (Liu & Nocedal, 1989) implemented in Alchemy to learn discriminative MLN weights for the clauses in the final theory returned by Aleph. The objective function is CLL with l_2 regularization. The CLL is computed exactly as described in Section 3.2.2.

Aleph**++PSCG:** Like AlephPSCG, but learns weights for the larger set of clauses returned by Aleph++.

Aleph**++ExactL2:** Like AlephExactL2, but learns weights for the larger set of clauses returned by Aleph++.

Aleph**++ExactL1:** Our full proposed approach using exact inference and l_1-regularization to learn weights on the clauses returned by Aleph++.

To force the predictions for the target predicate to properly constitute a partial ordering, we also tried adding to the learned MLNs a hard constraint (i.e. a clause with infinite weight) stating the transitive property of the target predicate, and used the MC-SAT algorithm to perform prediction on the test data. This exploits the ability of MLNs to perform collective classification for the complete set of test examples.

In testing, only the background facts are provided as evidence to ensure that all predictions are based on the chemical structure of a drug. For all systems except Aleph, a threshold of 0.5 was used to convert predicted probabilities into boolean values. The predictive accuracy of these algorithms

Table 3.3: Average predictive accuracies and standard deviations for all systems. Bold numbers indicate the best result on a data set.

Data set	ALCHEMY	BUSL	ALEPH	ALEPH **PSCG**	ALEPH **ExactL2**	ALEPH++ **PSCG**	ALEPH++ **ExactL2**	ALEPH++ **ExactL1**
Alzheimer amine	50.1 ± 0.5	51.3 ± 2.5	81.6 ± 5.1	64.6± 4.6	83.5 ± 4.7	72.0± 5.2	86.8± 4.4	**89.4 ± 2.7**
Alzheimer toxic	54.7 ± 7.4	51.7 ± 5.3	81.7 ± 4.2	74.7± 1.9	87.5 ± 4.8	69.9± 1.2	89.5± 3.0	**91.3 ± 2.8**
Alzheimer acetyl	48.2 ± 2.9	55.9 ± 8.7	79.6 ± 2.2	78.0± 3.2	79.5 ± 2.0	76.5± 3.7	82.1± 2.1	**85.1 ± 2.4**
Alzheimer memory	50 ± 0.0	49.8 ± 1.6	76.0 ± 4.9	60.3± 2.1	72.6 ± 3.4	65.6± 5.4	72.9± 5.2	**77.6 ± 4.9**

Table 3.4: Average AUC-ROC and standard deviations for all systems. Bold numbers indicate the best result on a data set.

Data set	ALCHEMY	BUSL	ALEPH **PSCG**	ALEPH **ExactL2**	ALEPH++ **PSCG**	ALEPH++ **ExactL2**	ALEPH++ **ExactL1**
Alzheimer amine	.483 ± .115	.641 ± .110	.846 ± .041	.904 ± .027	.777 ± .052	.935 ± .032	**.954 ± .019**
Alzheimer toxic	.622 ± .079	.511 ± .079	.904 ± .034	.930 ± .035	.874 ± .041	.937 ± .029	**.939 ± .035**
Alzheimer acetyl	.473 ± .037	.588 ± .108	.850 ± .018	.850 ± .020	.810 ± .040	.899 ± .015	**.916 ± .013**
Alzheimer memory	.452± .088	.426 ± .065	.744 ± .040	.768 ± .032	.737 ± .059	.813 ± .059	**.844 ± .052**

for the target predicate were compared using 10-fold cross-validation. The significance of the results were evaluated using a two-tailed paired t-test test with a 95% confidence level. To compare the quality of the predicted probabilities, we also report the average area under the ROC curve (AUC-ROC) (Provost, Fawcett, & Kohavi, 1998) for all probabilistic systems by using the AUCCalculator package (Davis & Goadrich, 2006).

3.3.3 Results and Discussion

Tables 3.3 and 3.4 show the average accuracy and AUC-ROC with standard deviation for each system running on each data set. Our complete system (ALEPH++**ExactL1**) achieves significantly higher accuracy than both ALCHEMY and BUSL on all 4 data sets and significantly higher than ALEPH

Table 3.5: Average predictive accuracies and standard deviations for MLN systems with transitive clause added.

Data set	Alchemy	BUSL	Aleph **PSCG**	Aleph **ExactL2**	Aleph++ **PSCG**	Aleph++ **ExactL2**	Aleph++ **ExactL1**
Alzheimer amine	50.0 ± 0.0	52.2 ± 5.3	61.4 ± 3.6	87.0 ± 3.3	72.9± 3.5	**91.7**± **3.5**	90.5 ± 3.6
Alzheimer toxic	50.0 ± 0.0	50.1 ± 0.8	73.3 ± 1.8	88.8 ± 4.8	68.4± 1.5	91.4± 3.6	**91.9** ± **4.1**
Alzheimer acetyl	53.0 ± 6.2	54.1 ± 4.9	80.4 ± 2.7	84.1 ± 3.1	83.3± 2.5	**88.7**± **2.1**	87.6 ± 2.7
Alzheimer memory	50.0 ± 0.0	50.1 ± 0.5	58.9 ± 2.3	76.5 ± 3.5	70.1± 5.2	81.3± 4.8	**81.3** ± **4.1**

Table 3.6: Average number of clauses learned

Data set	Aleph++	Aleph++ **ExactL2**	Aleph++ **ExactL1**
Alzheimer amine	7061	5070	3477
Alzheimer toxic	2034	1194	747
Alzheimer acetyl	8662	5427	2433
Alzheimer memory	6524	4250	2471

on all except the **memory** data set, answering questions 1(a) and 1(b). In turn, Aleph has been shown to give higher accuracy on these data sets than other standard ILP systems like Foil (Landwehr et al., 2007). Both MSL and BUSL find only a few (3–5) simple clauses. Two of them are unit clauses for the target predicate, such as *great_ne(a1,a1)* and *great_ne(a1,a2)*; the others capture the transitive nature of the target relation. Therefore, even after they are discriminatively weighted, their predictions are not significantly better than random guessing.

The ablations that remove components from our overall system demonstrate the important contribution of each component. Regarding question 2(b), the systems using general approximate inference (Aleph**PSCG** and

Table 3.7: Average predictive accuracies and standard deviations of our best results and other "advanced" ILP systems.

Data set	Our best results	tFOIL	kFOIL	Rumble
Alzheimer amine	**91.7± 3.5**	87.5 ± 4.4	88.8 ± 5.0	91.1
Alzheimer toxic	91.9 ± 4.1	**92.1 ± 2.6**	89.3 ± 3.5	91.2
Alzheimer acetyl	**88.7± 2.1**	82.8 ± 3.8	87.8 ± 4.2	88.4
Alzheimer memory	81.3 ± 4.1	80.4 ± 5.3	80.2 ± 4.0	**83.2**

Aleph++**PSCG**) perform much worse than the corresponding versions that use exact inference (Aleph**ExactL2** and Aleph++**ExactL2**). Therefore, when there is a target predicate that can be accurately inferred using non-recursive clauses, exploiting this restriction to perform exact inference is a clear win.

Regarding question 2(a), Aleph++**ExactL2** performs significantly better than Aleph**ExactL2**, demonstrating the advantage of learning a large set of potential clauses and combining them with learned weights in an overall MLN. Across the four datasets, Aleph++ returns an average of 6,070 clauses compared to only 10 for Aleph.

Table 3.5 presents average accuracies with standard deviations for the MLN systems when we include a transitivity clause for the target predicate. This constraint improves the accuracies of Aleph**ExactL2**, Aleph++**ExactL2**, and Aleph++**ExactL1**, but sometimes decreases the accuracy of other systems, such as Aleph**PSCG**. This can be explained as follows. Since most of the predictions of Aleph++**ExactL1** are correct, enforcing transitivity can correct some of the wrong ones. However, Aleph**PSCG** produces many wrong predictions, so forcing them to obey transitivity can produce additional incorrect predictions.

Regarding question 2(c), using l_1-regularization gives significantly higher accuracy and AUC-ROC than using standard l_2-regularization. This comparison was only performed for ALEPH++ since this is when the weight-learner must choose from a large set of candidate clauses by encouraging zero weights. Table 3.6 compares the average number of clauses learned (after zero-weight clauses are removed) for l_1 and l_2 regularization. As expected, the final learned MLNs are much simpler when using l_1-regularization. On average, l_1-regularization reduces the size of the final set of clauses by 26% compared to l_2-regularization.

Regarding question 1(c), several researchers have tested "advanced" ILP systems on our datasets. Table 3.7 compares our best results to those reported for TFOIL (a combination of FOIL and tree augmented naive Bayes), kFOIL (a kernelized version of FOIL), and RUMBLE (a max-margin approach to learning a weighted rule set). Our results are competitive with these recent systems. Additionally, unlike MLNs, these methods do not create "declarative" theories that have a well-defined possible worlds semantics.

3.4 Related Work

Using an off-the-shelf ILP system to learn clauses for MLNs is not a new idea. Richardson and Domingos (2006) used CLAUDIEN, an non-discriminative ILP system that can learn arbitrary first-order clauses, to learn MLN structure and to refine the clauses from a knowledge base. Kok and Domingos (2005) reported experimental results comparing their MLN structure learner to learning clauses using CLAUDIEN, FOIL, and ALEPH. However, since this previous work used the relatively small set of clauses produced by these unaltered ILP systems, the performance was not very good. ILP systems have also been used to learn structures for other SRL models. The SAYU system (Davis, Burnside, de Castro Dutra, Page, & Costa, 2005) used ALEPH to propose candidate features for a Bayesian network classifier. Muggleton(Muggleton, 2000) used PROGOL, another popular ILP system, to learn clauses for Stochastic Logic Programs (SLPs).

When restricted to learning non-recursive clauses for classification, our approach is equivalent to using ALEPH to construct features for use by l_1-regularized logistic regression. Under this view, our approach is closely related to MACCENT (Dehaspe, 1997), which uses a greedy approach to induce clausal constraints that are used as features for maximum-entropy classification. One difference between our approach and MACCENT is that we use a two-step process instead of greedily adding one feature at a time. In addition, our clauses are induced in a bottom-up manner while MACCENT uses top-down search; and our weight learner employs l_1-regularization which makes it less prone to overfitting. Unfortunately, we could not compare experimentally to MACCENT since "only an implementation of a propositional version of MACCENT is available, which only handles data in attribute-value (vector) format" (Landwehr et al., 2007). Additionally, MLNs are a more expressive formalism that also allows for structured prediction, as demonstrated by our results that include a transitivity constraint on the target relation.

3.5 Chapter Summary

We have found that existing methods for learning Markov Logic Networks perform very poorly when tested on several benchmark ILP problems in drug design. We present a new approach to discriminatively learns both the structure and parameter of an MLN with non-recursive clauses. The proposed approach uses a variant of an existing ILP system (Aleph) to construct a large number of potential clauses and then effectively learns their parameters by altering existing discriminative MLN weight-learning methods to utilize exact inference and l_1 regularization. Experimental results show that the resulting system outperforms existing MLN and ILP methods and gives state-of-the-art results for the Alzheimer's-drug benchmarks.

Chapter 4

Max-Margin Weight Learning for MLNs

4.1 Introduction

In the previous chapter, we aim to learn a model that maximizes the CLL of the data. If the goal is to predict accurate target-predicate probabilities, that approach is well motivated. However, in many applications, the actual goal is to maximize an alternative performance metric such as classification accuracy or F-measure. Max-margin training provides a framework for maximizing a variety of performance metrics (Joachims, 2005). In this chapter, we present a max-margin approach to weight learning in MLNs based on the general framework of max-margin training for structured prediction (section 2.4).

The remainder of the chapter is organized as follows. Section 4.2 formulates the max-margin weight learning problem. Section 4.3 discusses approximate inference for MLNs based on Linear Programming (LP) relaxation. Section 4.4 reports experimental evaluation. Section 4.5 discusses related work and section 4.6 summarizes the chapter.

4.2 Max-Margin Formulation

All of the current discriminative weight learners for MLNs try to find a weight vector $\mathbf{w}$ that optimizes the conditional log-likelihood $P(\mathbf{y}|\mathbf{x})$ of the query atoms $\mathbf{y}$ given the evidence $\mathbf{x}$. However, an alternative approach is to learn a weight vector $\mathbf{w}$ that maximizes the ratio:

$$\frac{P(\mathbf{y}|\mathbf{x})}{P(\hat{\mathbf{y}}|\mathbf{x})}$$

between the probability of the correct truth assignment $\mathbf{y}$ and the closest competing incorrect truth assignment $\hat{\mathbf{y}} = \arg\max_{\bar{\mathbf{y}} \in \mathbf{Y} \setminus \mathbf{y}} P(\bar{\mathbf{y}}|\mathbf{x})$. Applying equation 2.1 and taking the log, this problem translates to maximizing the margin:

$$\begin{aligned}\gamma(\mathbf{x}, \mathbf{y}; \mathbf{w}) &= \mathbf{w}^T \mathbf{n}(\mathbf{x}, \mathbf{y}) - \mathbf{w}^T \mathbf{n}(\mathbf{x}, \hat{\mathbf{y}}) \\ &= \mathbf{w}^T \mathbf{n}(\mathbf{x}, \mathbf{y}) - \max_{\bar{\mathbf{y}} \in \mathbf{Y} \setminus \mathbf{y}} \mathbf{w}^T \mathbf{n}(\mathbf{x}, \bar{\mathbf{y}})\end{aligned}$$

Note that, this translation holds for all log-linear models (Collins, 2004). For example, if we apply it to a CRF (Lafferty, McCallum, & Pereira, 2001) then the resulting model is an M3N (Taskar et al., 2004). Similarly, when changing the objective of MLNs to maximize the margin, we create a max-margin version of MLNs, abbreviated as M3LNs.

In turn, the max-margin problem above can be formulated as a "1-slack" structural SVM as described in section 2.4:

Optimization Problem 4 (OP4): Max-Margin Markov Logic Networks

$$\min_{\mathbf{w},\xi\geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$$
$$s.t.\ \forall \bar{\mathbf{y}} \in Y : \mathbf{w}^T[\mathbf{n}(\mathbf{x},\mathbf{y}) - \mathbf{n}(\mathbf{x},\bar{\mathbf{y}})] \geq \Delta(\mathbf{y},\bar{\mathbf{y}}) - \xi$$

So for MLNs, the number of true groundings of the clauses $\mathbf{n}(\mathbf{x},\mathbf{y})$ plays the role of the feature vector function $\boldsymbol{\phi}(\mathbf{x},\mathbf{y})$ in the general structural SVM problem. In other words, each clause in an MLN can be viewed as a feature representing a dependency between a subset of inputs and outputs or a relation among several outputs.

As mentioned, in order to apply Algorithm 2.1 to MLNs, we need algorithms for solving the following two problems:

Prediction: $\arg\max_{\mathbf{y}\in Y} \mathbf{w}^T\mathbf{n}(\mathbf{x},\mathbf{y})$

Separation Oracle: $\arg\max_{\bar{\mathbf{y}}\in Y}\{\Delta(\mathbf{y},\bar{\mathbf{y}}) + \mathbf{w}^T\mathbf{n}(\mathbf{x},\bar{\mathbf{y}})\}$

The prediction problem is just the (intractable) MPE inference problem discussed in section 2.3. We can use MaxWalkSAT to get an approximate solution, but we have found that models trained with MaxWalkSAT have very low predictive accuracy. On the other hand, recent work (Finley & Joachims, 2008) has found that fully-connected pairwise Markov random fields, a special class of structural SVMs, trained with overgenerating approximate inference methods (such as relaxation) preserves the theoretical guarantees of structural SVMs trained with exact inference, and exhibits good empirical performance. Based on this result, we sought a relaxation-based approximation for MPE inference. We first present an LP-relaxation algorithm for MPE inference, then show how to modify it to solve the separation oracle problem for some specific loss functions.

4.3 Approximate Inference

4.3.1 Approximate MPE inference for MLNs

MPE inference in MLNs is a special case of MAP inference in Markov networks with binary variables, and there has been a lot of work on approximation algorithms for solving MAP inference using convex relaxation, see Kumar, Kolmogorov, and Torr (2009) for more details. However, these methods are not suitable for MLNs. First, most of them are for Markov networks with unary and pairwise potential functions while a ground MLN may contain many high-order cliques. The algorithms can be extended to handle high-order potential functions (Werner, 2008), but they become computationally expensive. Second, they do not handle deterministic factors, i.e. potential functions with some entries are zero. On the other hand, MPE inference in MLNs is equivalent to the Weighted MAX-SAT problem, and there are also significant work on approximating this NP-hard problem using LP-relaxation (Asano & Williamson, 2002; Asano, 2006). The existing algorithms first relax and convert the Weighted MAX-SAT problem into a linear or semidefinite programming problem, then solve it and apply a randomized rounding method to obtain an approximate integral solution. These methods cannot be directly applied to MLNs, since they require the weights to be positive while MLN weights can be negative or infinite. However, we can modify the conversion used in these approaches to handle the case of negative and infinite weights.

Based on the evidence and the closed world assumption, a ground MLN contains only ground clauses of the unknown ground atoms after removing all trivially satisfied and unsatisfied clauses. The following procedure translates the MPE inference in a ground MLN into an Integer Linear Programming problem.

1. Assign a binary variable y_i to each unknown ground atom. y_i is 1 if the corresponding ground atom is $TRUE$ and 0 if the ground atom is $FALSE$.

2. For each ground clause C_j with infinite weight, add the following linear constraint to the Integer Linear Programming problem:

   $$\sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i) \geq 1$$

   where I_j^+, I_j^- are the sets of positive and negative ground literals in clause C_j respectively.

3. For each ground clause C_j with positive weight w_j, introduce a new auxiliary binary variable z_j, add the term $w_j z_j$ to the objective function, and add the following linear constraint to the Integer Linear Programming problem:

   $$\sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i) \geq z_j$$

   z_j is 1 if the corresponding ground clause is satisfied.

4. For each ground clause C_j with k ground literals and negative weight w_j, introduce a new auxiliary boolean variable z_j, add the term $-w_j z_j$ to the objective function and add the following k linear constrains to the Integer Linear Programming problem:

$$1 - y_i \geq z_j, \quad i \in I_j^+$$
$$y_i \geq z_j, \quad i \in I_j^-$$

The final Integer Linear Programming has the following form:

Optimization Problem 5 (OP5):

$$\begin{aligned}
&\max_{y_i, z_i} \sum_{C_j \in C^+} w_j z_j + \sum_{C_j \in C^-} -w_j z_j \\
&s.t. \sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i) \geq 1 \quad \forall\, C_j \text{ } where \text{ } w_j = \infty \\
&\sum_{i \in I_j^+} y_i + \sum_{i \in I_j^-} (1 - y_i) \geq z_j \quad \forall C_j \in C^+ \\
&1 - y_i \geq z_j \quad \forall\, i \in I_j^+ \text{ } and \text{ } C_j \in C^- \\
&y_i \geq z_j \quad \forall\, i \in I_j^- \text{ } and \text{ } C_j \in C^- \\
&y_i, z_j \in \{0, 1\}
\end{aligned}$$

where C^+ and C^- are the set of clauses with positive and negative weights respectively. This Integer Linear Programming problem can be simplified by not introducing an auxiliary variable z_j for unit clauses, where we can use the variable y_i directly. This reduces the problem considerably, since ground MLNs typically contain many unit clauses (Alchemy combines all the non-recursive clauses containing the query atom into a unit clause whose weight is the sum of all the clauses' weights). Note that our mapping from a ground MLN to an Integer Linear Programming problem is a bit different from the one presented by Riedel (2008) which generates two sets of constraints for every ground clause: one when the clause is satisfied and one when it is not. For a clause with positive weight, our mapping only generates a constraint when the clause is satisfied; and for a clause with negative weight, the mapping only imposes constraints when the clause is unsatisfied. The final Integer Linear Programming problem has the same solution with the one in (Riedel, 2008), but it has fewer constraints since our mapping does not generate unnecessary constraints. We then relax the integer constraints $y_i, z_j \in \{0, 1\}$ to linear constraints $y_i, z_j \in [0, 1]$ to obtain an LP-relaxation of the MPE problem.

This LP problem can be solved by any general LP solver. If the LP solver returns an integral solution, then it is also the optimal solution to the original Integer Linear Programming problem. In our case, the original Integer Linear Programming problem is an NP-hard problem, so the LP solver usually returns non-integral solutions. Therefore, the LP solution needs to be rounded to give an approximate Integer Linear Programming solution. We first tried some of the randomized rounding methods in (Asano, 2006) but they gave poor results since the LP solution has a lot of fractional components with value 0.5. We then adapted a rounding procedure called ROUNDUP (Boros & Hammer, 2002), a procedure for producing an upper bound binary solution for a pseudo-Boolean function, to the case of pseudo-Boolean functions with linear constraints (algorithm 4.1), which we found to work well. In each step,

Algorithm 4.1 The modified ROUNDUP procedure

```
1: Input: The LP solution $\mathbf{y} = \{y_1, ..., y_n\}$
2: $F \leftarrow \emptyset$
3: for $i = 1$ to $n$ do
4:     if $y_i$ is integral then
5:         Remove all the ground clauses satisfied by assigning the value of $y_i$ to the
           corresponding ground atom
6:     else
7:         add $y_i$ to F
8:     end if
9: end for
10: repeat
11:     Remove the last item $y_i$ in $F$
12:     Compute the sum $w^+$ of the unsatisfied clauses where $y_i$ appears as a positive
        literal
13:     Compute the sum $w^-$ of the unsatisfied clauses where $y_i$ appears as a negative
        literal
14:     if $w^+ > w^-$ then
15:         $y_i \leftarrow 1$
16:     else
17:         $y_i \leftarrow 0$
18:     end if
19:     Remove all the ground clauses satisfied by assigning the value of $y_i$ to the
        corresponding ground atom
20: until $F$ is empty
21: return $\mathbf{y}$
```

this procedure picks one fractional component and rounds it to 1 or 0. Hence, this process terminates in at most n steps, where n is the number of query atoms. Note that due to the dependencies between the variables y_i's and z_j's (the linear constraints of the LP problem), this modified ROUNDUP procedure does not guarantee an improvement in the value of the objective function in each step like the original ROUNDUP procedure where all the variables are independent.

4.3.2 Approximation algorithm for the separation oracle

The separation oracle adds an additional term, the loss term, to the objective function. So, if we can represent the loss as a linear function of the y_i variables of the LP-relaxation, then we can use the above approximation algorithm to also approximate the separation oracle. In this work, we consider two loss functions. The first one is the 0/1 loss function, $\Delta_{0/1}(\mathbf{y^T}, \mathbf{y})$ where $\mathbf{y^T}$ is the true assignment and $\mathbf{y}$ is some predicted assignment. For this loss function, the separation oracle is the same as the MPE inference problem since the loss function only adds a constant 1 to the objective function. Hence, in this case, to find the most violated constraint, we can use the LP-relaxation algorithm above or any other MPE inference algorithm. This 0/1 loss makes the separation oracle problem easier but it does not scale the margin by how different $\mathbf{y^T}$ and $\mathbf{y}$ are. It only requires a unit margin for all assignments $\mathbf{y}$ different from the true assignment $\mathbf{y^T}$. To take into account this problem, we consider the second loss function that is the number of misclassified atoms or the Hamming loss:

$$\begin{aligned}\Delta_{Hamming}(\mathbf{y^T}, \mathbf{y}) &= \sum_i^n [y_i^T \neq y_i] \\ &= \sum_i^n [(y_i^T = 0 \wedge y_i = 1) \vee (y_i^T = 1 \wedge y_i = 0)]\end{aligned}$$

From the definition, this loss can be represented as a function of the y_i's:

$$\Delta_{Hamming}(\mathbf{y^T}, \mathbf{y}) = \sum_{i:y_i^T=0} y_i + \sum_{i:y_i^T=1} (1 - y_i)$$

which is equivalent to adding 1 to the coefficient of y_i if the true value of y_i is 0 and subtracting 1 from the coefficient of y_i if the true value of y_i is 1. So we can use the LP-relaxation algorithm above to approximate the separation oracle with this Hamming loss function. Another possible loss function is the F_1 loss which is equivalent to $1 - F_1$. Unfortunately, this loss is a non-linear function, so we cannot use the above approach to optimize it. Developing algorithms for optimizing or approximating this loss function is an area for future work.

4.4 Experimental Evaluation

This section presents experiments comparing the max-margin weight learner to the weight learners in section 3.2 and the PSCG algorithm.

4.4.1 Datasets

Besides those Alzheimer's datasets described in section 3.3.1, we also ran experiments on two other large, real-world datasets: WebKB for collective web-page classification, and CiteSeer for bibliographic citation segmentation.

The WebKB dataset, mentioned in chapter 1, consists of labeled web pages from the computer science departments of four universities. Different versions of this data have been used in previous work. To make a fair comparison, we used the version from (Lowd & Domingos, 2007), which contains 4,165 web pages and 10,935 web links. Each page is labeled with a subset of the categories: *course*, *department*, *faculty*, *person*, *professor*, *research project*, and *student*. The goal is to predict these categories from the words and links on the web pages. We used the same simple MLN from (Lowd & Domingos, 2007), which only has clauses relating words to page classes, and page classes to the classes of linked pages.

$$Has(+word, page) \Rightarrow PageClass(+class, page)$$
$$\neg Has(+word, page) \Rightarrow PageClass(+class, page)$$
$$PageClass(+c1, p1) \wedge Linked(p1, p2) \Rightarrow PageClass(+c2, p2)$$

The plus notation creates a separate clause for each pair of word and page class, and for each pair of classes. The final MLN consists of 10,891 clauses, and a weight must be learned for each one. After grounding, each department results in an MLN with more than 100,000 ground clauses and 5,000 query atoms in a complex network. This also results in a large LP-relaxation problem for MPE inference.

For CiteSeer (Lawrence, Giles, & Bollacker, 1999), we used the version created by Poon and Domingos (Poon & Domingos, 2007). The dataset contains 1,563 bibliographic citations such as:

J. Jaffar, J. - L. Lassez. Constraint logic programming. In Proceedings of the Fourteenth ACM symposium of the principles of programming languages, pages 111-119, Munich, 1987.

The task is to segment each of these citations into three fields: *Author*, *Title* and *Venue*. The dataset has four independent subsets consisting of citations to disjoint publications in four different research areas. We used the MLN for isolated segmentation model in (Poon & Domingos, 2007). After grounding, this model results in a large network with more than 30,000 query atoms and 110,000 ground clauses.

All the datasets except Alzheimer's datasets and MLNs can be found at the Alchemy website.[1]

4.4.2 Methodology

For the max-margin weight learner, we used a simple process for selecting the value of the C parameter. For each train/test split, we trained the algorithm with five different values of C: 1, 10, 100, 1000, and 10000, then selected the one which gave the highest average F_1 score on training. The ϵ parameter was set to 0.001. To solve the QP problems in Algorithm 2.1 and LP problems in the LP-relaxation MPE inference, we used the Mosek [2] solver. The PSCG algorithm was carefully tuned by its author. For MC-SAT, we used the default setting, 100 burn-in and 1000 sampling iterations, and predict that an atom is true iff its probability is at least 0.5.

For the Alzheimer's datasets, we used the same experimental setup mentioned in section 3.3.2, and ran four-fold cross-validation (i.e. leave one university/topic out) on the WebKB and CiteSeer datasets.

We used F_1 to measure the performance of each algorithm on the WebKB and CiteSeer datasets.

[1]http://alchemy.cs.washington.edu
[2]http://www.mosek.com/

Table 4.1: F_1 scores on WebKB

	Cornell	Texas	Washington	Wisconsin	Average
PSCG-MCSAT	0.418	0.298	0.577	0.568	0.465
PSCG-LPRelax	0.420	0.310	0.588	0.575	0.474
MM-$\Delta_{0/1}$-MaxWalkSAT	0.150	0.162	0.122	0.122	0.139
MM-$\Delta_{0/1}$-LPRelax	0.282	0.372	0.675	0.521	0.462
MM-$\Delta_{Hamming}$-LPRelax	**0.580**	**0.451**	**0.715**	**0.659**	**0.601**

Table 4.2: F_1 scores of different inference algorithms on WebKB

	Cornell	Texas	Washington	Wisconsin	Average
PSCG-MCSAT	0.418	0.298	0.577	0.568	0.465
PSCG-MaxWalkSAT	0.161	0.140	0.119	0.129	0.137
PSCG-LPRelax	0.420	0.310	0.588	0.575	0.474
MM-$\Delta_{Hamming}$-MCSAT	0.470	0.370	0.573	0.481	0.473
MM-$\Delta_{Hamming}$-MaxWalkSAT	0.185	0.184	0.150	0.154	0.168
MM-$\Delta_{Hamming}$-LPRelax	0.580	0.451	0.715	0.659	0.601

4.4.3 Results and Discussion

Table 4.1 and 4.3 present the performance of different systems on the WebKB and Citeseer datasets. Each system is named by the weight learner used, the loss function used in training, and the inference algorithm used in testing. For max-margin (MM) learner with margin rescaling, the inference used in training is the loss-augmented version of the one used in testing. For example, MM-$\Delta_{Hamming}$-LPRelax is the max-margin weight learner using the loss-augmented (Hamming loss) LP-relaxation MPE inference algorithm in training and the LP-relaxation MPE inference algorithm in testing.

Table 4.1 shows that the model trained using MaxWalkSAT has very low predictive accuracy. This result is consistent with the result presented in (Riedel, 2008) which also found that the MPE solution found by MaxWalkSAT

Table 4.3: F_1 scores on CiteSeer

	Constraint	Face	Reasoning	Reinforcement	Average
PSCG-MCSAT	0.937	0.914	0.931	0.975	0.939
MM-$\Delta_{Hamming}$-LPRelax	0.933	0.922	0.924	0.958	0.934

Table 4.4: F_1 scores on CiteSeer with different parameter values

	Constraint	Face	Reasoning	Reinforcement	Average
PSCG-MCSAT-5	0.852	0.844	0.836	0.923	0.864
PSCG-MCSAT-10	0.937	0.914	0.931	0.973	0.939
PSCG-MCSAT-15	0.878	0.896	0.780	0.891	0.861
PSCG-MCSAT-20	0.850	0.859	0.710	0.784	0.801
PSCG-MCSAT-100	0.658	0.697	0.600	0.668	0.656
MM-$\Delta_{Hamming}$-LPRelax-1	0.933	0.922	0.924	0.955	0.934
MM-$\Delta_{Hamming}$-LPRelax-10	0.926	0.922	0.925	0.955	0.932
MM-$\Delta_{Hamming}$-LPRelax-100	0.926	0.922	0.925	0.954	0.932
MM-$\Delta_{Hamming}$-LPRelax-1000	0.931	0.918	0.925	0.958	0.933
MM-$\Delta_{Hamming}$-LPRelax-10000	0.932	0.922	0.919	0.968	0.935

is not very accurate. Using the proposed LP-relaxation MPE inference improves the F_1 score from 0.139 to 0.462, the MM-$\Delta_{0/1}$-LPRelax system. Then the best system is obtained by rescaling the margin and training with our loss-augmented LP-relaxation MPE inference, which is the only difference between MM-$\Delta_{Hamming}$-LPRelax and MM-$\Delta_{0/1}$-LPRelax. The MM-$\Delta_{Hamming}$-LPRelax achieves the best F_1 score (0.601), which is much higher than the 0.465 F_1 score obtained by the previously best discriminative weight learner for MLNs, PSCG-MCSAT.

Table 4.2 compares the performance of the proposed LP-relaxation MPE inference algorithm against MCSAT and MaxWalkSAT on the best trained models by PSCG and MM on the WebKB dataset. In both cases, the LP-relaxation MPE inference achieves much better F_1 scores than those

Table 4.5: Average predictive accuracies and standard deviations on Alzheimer's datasets with transitive clause added

Data set	Aleph ExactL2	Aleph++ ExactL2	Aleph++ ExactL1	Aleph MM-LPRelax	Aleph++ MM-LPRelax	Aleph++ MM-L1-LPRelax
Alzheimer amine	87.0 ± 3.3	91.7± 3.5	90.5 ± 3.6	87.0 ± 2.2	89.2± 2.9	88.8 ± 3.0
Alzheimer toxic	88.8 ± 4.8	91.4± 3.6	91.9 ± 4.1	88.5 ± 4.2	90.8± 3.6	91.6 ± 4.3
Alzheimer acetyl	84.1 ± 3.1	88.7± 2.1	87.6 ± 2.7	86.3 ± 2.8	88.3± 2.9	87.9 ± 2.8
Alzheimer memory	76.5 ± 3.5	81.3± 4.8	81.3 ± 4.1	79.1 ± 3.0	81.5± 4.2	80.7 ± 4.0

Table 4.6: Average number of clauses learned on Alzheimer's datasets

Data set	Aleph	Aleph++	Aleph++ ExactL2	Aleph++ ExactL1	Aleph++ MM-LPRelax	Aleph++ MM-L1-LPRelax
Alzheimer amine	10	7061	5070	3477	6981	35
Alzheimer toxic	9	2034	1194	747	2034	25
Alzheimer acetyl	12	8662	5427	2433	8621	51
Alzheimer memory	11	6524	4250	2471	6297	31

of MCSAT and MaxWalkSAT. This demonstrates that the approximate MPE solution found by the LP-relaxation algorithm is much more accurate than the one found by the MaxWalkSAT algorithm. The fact that the performance of the LP-relaxation is higher than that of MCSAT shows that in collective classification it is better to use the MPE solution as the prediction than the marginal prediction.

For the WebKB dataset, there are other results reported in previous work, such as those in (Taskar et al., 2004), but those results cannot be directly compared to our results since we use a different version of the dataset and test on a more complicated task (a page can have multiple labels not just one).

On the Citeseer results presented in Table 4.3, the performance of max-margin methods are very close to those of PSCG. However, its performance is much more stable. Table 4.4 shows the performance of MM weight learners and PSCG with different parameter values by varying the C value for MM and the number of iterations for PSCG. The best number of iterations for PSCG is 9 or 10. In principle, we should run PSCG until it converges to get the optimal weight vector. However, in this case, the performance of PSCG drops drastically on both training and testing after a certain number of iterations. For example, from Table 4.4 we can see that at 10 iterations PSCG achieves the best F_1 score of 0.939, but after 15 iterations, its F_1 score drops to 0.861 which is much worse than those of the max-margin weight learners. Moreover, if we let it run until 100 iterations, then its F_1 score is only 0.656. On the other hand, the performance of MM only varies a little bit with different values of C and we don't need to tune the number of iterations of MM. On this dataset, (Poon & Domingos, 2007) achieved a F_1 score of 0.944 with the same MLN by using a version of the voted perceptron algorithm called Contrastive Divergence (CD) (Hinton, 2002) to learn the weights. However, the performance of the CD algorithm is very sensitive to the learning rate (Lowd & Domingos, 2007), which requires a very careful tuning process to learn a good model.

Table 4.5 and 4.6 compares the performance of the MM weight learners against the some of the systems described in section 3.2 for the case when the transitive clause is included. For the MM weight learner, instead of adding the transitive clause to the learnt MLNs in testing, we learned the weights with the presence of the transitive clause since it can handle recursive clauses. In term of the accuracy, the MM weight learner is a little bit worse than the ones proposed in the previous chapter. However, the 1-norm MM weight learner (MM-L1-LPRelax) produces a very compact model, with less than 50 clauses, with high accuracy while the models learnt by other systems have thousands of clauses.

Regarding training time, the max-margin weight learner is comparable to other learners. On the Alzheimer's datasets, it took less than 100 iterations to find the optimal weights, which resulted in a few minutes of training. For the WebKB and CiteSeer datasets, the number of training iterations are about 200 and 50 respectively, which takes a few hours of training for WebKB and less than an hour for CiteSeer.

4.5 Related Work

The work in this chapter is related to various previous projects. Among them, M3N (Taskar et al., 2004) is probably the most related. It is a special case of structural SVMs where the feature function $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$ is represented by a Markov network. When the Markov network can be triangulated and the loss function can be linearly decomposed, the original exponentially-sized QP can be reformulated as a polynomially-sized QP (Taskar et al., 2004). Then, the polynomially-sized QP can be solved by general QP solvers (Anguelov, Taskar, Chatalbashev, Koller, Gupta, Heitz, & Ng, 2005), decomposition methods (Taskar et al., 2004), extragradient methods (Taskar, Lacoste-Julien, & Jordan, 2006), or exponentiated gradient methods (Collins, Globerson, Koo, Carreras, & Bartlett, 2008). As mentioned by Taskar et al. (2004), these methods can also be used when the graph cannot be triangulated, but the algorithms only yield approximate solutions like our approach. However, these algorithms are restricted to the cases where a polynomially-sized reformulation exists (Joachims et al., 2009). Consequently, in this work we used the general cutting plane algorithm which imposes no restrictions on the representation. The ground MLN can be any kind of graph. On the other hand, since an MLN is a template for constructing Markov networks (Richardson & Domingos, 2006), the proposed model, M3LN, can also be seen as a template for constructing M3Ns. Hence, when the ground MLN can be triangulated and the loss is a linearly decomposable function, the algorithms developed for M3Ns can be applied. Our work is also closely related to the Relational Markov Networks (RMNs) (Taskar et al., 2002). However, by using MLNs, M3LNs are more powerful than RMNs in term of representation (Richardson & Domingos, 2006). Besides, the objectives of M3LNs and RMNs are different. One tries to maximize the margin between the true assignment and other competing assignments, and one tries to maximize the conditional likelihood of the true assignment. Another related system is RUMBLE (Rückert & Kramer, 2007), a margin-based approach to first-order rule learning. In that work, the goal is to find a set of weighted rules that maximizes a quantity called margin minus variance. However, unlike M3LNs, RUMBLE only applies to independent binary classification problems and is unable to perform structured prediction or collective classification. In terms of applying the general structural SVM framework to a specific representation, our work is related to the work in (Szummer, Kohli, & Hoiem, 2008) which used CRFs as the representation and graph cuts as the inference algorithm. In the context of discriminative learning, our work is related to previous work on discriminative training for MLNs (Singla & Domingos, 2005; Lowd & Domingos, 2007; Biba et al., 2008). We have mentioned some of them (Singla & Domingos, 2005; Lowd & Domingos, 2007) in previous sections. The main difference between the work in (Biba et al., 2008) and ours is that we assume the structure is given and apply max-margin framework to learn the weights while (Biba et al., 2008) tries to learn a structure that maximizes the conditional likelihood of the data.

4.6 Chapter Summary

We have presented a max-margin weight learning method for MLNs based on the framework of structural SVMs. It resulted in a new model, M3LN, that has the representational expressiveness of MLNs and the predictive performance of SVMs. M3LNs can be trained to optimize different performance measures depending on the needs of the application. To train the proposed model, we developed a new approximation algorithm for loss-augmented MPE inference in MLNs based on LP-relaxation. The experimental results showed that the new max-margin learner generally has better or equally good but more stable predictive accuracy (as measured by F_1) than the previously best discriminative MLN weight learner.

Chapter 5

Online Max-Margin Weight Learning for MLNs

5.1 Introduction

In the previous chapter, we presented a max-margin algorithm to learning weights for MLNs. However, like other existing weight learning algorithms for MLNs, the algorithm uses batch training which becomes computationally expensive and even infeasible for very large datasets since the training examples may not fit in main memory. To address this issue, in this chapter, we derive a new online max-margin algorithm for structured prediction from the primal-dual framework for strongly convex loss functions (section 2.5).

The remainder of the chapter is organized as follows. Section 5.2 presents the new online max-margin algorithm for structured prediction. Section 5.3 reports experimental evaluation. Section 5.4 discusses related work and section 5.5 summarizes the chapter.

5.2 Online Coordinate-Dual-Ascent Algorithms for Max-Margin Structured Prediction

In this section, we derive new online algorithms for structured prediction based on the algorithmic framework described in section 2.5 using the CDA update rule. A standard complexity function used in structured prediction is $f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2$. Regarding the loss function g_t, a generalized version of the Hinge loss is widely used in max-margin structured prediction (Taskar et al., 2004; Tsochantaridis et al., 2004):

$$l_{MM}(\mathbf{w}, (\mathbf{x}_t, \mathbf{y}_t)) =$$
$$\max_{y \in \mathcal{Y}}[\Delta(\mathbf{y}_t, \mathbf{y}) - \langle \mathbf{w}, (\boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}) \rangle]_+$$

However, minimizing the above loss results in an optimization problem with a lot of constraints in the primal (one constraint for each possible label $\mathbf{y} \in \mathcal{Y}$) which is usually expensive to solve. To overcome this problem, we consider two simpler variants of the max-margin loss which only involves a particular label: the *maximal* loss function and the *prediction-based* loss function.

Maximal loss (ML) function This loss function is based on the maximal loss label at step t, $\mathbf{y}_t^{ML} = \arg\max_{\mathbf{y} \in \mathbf{Y}}\{\Delta(\mathbf{y}_t, \mathbf{y}) + \langle \mathbf{w}_t, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}) \rangle\}$:

$$l_{ML}(\mathbf{w}, (\mathbf{x}_t, \mathbf{y}_t)) =$$
$$\left[\Delta(\mathbf{y}_t, \mathbf{y}_t^{ML}) - \left\langle \mathbf{w}, \left(\boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t^{ML})\right) \right\rangle\right]_+$$

The loss $l_{ML}(\mathbf{w}_t, (\mathbf{x}_t, \mathbf{y}_t))$ is the greatest loss the algorithm would suffer at step t if it used the maximal loss label $\mathbf{y}_t^{ML}$ as the prediction. On the other hand, it checks whether the max-margin constraints are satisfied since if $l_{ML}(\mathbf{w}_t, (\mathbf{x}_t, \mathbf{y}_t)) = 0$ then $\mathbf{y}_t^{ML} = \mathbf{y}_t$, and it means that the current weight vector $\mathbf{w}_t$ scores the correct label $\mathbf{y}_t$ higher than any other label $\mathbf{y}'_t$ where the difference is at least $\Delta(\mathbf{y_t}, \mathbf{y}'_t)$. Note that the maximal loss label $\mathbf{y}_t^{ML}$ is the input to the maximal loss (it is possible in online learning since the loss is computed after the weight vector $\mathbf{w}_t$ is chosen), therefore it does not depend on the weight vector $\mathbf{w}$ for which we want to compute the loss. So the maximal loss function only concerns the particular constraint for whether the true label $\mathbf{y}_t$ is scored higher than the maximal loss label with a margin of $\Delta(\mathbf{y_t}, \mathbf{y}_t^{ML})$. This is the key difference between the maximal loss and the max-margin loss since the latter looks at the constraints of all possible labels. The main drawback of the maximal loss is that finding the maximal loss label $\mathbf{y}_t^{ML}$, which is also called the loss-augmented inference problem (section 2.4), is only feasible for some decomposable label loss functions such as Hamming loss since the maximal loss label depends on the label loss function $\Delta(\mathbf{y}_t, \mathbf{y}')$. This is the reason why we want to consider the second loss function, prediction-based loss, which can be used with any label loss function such as $(1 - F_1)$ loss.

Prediction-based loss (PL) function This loss function is based on the predicted label $\mathbf{y}_t^P = h_{\mathbf{w}_t}(\mathbf{x}_t) = \arg\max_{\mathbf{y} \in \mathbf{Y}} \langle \mathbf{w}_t, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}) \rangle$:

$$\begin{aligned} &l_{PL}(\mathbf{w}, (\mathbf{x}_t, \mathbf{y}_t)) = \\ &\quad \big[\Delta(\mathbf{y}_t, \mathbf{y}_t^P) - \big\langle \mathbf{w}, \big(\boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t^P)\big)\big\rangle\big]_+ \end{aligned}$$

Like the maximal loss, the prediction-based loss only concerns the constraint for the prediction label $\mathbf{y}_t^P$. We have $l_{PL}(\mathbf{w}_t, (\mathbf{x}_t, \mathbf{y}_t)) \leq l_{ML}(\mathbf{w}_t, (\mathbf{x}_t, \mathbf{y}_t))$ since $\mathbf{y}_t^{ML}$ is the maximal loss label for $\mathbf{w}_t$. As a result, the update based on the prediction-based loss function is less aggressive than the one based on the maximal loss function. However, the prediction-based loss function can be used with any label loss function since the predicted label $\mathbf{y}_t^P$ does not depend on the label loss function.

To apply the primal-dual algorithmic framework described in section 2.5, we need to find the Fenchel conjugate function of the complexity function $f(\mathbf{w})$ and the loss function $g(\mathbf{w})$. The Fenchel conjugate function of the complexity function $f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2$ is itself, i.e. $f^*(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{\theta}||_2^2$ (Boyd & Vandenberghe, 2004). For the loss function, recall that the Fenchel conjugate function of the Hinge-loss $g(\mathbf{w}) = [\gamma - \langle \mathbf{w}, \mathbf{x} \rangle]_+$ is:

$$g^*(\boldsymbol{\theta}) = \begin{cases} -\gamma\alpha & \text{if } \boldsymbol{\theta} \in \{-\alpha\mathbf{x} : \alpha \in [0,1]\} \\ \infty & \text{otherwise} \end{cases}$$

(Appendix A in (Shalev-Shwartz & Singer, 2007a)). We can see that both the prediction-based loss and the maximal loss have the same form as the Hinge-loss where γ is replaced by the label loss function $l(\mathbf{y}_t, \mathbf{y}_t^P)$ and $l(\mathbf{y}_t, \mathbf{y}_t^{ML})$, and $\mathbf{x}$ is replaced by $\Delta\boldsymbol{\phi}_t^{PL} = \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t^P)$ and $\Delta\boldsymbol{\phi}_t^{ML} = \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t^{ML})$ for the prediction-based loss and the maximal loss respectively. Using the result of the Hinge-loss, we have the Fenchel conjugate function of the prediction-based loss and the maximal loss as follows:

$$g_t^*(\boldsymbol{\theta}) = \begin{cases} -\Delta(\mathbf{y}_t, \mathbf{y}_t^{P|ML})\alpha & \text{if } \boldsymbol{\theta} \in \{-\alpha\Delta\boldsymbol{\phi}_t^{PL|ML} : \alpha \in [0,1]\} \\ \infty & \text{otherwise} \end{cases}$$

The next step is to derive the closed-form solution of the CDA update rule. The optimization problem that we need to solve is:

$$\operatorname{argmax}_{\boldsymbol{\lambda}_t} -(\sigma t) f^*\left(-\frac{\boldsymbol{\lambda}_{1:(t-1)} + \boldsymbol{\lambda}_t}{(\sigma t)}\right) - g_t^*(\boldsymbol{\lambda}_t) \tag{5.1}$$

where $\boldsymbol{\lambda}_{1:(t-1)} = \sum_{i=1}^{t-1} \boldsymbol{\lambda}_i$. Substituting the conjugate function f^* and g_t^* as above in the equation 5.1, we obtain the following optimization problem:

$$\begin{aligned}
&\arg\max_{\alpha\in[0,1]} -\frac{(\sigma t)}{2}\left|\left|-\frac{\boldsymbol{\lambda}_{1:(t-1)} - \alpha\Delta\boldsymbol{\phi}_t^{PL|ML}}{(\sigma t)}\right|\right|_2^2 + \alpha\Delta(\mathbf{y}_t, \mathbf{y}_t^{P|ML}) \\
&= \arg\max_{\alpha\in[0,1]} -\alpha^2\frac{||\Delta\boldsymbol{\phi}_t^{PL|ML}||_2^2}{2(\sigma t)} - \frac{||\boldsymbol{\lambda}_{1:(t-1)}||_2^2}{2(\sigma t)} \\
&\qquad + \alpha\left(\Delta(\mathbf{y}_t, \mathbf{y}_t^{P|ML}) + \frac{1}{(\sigma t)}\left\langle \boldsymbol{\lambda}_{1:(t-1)}, \Delta\boldsymbol{\phi}_t^{PL|ML}\right\rangle\right)
\end{aligned}$$

This objective function is a function of α only and in fact it is a concave parabola whose maximum attains at the point:

$$\alpha^* = \frac{(\sigma t)\Delta(\mathbf{y}_t, \mathbf{y}_t^{P|ML}) + \left\langle \boldsymbol{\lambda}_{1:(t-1)}, \Delta\boldsymbol{\phi}_t^{PL|ML}\right\rangle}{||\Delta\boldsymbol{\phi}_t^{PL|ML}||_2^2}$$

If $\alpha^* \in [0,1]$, then α^* is the maximizer of the problem. If $\alpha^* < 0$, then 0 is the maximizer and if $\alpha^* > 1$ then 1 is the maximizer. In summary, the solution of the above optimization is:

$$\begin{aligned}
&\alpha^{max} = \\
&\quad \min\left\{1, \frac{\left[(\sigma t)\Delta(\mathbf{y}_t, \mathbf{y}_t^{P|ML}) + \left\langle \boldsymbol{\lambda}_{1:(t-1)}, \Delta\boldsymbol{\phi}_t^{PL|ML}\right\rangle\right]_+}{||\Delta\boldsymbol{\phi}_t^{PL|ML}||_2^2}\right\}
\end{aligned}$$

To obtain the update in terms of the weight vectors $\mathbf{w}$, we have:

$$\begin{aligned}
\mathbf{w}_{t+1} &= \nabla f^* \left(-\frac{1}{\sigma t}\boldsymbol{\lambda}_{1:t}\right) \\
&= -\frac{1}{\sigma t}(\boldsymbol{\lambda}_{1:(t-1)} + \boldsymbol{\lambda}_t) \\
&= -\frac{\boldsymbol{\lambda}_{1:(t-1)}}{\sigma t} - \frac{1}{\sigma t}(-\alpha^{max}\Delta\boldsymbol{\phi}_t^{PL|ML}) \\
&= -\frac{-(\sigma(t-1))\mathbf{w}_t}{\sigma t} + \\
&\frac{1}{\sigma t}\min\left\{1, \frac{\left[(\sigma t)\Delta(\mathbf{y}_t, \mathbf{y}_t^{P|ML}) + \left\langle -(\sigma(t-1))\mathbf{w}_t, \Delta\boldsymbol{\phi}_t^{PL|ML}\right\rangle\right]_+}{||\Delta\boldsymbol{\phi}_t^{PL|ML}||_2^2}\right\}\Delta\boldsymbol{\phi}_t^{PL|ML} \\
&= \frac{t-1}{t}\mathbf{w}_t + \\
&\min\left\{\frac{1}{\sigma t}, \frac{\left[\Delta(\mathbf{y}_t, \mathbf{y}_t^{P|ML}) - \frac{t-1}{t}\left\langle \mathbf{w}_t, \Delta\boldsymbol{\phi}_t^{PL|ML}\right\rangle\right]_+}{||\Delta\boldsymbol{\phi}_t^{PL|ML}||_2^2}\right\}\Delta\boldsymbol{\phi}_t^{PL|ML}
\end{aligned}$$

The new method is summarized in Algorithm 5.1. Interestingly, this update formula has the same form as that of the subgradient algorithm (Nathan Ratliff & Zinkevich, 2007) which is derived from the simple update criterion:

$$\begin{aligned}
\mathbf{w}_{t+1} &= \mathbf{w}_t - \frac{1}{\sigma t}(\sigma\mathbf{w}_t - \Delta\boldsymbol{\phi}_t^{ML}) \\
&= \frac{t-1}{t}\mathbf{w}_t + \frac{1}{\sigma t}\Delta\boldsymbol{\phi}_t^{ML}
\end{aligned}$$

The key difference is in the learning rate. The learning rate of the subgradient algorithm, which is equal to $1/(\sigma t)$, does not depend on the loss suffered at each step, while the learning rate of CDA is the minimization of $1/(\sigma t)$ and the loss suffered at each step. In the beginning, when t is small and therefore $1/(\sigma t)$ is large (assuming σ is small), CDA's learning rate is controlled by the

Algorithm 5.1 Online Coordinate-Dual-Ascent Algorithms for Structured Prediction

```
1: Parameters: A constant $\sigma > 0$; Label loss function $\Delta(\mathbf{y}, \mathbf{y}')$
2: Initialize: $\mathbf{w}_1 = 0$
3: for $i = 1$ to $T$ do
4:     Receive an instance $\mathbf{x}_t$
5:     Predict $\mathbf{y}_t^P = \arg\max_{\mathbf{y} \in \mathbf{Y}} \langle \mathbf{w}_t, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}) \rangle$
6:     Receive the correct target $\mathbf{y}_t$
7:     (For maximal loss) Compute $\mathbf{y}_t^{ML} = \arg\max_{\mathbf{y} \in \mathbf{Y}} \{\Delta(\mathbf{y}_t, \mathbf{y}) + \langle \mathbf{w}_t, \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}) \rangle\}$
8:     Compute $\Delta\boldsymbol{\phi}_t$:
8:     PL: $\Delta\boldsymbol{\phi}_t = \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t^P)$
8:     ML: $\Delta\boldsymbol{\phi}_t = \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t^{ML})$
9:     Compute loss:
9:     PL (CDA): $l_t = \left[\Delta(\mathbf{y}_t, \mathbf{y}_t^P) - \frac{t-1}{t} \langle \mathbf{w}_t, \Delta\boldsymbol{\phi}_t \rangle\right]_+$
9:     ML (CDA): $l_t = \left[\Delta(\mathbf{y}_t, \mathbf{y}_t^{ML}) - \frac{t-1}{t} \langle \mathbf{w}_t, \Delta\boldsymbol{\phi}_t \rangle\right]_+$
10:    Update:
10:    CDA: $\mathbf{w}_{t+1} = \frac{t-1}{t} \mathbf{w}_t + \min\{1/(\sigma t), \frac{l_t}{||\Delta\boldsymbol{\phi}||_2^2}\} \Delta\boldsymbol{\phi}_t$
11: end for
```

loss suffered at each step. In contrast, when t is large and therefore $1/(\sigma t)$ is small, then the learning rate of CDA is driven by the quantity $1/(\sigma t)$. In other words, at the beginning, when the model is not good, CDA aggressively updates the model based on the loss suffered at each step; and later when the model is good, it updates the model less aggressively.

We can use the derived CDA algorithm to perform online weight learning for MLNs since the weight learning problem in MLNs can be cast as a max-margin structured prediction problem as described in 4.2.

5.3 Experimental Evaluation

In this section, we conduct experiments to answer the following questions in the context of MLNs:

1. How does our new online learning algorithm, CDA, compare to existing online max-margin learning methods? In particular, is it better than the subgradient method due to its more aggressive update in the dual?
2. How does it compare to the batch max-margin weight learning method developed in the previous chapter?
3. How well does using the prediction-based loss compare to the maximal loss in practice?

5.3.1 Datasets

We ran experiments on three large, real-world datasets with thousands of examples: the CiteSeer dataset for bibliographic citation segmentation described in 4.4.1, a web search query dataset (Mihalkova & Mooney, 2009) obtained from Microsoft Research for query disambiguation, and the CoNLL 2005 dataset (Carreras & Màrquez, 2005) for Semantic Role Labeling. We did not run experiments on Alzheimer's datasets and WebKB dataset since those are datasets with a few mega-examples (when taking into account the transitive relationship, each Alzheimer's dataset becomes a mega-example).

For the search query disambiguation, we used the data created by Mihalkova and Mooney (2009). The dataset consists of thousands of search sessions where ambiguous queries are asked. The data are split into 3 disjoint sets: training, validation, and test. There are $4,618$ search sessions in the training set, $4,803$ sessions in the validation set, and $11,234$ sessions in the test set. In each session, the set of possible search results for a given ambiguous query is given, and the goal is to rank these results based on how likely it will be clicked by the user. A user may click on more than one result for a given query. To solve this problem, Mihalkova and Mooney (2009) proposed three different MLNs which correspond to different levels of information used in disambiguating the query. We used all three MLNs in our experiments. In comparison to the Citeseer dataset, the search query dataset is larger but is much noisier since a user can click on a result because it is relevant or because the user is just doing an exploratory search.

The CoNLL 2005 dataset contains over $40,000$ sentences from Wall Street Journal (WSJ). Given a sentence, the task is to analyze the propositions expressed by some target verbs of the sentence. In particular, for each target verb, all of its semantic components must be identified and labeled with their semantic roles as in the following sentence for the verb *accept.*

[$_{A0}$ He] [$_{AM-MOD}$ would] [$_{AM-NEG}$ n't] [$_V$ accept] [$_{A1}$ anything of value] from [$_{A2}$ those he was writing about].

A verb and its set of semantic roles form a proposition in the sentence, and a sentence usually contains more than one proposition. Each proposition serves as a training example. The dataset consists of three disjoint subsets: training, development, and test. The number of propositions (or examples) in the training, development, and test sets are: $90,750$; $3,248$; and $5,267$ respectively.[1] We used the MLN constructed by Riedel (2008) which contains clauses that capture the features of constituents and dependencies between semantic components of the same verb.

5.3.2 Methodology

To answer the above questions, we ran experiments with the following systems:

MM: The offline max-margin weight learner for MLNs presented in previous chapter.

1-best MIRA: MIRA is one of the first online learning algorithms for structured prediction proposed by McDonald, Crammer, and Pereira (2005). A simple version of MIRA, called 1-best MIRA, is widely used in practice since its update rule has a closed-form solution. 1-best MIRA has been used in previous work (Riedel & Meza-Ruiz, 2008) to learn weights for MLNs. In each round, it updates the weight vectors $\mathbf{w}$ as follows:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \frac{\left[\Delta(\mathbf{y}_t, \mathbf{y}_t^P) - \left\langle \mathbf{w}_t, \Delta\boldsymbol{\phi}_t^{PL} \right\rangle\right]_+}{||\Delta\boldsymbol{\phi}_t^{PL}||_2^2} \Delta\boldsymbol{\phi}_t^{PL}$$

Subgradient: This algorithm proposed by Nathan Ratliff and Zinkevich (2007) is an extension of the Greedy Projection algorithm (Zinkevich, 2003) to the case of structured prediction.

[1]We only used the WSJ part of the test set.

CDA: Our newly derived online learning algorithm presented in Algorithm 5.1.

Regarding label loss functions, we use Hamming (HM) loss described in section 4.3.2. As mentioned earlier, Hamming loss is a decomposable loss function, so it can be used with both maximal loss and prediction-based loss. Since F_1 is the standard evaluation metric for the citation segmentation task on Citeseer, we also considered the label loss function $(1 - F_1)$ (Joachims, 2005). However, since this loss function is not decomposable, we can only use it with the prediction-based loss.

In training, for online learning algorithms, since the algorithms process one example at a time it is feasible to use the exact MPE inference method based on Integer Linear Programming described in section 4.3.1 on Citeseer and web search query datasets, and Cutting Plane Inference on the CoNLL 2005 dataset. For the offline weight learner MM, we use the approximate inference algorithm described in section 4.3.1 since it is computationally intractable to run exact inference for all training examples at once. In testing, we use MCSAT to compute marginal probabilities for the web search query dataset since we want to rank the query results, and exact MPE inference on the other two datasets. For all online learning algorithms, we ran one pass over the training set and used the average weight vector to predict on the test set. For CiteSeer, we ran four-fold cross-validation (i.e. leave one topic out). The parameter σ of the Subgradient and CDA is set based on the performance on the validation set except Citeseer where the parameter is set based on training performance.

Like previous work, for citation segmentation on Citeseer, we used F_1 at the token level to measure the performance of each algorithm; for search query disambiguation, we used MAP (Mean Average Precision) which measures how close the relevant results are to the top of the ranking; and for semantic role labeling on CoNLL 2005, we used F_1 of the predicted arguments as described by Carreras and Màrquez (2005).

For testing the statistical significance between the performance of different algorithms, we use McNemar's test (Dietterich, 1998) on Citeseer and a two-sided paired t-test on the web search query. The significance level was set to 5% (p-value smaller than 0.05) for both cases.

5.3.3 Results and Discussion

Table 5.1 presents the F_1 scores of different algorithms on Citeseer. On this dataset, the CDA algorithm with maximal loss, CDA-ML-HM, has the best F_1 scores across four folds. These results are statistically significantly better than those of subgradient method. So aggressive update in the dual results in a better F_1 scores. The F_1 scores of CDA-ML-HM are a little bit higher than those of 1-best-MIRA, but the difference is not significant. Interestingly, with the possibility of using exact inference in training, CDA is a little bit more accurate than the batch max-margin algorithm (MM) since the batch learner can only afford to use approximate inference in training. Other advantages of

Table 5.1: F_1 scores on CiteSeer dataset. Highest F_1 scores are shown in bold.

Algorithms	Constraint	Face	Reasoning	Reinforcement
MM-HM	93.187	92.467	92.581	95.496
1-best-MIRA-HM	90.982	90.598	93.124	97.518
1-best-MIRA-F_1	89.764	90.046	93.200	96.841
Subgradient-HM	90.957	89.859	91.505	95.318
CDA-PL-HM	91.245	90.992	92.589	96.516
CDA-PL-F_1	91.742	92.368	92.726	96.994
CDA-ML-HM	**93.287**	**93.204**	**93.448**	**97.560**

online algorithms are in terms of training time and memory. Table 5.2 shows the average training time of different algorithms on this dataset. All online learning algorithms took on average about 12-13 minutes for training while the batch one took an hour and a half on the same machine. In addition, since online algorithms process one example at a time, they use much less memory than batch methods. On the other hand, the running time results also confirm that the new algorithm, CDA, has the same computational complexity as other existing online methods. Regarding the comparison between maximal loss and prediction-based loss, the former is better than the latter on this dataset due to its more aggressive updates. For prediction-based loss function, there is not much difference between using different label loss functions in this case.

Table 5.3 shows the MAP scores of different algorithms on the Microsoft web search query dataset. The first row in the table is from Mihalkova and Mooney (Mihalkova & Mooney, 2009) who used a variant of the structured perceptron (Collins, 2002) called Contrastive Divergence (CD) (Hinton, 2002)

Table 5.2: Average training time on CiteSeer dataset.

Algorithms	Average training time
MM-HM	90.282 min.
1-best-MIRA-HM	11.772 min.
1-best-MIRA-F_1	11.768 min.
Subgradient-HM	12.655 min.
CDA-PL-HM	11.869 min.
CDA-PL-F_1	11.915 min.
CDA-ML-HM	12.887 min.

Table 5.3: MAP scores on Microsoft search query dataset. Highest MAP scores are shown in bold.

Algorithms	MLN1	MLN2	MLN3
CD	0.375	0.386	0.366
1-best-MIRA-HM	0.366	0.375	0.379
Subgradient-HM	0.374	**0.397**	0.396
CDA-PL-HM	**0.382**	**0.397**	**0.398**
CDA-ML-HM	0.380	**0.397**	0.397

to do online weight learning for MLNs. It is clear that the CDA algorithm has better MAP scores than CD. For this dataset, we were unable to run offline weight learning since the large amount of training data exhausted memory during training. The 1-best MIRA has the worst MAP scores on this dataset. This behavior can be explained as follows. From the update rule of the 1-best MIRA algorithm, we can see that it aggressively updates the weight vector according to the loss incurred in each round. Since this dataset is noisy, this update rule leads to overfitting. This also explains why the subgradient algorithm has good performance on this data since its update rule does not

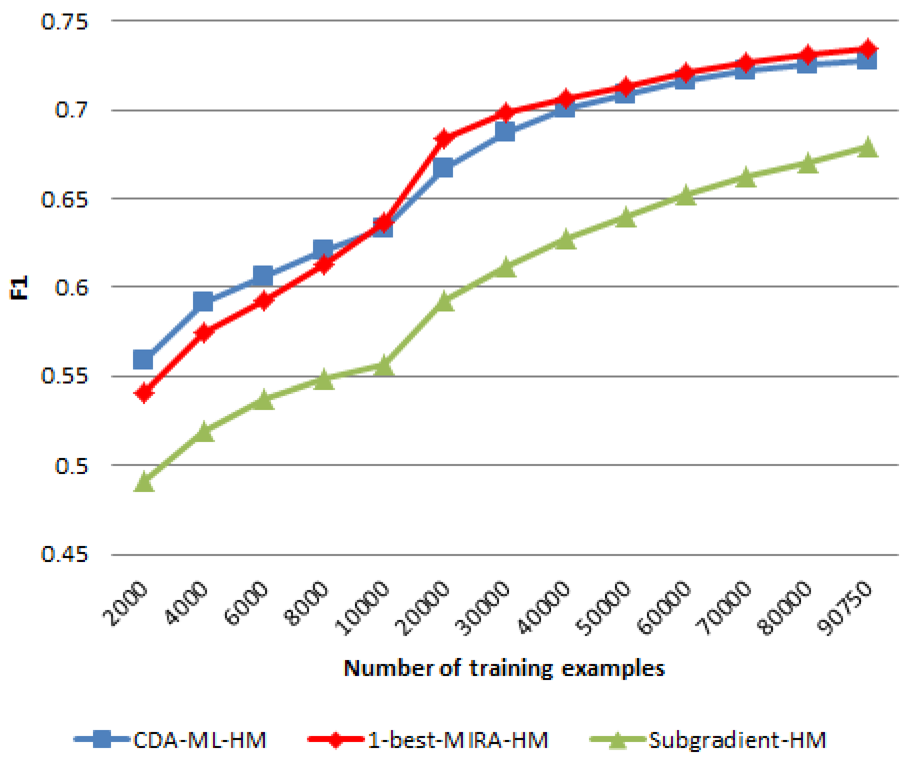

Figure 5.1: Learning curve on CoNLL 2005

depend on the loss incurred in each round. The MAP scores of the CDA algorithms are not significantly better than that of the subgradient method, but their performance is more consistent across the three MLNs. Regarding the loss function, the MAP scores of CDA-PL and CDA-ML are almost the same.

Figure 5.1 shows the learning curve of three online learning algorithms: CDA, 1-best MIRA and subgradient on the CoNLL 2005 dataset. In general, the relative accuracy of three algorithms is similar to what we have seen on

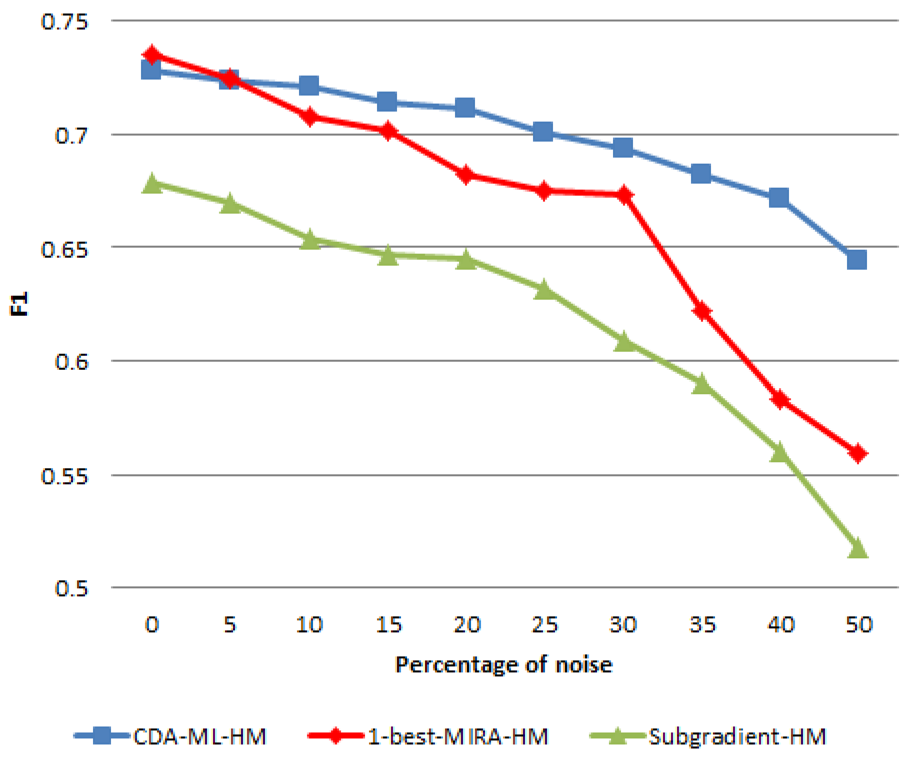

Figure 5.2: F_1 scores on noisy CoNLL 2005

Citeseer. CDA outperforms the subgradient method across the whole learning curve. In particular, at $30,000$ training examples, about $1/3$ of the training set, the F_1 score of CDA is already better than the that of the subgradient method trained on the whole training set. The performance of CDA and 1-best MIRA are comparable to each other, except on the early part of the learning curve (less than $10,000$ examples) where the F_1 scores of CDA are about 1 to 2 percentage points higher than those of 1-best MIRA.

The CoNLL 2005 dataset was carefully annotated by experts (Palmer, Gildea, & Kingsbury, 2005), which is a time consuming and expensive process. Nowadays, a faster and cheaper way to obtain this type of annotation is using crowdsourcing services such as Amazon Mechanical Turk,[2] which is possible to assign annotation jobs to thousands of people and get results back in a few hours (Snow, O'Connor, Jurafsky, & Ng, 2008). However, a downside of this approach is the big variance in the quality of labels obtained from different annotators. As a result, there is a lot of noise in the annotated data. To simulate this type of noisy labeled data, we introduce random noise to the CoNLL 2005 dataset. At p percent noise, there is probability p that an argument in a proposition is swapped with another argument in the same proposition. For example, an argument with role "A0" may be swapped to an argument with role "A1" and vice versa. Figure 5.2 shows the F_1 scores of the above three online learning algorithms on noisy CoNLL 2005 dataset at various levels of noise. With the presence of noise, CDA is the most accurate and also the most robust to noise among the three algorithms. For 10% noise and higher, CDA is significantly better than the other two methods. The F_1 score of CDA at a noise level of 50% is 8.5% higher than that of 1-best MIRA and 12.6% higher than that of the subgradient method. On the other hand, comparing with the F_1 score on the clean dataset, the F_1 score of CDA at 50% of noise only drops 8.4 points while those of 1-best MIRA and subgradient drop about 17.6 and 16.1 respectively. In addition, the F_1 score of CDA at 50% noise is higher than the F_1 score of 1-best MIRA at 35% noise and comparable

[2]`https://www.mturk.com/mturk/`

to the F_1 score of subgradient method at 20% noise.

In summary, our new online learning algorithm CDA has generally better accuracy than existing max-margin online methods for structured prediction such as 1-best MIRA and the subgradient method which have been shown to achieve good performance in previous work. In particular, CDA is significantly better than other methods on noisy datasets.

5.4 Related Work

Online learning for max-margin structured prediction has been studied in several pieces of previous work. In addition to those mentioned earlier, a family of online algorithms similar to the 1-best MIRA, called passive-aggressive algorithms, was presented in (Crammer, Dekel, Keshet, Shalev-Shwartz, & Singer, 2006). Another piece of related work is the exponentiated gradient algorithm (Bartlett, Collins, Taskar, & McAllester, 2005; Collins et al., 2008) which also performs updates based on the dual of the primal problem. However, the dual problem in (Bartlett et al., 2005; Collins et al., 2008) is more complicated and expensive to solve since it was derived based on the max-margin loss, l_{MM}. As a result, to efficiently solve the problem, the authors assume that each label $\mathbf{y}$ is a set of parts and both the joint feature and the label loss function can be decomposed into a sum over those for the individual parts. Even under this assumption, efficiently computing the marginal values of the part variables is still a challenging problem.

In the context of online weight learning for MLNs, one related algorithm is SampleRank (Culotta, 2008) which uses a sampling algorithm to generate samples from a given training example and updates the weight vector whenever it misranks a pair of samples. So unlike traditional online learning algorithms that perform one update per example, SampleRank performs multiple updates per example. However, the performance of SampleRank highly depends on the sampling algorithm, and which sampling algorithms are best is an open research question.

The issue of prediction-based loss versus maximal loss has been discussed previously (Crammer et al., 2006; Shalev-Shwartz, 2007), but no experiments have been conducted to compare them on real-world datasets.

5.5 Chapter Summary

We have presented a comprehensive study of online weight learning for MLNs. Based on the primal-dual framework, we derived a new CDA online algorithm for structured prediction and applied it to learn weights for MLNs and compared it to existing online methods on three large, real-world datasets. Our new algorithm generally achieved better accuracy than existing online methods. In particular, our new algorithm is more accurate and robust when training data is noisy.

Chapter 6

Online Structure Learning for MLNs

6.1 Introduction

In the previous chapter, we derived a new online max-margin weight learning algorithm for MLNs. However, like other existing online algorithms, the algorithm assumes the input MLN's structure is complete and only updates the weights. In practice, the input structure is usually incomplete, so it should also be updated. To address this issue, in this chapter, we propose a new algorithm that performs both online structure and parameter learning.

The remainder of the paper is organized as follows. Section 6.2 provides some background on the field segmentation task. Section 6.3 presents the proposed algorithm OSL. Section 6.4 reports the experimental evaluation on two real-world datasets. Section 6.5 discusses the related work and section 6.6 summarizes the chapter.

6.2 Task

In this chapter, we focus on an information extraction task, called field segmentation which is the generalized version of the citation segmentation task described in previous chapters. Field segmentation is an instance of structured prediction problems where the data contain many examples such as a corpus of documents. In this task, a document is represented as a sequence of tokens, and the goal is to segment the document into fields (i.e. to label each token in the document with a field label). For example, in segmenting advertisements for apartment rentals (Grenager et al., 2005), the goal is to segment each advertisement into fields such as *Features*, *Neighborhood*, *Rent*, *Contact*, and so on. Below are descriptions of some key predicates in this domain:

- $Token(string, position, docID)$: the token at a particular position in a document such as $Token(Entirely, P4, Ad001)$
- $Next(position, position)$: the later position is next to the former position such as $Next(P01, P02)$
- $LessThan(position, position)$: the former position is appeared before the later position such as $LessThan(P01, P05)$
- $InField(field, position, docID)$: the field label of the token at a particular position in a document such as $InField(Features, P4, Ad001)$

Only $InField$ is the target predicate and the rest are evidence predicates.

6.3 Online Max-Margin Structure and Parameter Learning

In this section, we describe OSL—the new online max-margin learning algorithm for updating both the structure and parameters of an MLN. In each step, whenever the model makes wrong predictions on a given example, based on the wrongly predicted atoms OSL finds new clauses that discriminate the ground-truth possible world from the predicted one, then uses an adaptive subgradient method with l_1-regularization to update weights for both old and new clauses. Algorithm 6.1 gives the pseudocode for OSL. Lines 3 to 20 are pseudocode for structure learning and lines 21 to 35 are pseudocode for parameter learning.

6.3.1 Online Max-Margin Structure Learning with Mode-Guided Relational Pathfinding

Most existing structure learning algorithms for MLNs only consider ground-truth possible worlds and search for clauses that improve the likelihood of those possible worlds. However, these approaches may spend a lot of time searching over insignificant clauses that are likely true in most possible worlds. Therefore, instead of only considering ground-truth possible worlds, OSL also takes into account the predicted possible worlds which are the most probable possible worlds predicted by the model. At each step, if the predicted possible world is different from the ground-truth one, then OSL focuses on where the two possible worlds differ and searches for clauses that differentiate them. This is related to the idea of using implicit negative examples in (Zelle, Thompson, Califf, & Mooney, 1995). In this case, each ground-truth possible world plays the role of a positive example in traditional ILP. Making a closed world assumption (Genesereth & Nilsson, 1987), any possible world that differs from the ground-truth possible world is incorrect and can be considered as a negative example (the predicted possible world in this case). In addition, this follows the max-margin training criterion (section 2.4) which discriminates the true label (the ground-truth possible world) from the closest incorrect one (the predicted possible world).

At each time step t, OSL receives an example $\mathbf{x}_t$, produces the predicted label $\mathbf{y}_t^P = \arg\max_{\mathbf{y}\in\mathbf{Y}} \langle \mathbf{w}_{\mathbb{C}}, \boldsymbol{n}_{\mathbb{C}}(\mathbf{x}_t, \mathbf{y})\rangle$, then receives the true label $\mathbf{y}_t$. Given $\mathbf{y}_t$ and $\mathbf{y}_t^P$, in order to find clauses that separate $\mathbf{y}_t$ from $\mathbf{y}_t^P$, OSL first finds atoms that are in $\mathbf{y}_t$ but not in $\mathbf{y}_t^P$, $\Delta y_t = \mathbf{y}_t \setminus \mathbf{y}_t^P$. Then OSL searches the ground-truth possible world $(\mathbf{x}_t, \mathbf{y}_t)$ for clauses that are specific to the true ground atoms in Δy_t.

A simple way to find useful clauses specific to a set of atoms is to use relational pathfinding (Richards & Mooney, 1992), which considers a relational example as a hypergraph with constants as nodes and true ground atoms as hyperedges connecting the nodes that are its arguments, and searches in the hypergraph for paths that connect the arguments of an input literal. A path of hyperedges corresponds to a conjunction of true ground atoms connected by their arguments and can be generalized into a first-order clause by variabilizing their arguments. Starting from a given atom, relational pathfinding searches for all paths connecting the arguments of the given atom. Therefore, relational pathfinding may be very slow or even intractable when there are a large (or exponential) number of paths. To speed up relational pathfinding, we use mode declarations (Muggleton, 1995) to constrain the search for paths. As defined in (Muggleton, 1995), mode declarations are a form of language bias to constrain the search for definite clauses. Since our goal is to use mode declarations for constraining the search space of paths, we introduce a new mode declaration: $modep(r, p)$ for paths. It has two components: a recall number r which is a positive interger, and an atom p whose arguments are place-makers. A place-maker is either '+' (input), '-' (output), or '.' (don't explore). The recall number r limits the number of appearances of the predicate p in a path to r. The place-maker restricts the search of relational pathfinding. Only paths connecting 'input' or 'output' nodes will be considered. A ground atom can only added to a path if one of its arguments has previously appeared as 'input' or 'output' arguments in the path and all of its 'input' arguments are 'output' arguments of previous atoms. Here are some examples of mode declarations for paths:

$$modep(2, Token(., +, .)) \quad modep(1, Next(-, -)) \quad modep(2, InField(., -, .)$$

The above mode declarations require that a legal path contains at most two ground atoms of each predicate *Token* and *InField* and one ground atom of predicate *Next*. Moreover, the second argument of *Token* is an 'input' argument; the second argument of *InField* and all arguments of *Next* are 'output' arguments. Note that, in this case, all 'input' and 'output' arguments are of type position. These 'input' and 'output' modes constrain that the position constants in atoms of *Token* must appeared in some previous atoms of *Next* or *InField* in a path. From the graphical model perspective, these mode declarations restrict the search space to linear chain CRFs

(Sutton & McCallum, 2007) since they constrain the search to paths connecting ground atoms of two consecutive tokens. It is easy to modify the mode declarations to search for more complicated structure. For example, if we increase the recall number of $Next$ to 2 and the recall number of $InField$ to 3, then search space is constrained to second-order CRFs since they constrain the searches to paths connecting ground atoms of three consecutive tokens. If we add a new mode declaration $modep(1, LessThan(-,-))$ for predicate $LessThan$, then the search space becomes skip-chain CRFs (Sutton & McCallum, 2007). Algorithm 6.2 presents the pseudocode for efficiently constructing a hypergraph based on mode declarations by only constructing the hypergraph corresponding to input and output nodes. Algorithm 6.3 gives the pseudocode for mode-guided relational pathfinding, $ModeGuidedFindPaths$, on the constructed hypergraph. It is an extension of a variant[1] of relational pathfinding presented in (Kok & Domingos, 2009). Starting from each true ground atom $r(c_1, ..., c_r) \in \Delta y_t$, it recursively adds to the path ground atoms or hyperedges that satisfy the mode declarations. Its search terminates when the path reaches a specified maximum length or when no new hyperedge can be added. The algorithm stores all the paths encountered during the search. Below is an example path found by the algorithm:

$$\{InField(Size, P29, Ad001),\ Token(And, P29, Ad001),\ Next(P29, P30),$$
$$Token(Spacious, P30, Ad001)\ InField(Size, P30, Ad001)\}$$

[1]In this variant, a path doesn't need to connect arguments of the input atom. The only requirement is that any two consecutive atoms in a path must share at least one argument.

A standard way to generalize paths into first-order clauses is to replace each constant c_i in a conjunction with a variable v_i. However, for many tasks such as field segmentation, it is critical to have clauses that are specific to a particular constant. In order to create clauses with constants, we introduce mode declarations for creating clauses: $modec(p)$. This mode declaration has only one component which is an atom p whose arguments are either ‘c’ (constant) or ‘v’ (variable). Below are some examples of mode declarations for creating clauses:

$$modec(Token(c, v, v)) \quad modec(Next(v, v)) \quad modec(InField(c, v, v)$$

Based on these mode declarations, OSL variablises all constants in a conjunction except those are declared as constants. Then OSL converts the conjunction of positive literals to clausal form since this is the form used in Alchemy. In MLNs, a conjunction of positive literals with weight w is equivalent to a clause of negative literals with weight $-w$. Previous work (Kok & Domingos, 2009, 2010) found that it is also useful to add other variants of the clause by flipping the signs of some literals in the clause. Currently, we only add one variant—a Horn version of the clause by only flipping the first literal, the one for which the model made a wrong prediction. In summary, for each path, OSL creates two type of clauses: one with all negative literals and one in which only the first literal is positive. For example, from the sample path above, OSL creates the following two clauses:

$$\neg InField(Size, p1, a) \vee \neg Token(And, p1, a) \vee \neg Next(p1, p2) \vee$$
$$\neg Token(Spacious, p2, a) \vee \neg InField(Size, p2, a)$$

$$InField(Size, p1, a) \vee \neg Token(And, p1, a) \vee \neg Next(p1, p2) \vee$$
$$\neg Token(Spacious, p2, a) \vee \neg InField(Size, p2, a)$$

Finally, for each new clause c, OSL computes the difference in number of true groundings of c in the ground-truth possible world $(\mathbf{x}_t, \mathbf{y}_t)$ and the predicted possible world $(\mathbf{x}_t, \mathbf{y}_t^P)$, $\Delta n_c = n_c(\mathbf{x}_t, \mathbf{y}_t) - n_c(\mathbf{x}_t, \mathbf{y}_t^P)$. Then, only clauses whose difference in number of true groundings is greater than or equal to a predefined threshold $minCountDiff$ will be added to the existing MLN. The smaller the value of $minCountDiff$, the more clauses will be added to the existing MLN at each step.

6.3.2 Online Max-Margin l_1-regularized Weight Learning

The above online structure learner may introduce a lot of new clauses in each step, and some of them may not be useful in the long run. To address this issue, like in section 3.2.2, we use l_1-regularization but in an online setting. We employ a state-of-the-art online l_1-regularization method—ADAGRAD_FB which is a l_1-regularized adaptive subgradient method using composite mirror-descent update (Duchi, Hazan, & Singer, 2010). At each time step t, it updates the weight vector as follows:

$$\mathbf{w}_{t+1,i} = sign\left(\mathbf{w}_{t,i} - \frac{\eta}{H_{t,ii}}\mathbf{g}_{t,i}\right)\left[\left|\mathbf{w}_{t,i} - \frac{\eta}{H_{t,ii}}\mathbf{g}_{t,i}\right| - \frac{\lambda\eta}{H_{t,ii}}\right]_+ \quad (6.1)$$

Algorithm 6.1 OSL

```
Input:  $\mathcal{C}$: initial clause set (can be empty)
        $mode$: mode declaration for each predicate
        $maxLen$: maximum number of hyperedges in a path
        $minCountDiff$: minimum number of difference in true groundings for selecting new clauses
        $\lambda, \eta, \delta$: parameters for the $l_1$-regularization adaptive subgradient method
        $\rho(\mathbf{y}, \mathbf{y}')$: label loss function
Note:   Index H maps from each node $\gamma_i$ to set of hyperedges $r(\gamma_1, ..., \gamma_i, ..., \gamma_n)$ containing $\gamma_i$
        $Paths$ is a set of paths, each path is a set of hyperedges
1: Initialize: $\mathbf{w}_\mathcal{C} = 0, \mathbf{g}_\mathcal{C} = 0, nc = |\mathcal{C}|$
2: for $i = 1$ to $T$ do
3:     Receive an instance $\mathbf{x}_t$
4:     Predict $\mathbf{y}_t^P = \arg\max_{\mathbf{y}\in\mathbf{Y}} \langle \mathbf{w}_\mathcal{C}, \boldsymbol{n}_\mathcal{C}(\mathbf{x}_t, \mathbf{y})\rangle$
5:     Receive the correct target $\mathbf{y}_t$
6:     Compute $\Delta y_t = \mathbf{y}_t \setminus \mathbf{y}_t^P$
7:     if $\Delta y_t \neq \emptyset$ then
8:         $H = CreateHG((\mathbf{x}_t, \mathbf{y}_t), mode)$
9:         $Paths = \emptyset$
10:        for each true atom $r(c_1, ..., c_r) \in \Delta y_t$ do
11:            $V = \emptyset$
12:            for each $c_i \in \{c_1, ..., c_r\}$ do
13:                if isInputOrOutputVar($c_i$,mode) then
14:                    $V = V \cup \{c_i\}$
15:                end if
16:            end for
17:            $ModeGuidedFindPaths(\{r(c_1, ..., c_r)\}, V, H, mode, maxLen, Paths)$
18:        end for
19:    end if
20:    $\mathcal{C}_{new} = CreateClauses(\mathcal{C}, Paths, mode)$
21:    Compute $\Delta\boldsymbol{n}_\mathcal{C}, \Delta\boldsymbol{n}_{\mathcal{C}_{new}}$:
22:        $\Delta\boldsymbol{n}_\mathcal{C} = \boldsymbol{n}_\mathcal{C}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{n}_\mathcal{C}(\mathbf{x}_t, \mathbf{y}_t^P)$
23:        $\Delta\boldsymbol{n}_{\mathcal{C}_{new}} = \boldsymbol{n}_{\mathcal{C}_{new}}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{n}_{\mathcal{C}_{new}}(\mathbf{x}_t, \mathbf{y}_t^P)$
24:    for $i = 1$ to $|\mathcal{C}|$ do
25:        $\mathbf{g}_{\mathcal{C},i} = \mathbf{g}_{\mathcal{C},i} + \Delta\boldsymbol{n}_{\mathcal{C},\boldsymbol{i}} * \Delta\boldsymbol{n}_{\mathcal{C},\boldsymbol{i}}$
26:        $\mathbf{w}_{\mathcal{C},i} = sign\left(\mathbf{w}_{\mathcal{C},i} + \frac{\eta}{\delta+\sqrt{\mathbf{g}_{\mathcal{C},i}}}\Delta\boldsymbol{n}_{\mathcal{C},\boldsymbol{i}}\right)\left[\left|\mathbf{w}_{\mathcal{C},i} + \frac{\eta}{\delta+\sqrt{\mathbf{g}_{\mathcal{C},i}}}\Delta\boldsymbol{n}_{\mathcal{C},\boldsymbol{i}}\right| - \frac{\lambda\eta}{\delta+\sqrt{\mathbf{g}_{\mathcal{C},i}}}\right]_+$
27:    end for
28:    for $i = 1$ to $|\mathcal{C}_{new}|$ do
29:        if $\Delta\boldsymbol{n}_{\mathcal{C}_{new,i}} \geq minCountDiffer$ then
30:            $\mathcal{C} = \mathcal{C} \cup \mathcal{C}_{new,i}$
31:            $nc = nc + 1$
32:            $\mathbf{g}_{\mathcal{C},nc} = \Delta\boldsymbol{n}_{\mathcal{C}_{new,i}} * \Delta\boldsymbol{n}_{\mathcal{C}_{new,i}}$
33:            $\mathbf{w}_{\mathcal{C},nc} = \left[\frac{\eta}{\delta+\sqrt{\mathbf{g}_{\mathcal{C},nc}}}(\Delta\boldsymbol{n}_{\mathcal{C}_{new,i}} - \lambda)\right]_+$
34:        end if
35:    end for
36: end for
```

Algorithm 6.2 CreateHG(D,mode)

```
Input:  $D$: a relational example
        mode: mode declaration file
 1: for each constant $c$ in $D$ do
 2:    $H[c] = \emptyset$
 3: end for
 4: for each true ground atom $r(c_1, ..., c_r) \in D$ do
 5:    for each constant $c_i \in \{c_1, ..., c_r\}$ do
 6:       if isInputOrOutputVar($c_i$,mode) then
 7:          $H[c_i] = H[c_i] \cup \{r(c_1, ..., c_r)\}$
 8:       end if
 9:    end for
10: end for
11: return $H$
```

Algorithm 6.3 $ModeGuidedFindPaths(CurrPath, V, H, mode, maxLen, Paths)$

```
 1: if $|CurrPath| < maxLen$ then
 2:    for each constant $c \in V$ do
 3:       for each $r(c_1, ..., c_r) \in H[c]$ do
 4:          if $canBeAdded(r(c_1, ..., c_r), CurrPath, mode)$ == success then
 5:             if $CurrPath \notin Paths$ then
 6:                $CurrPath = CurrPath \cup \{r(c_1, ..., c_r)\}$
 7:                $Paths = Paths \cup \{CurrPath\}$
 8:                $V' = \emptyset$
 9:                for each $c_i \in \{c_1, ..., c_r\}$ do
10:                   if $c_i \notin V$ and isInputOrOutputVar($c_i$,mode) then
11:                      $V = V \cup \{c_i\}$
12:                      $V' = V' \cup \{c_i\}$
13:                   end if
14:                end for
15:                $ModeGuidedFindPaths(CurrPath, V, H, mode, maxLen, Paths)$
16:                $CurrPath = CurrPath \setminus \{r(c_1, ..., c_r)\}$
17:                $V = V \setminus V'$
18:             end if
19:          end if
20:       end for
21:    end for
22: end if
```

where λ is the regularization parameter, η is the learning rate, $\mathbf{g}_t$ is the subgradient of the loss function at step t, and $H_{t,ii} = \delta + ||\mathbf{g}_{1:t,i}||_2 = \delta + \sqrt{\sum_{j=1}^{t}(\mathbf{g}_{j,i})^2}$ ($\delta \geq 0$). Note that, ADAGRAD_FB assigns a different step size, $\frac{\eta}{H_{t,ii}}$, for each component of the weight vectors. Thus, besides the weights, ADAGRAD_FB also needs to retain the sum of the squared subgradients of each component.

From the equation 6.1, we can see that if a clause is not relevant to the current example (i.e. $g_{t,i} = 0$) then ADAGRAD_FB discounts its weight by $\frac{\lambda\eta}{H_{t,ii}}$. Thus, irrelevant clauses will be zeroed out in the long run.

Regarding the loss function, we use the prediction-based loss function l_{PL} described in section 5.2:

$$l_{PL}(\mathbf{w}_{\mathcal{C}}, (\mathbf{x}_t, \mathbf{y}_t)) = \left[\rho(\mathbf{y}_t, \mathbf{y}_t^P) - \left\langle \mathbf{w}_{\mathcal{C}}, \left(\mathbf{n}_{\mathcal{C}}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{n}_{\mathcal{C}}(\mathbf{x}_t, \mathbf{y}_t^P)\right)\right\rangle\right]_+$$

The subgradient of l_{PL} is:

$$\mathbf{g}_{PL} = \mathbf{n}_{\mathcal{C}}(\mathbf{x}_t, \mathbf{y}_t^{PL}) - \mathbf{n}_{\mathcal{C}}(\mathbf{x}_t, \mathbf{y}_t) = -\left[\mathbf{n}_{\mathcal{C}}(\mathbf{x}_t, \mathbf{y}_t) - \mathbf{n}_{\mathcal{C}}(\mathbf{x}_t, \mathbf{y}_t^{PL})\right] = -\Delta\mathbf{n}_{\mathcal{C}}$$

Substituting the gradient into equation 6.1, we obtain the following formulae for updating the weights of old clauses:

$$\mathbf{g}_{\mathcal{C},i} = \mathbf{g}_{\mathcal{C},i} + (\Delta \boldsymbol{n}_{\mathcal{C},i})^2$$

$$\mathbf{w}_{\mathcal{C},i} \leftarrow sign\left(\mathbf{w}_{\mathcal{C},i} + \frac{\eta}{\delta + \sqrt{\mathbf{g}_{\mathcal{C},i}}}\Delta \boldsymbol{n}_{\mathcal{C},i}\right)\left[\left|\mathbf{w}_{\mathcal{C},i} + \frac{\eta}{\delta + \sqrt{\mathbf{g}_{\mathcal{C},i}}}\Delta \boldsymbol{n}_{\mathcal{C},i}\right| - \frac{\lambda\eta}{\delta + \sqrt{\mathbf{g}_{\mathcal{C},i}}}\right]_+$$

For new clauses, the update formulae are simpler since all the previous weights and gradients are zero:

$$\mathbf{g}_{\mathcal{C},nc} = (\Delta \boldsymbol{n}_{\mathcal{C}_{new},i})^2$$

$$\mathbf{w}_{\mathcal{C},nc} = \left[\frac{\eta}{\delta + \sqrt{\mathbf{g}_{\mathcal{C},nc}}} (\Delta \boldsymbol{n}_{\mathcal{C}_{\boldsymbol{new},\boldsymbol{i}}} - \lambda) \right]_+$$

Lines $24-27$ in Algorithm 6.1 are the pseudocode for updating the weights of existing clauses, and lines $28-35$ are the pseudocode for selecting and setting weights for new clauses.

6.4 Experimental Evaluation

In this section, we conduct experiments to answer the following questions:

1. Starting from a given MLN, does OSL find new useful clauses that improve the predictive accuracy?
2. How well does OSL perform when starting from an empty knowledge base?
3. How does OSL compare to LSM, the state-of-the-art structure learner for MLNs (Kok & Domingos, 2010) ?

6.4.1 Data

We ran experiments on two real world datasets for field segmentation: CiteSeer described in section 4.4.1 and the advertisements dataset Craigslist (Grenager et al., 2005).

The Craigslist dataset[2] consists of advertisements for apartment rentals

[2] `http://nlp.stanford.edu/~grenager/data/unsupie.tgz`

posted on Craigslist. There are 8,767 ads in the dataset, but only 302 of them were labeled with 11 fields: *Available*, *Address*, *Contact*, *Features*, *Neighborhood*, *Photos*, *Rent*, *Restrictions*, *Roommates*, *Size*, and Utilities. The labeled ads are divided into 3 disjoint sets: training, development and test set. The number of ads in each set are 102, 100, and 100 respectively. We preprocessed the data using regular expressions to recognize numbers, dates, times, phone numbers, URLs, and email addresses.

6.4.2 Input MLNs

A standard model for sequence labeling tasks such as field segmentation is a linear chain CRF (Lafferty et al., 2001). Thus, we use a linear chain CRF as the input MLN. The following MLN, named LC_0, encodes a simple linear chain CRF that only use the current words as features:

$$Token(+t, p, c) \Rightarrow InField(+f, p, c)$$
$$Next(p1, p2) \wedge InField(+f1, p1, c) \Rightarrow InField(+f2, p2, c)$$
$$InField(f1, p, c) \wedge (f1 != f2) \Rightarrow \neg InField(f2, p, c).$$

The plus notation indicates that the MLN contains an instance of the first clause for each (*token, field*) pair, and an instance of the second clause for each pair of fields. Thus, the first set of rules captures the correlation between tokens and fields, and the second set of rules represents the transitions between fields. The third rule constrains that the token at a position p can be part of at most one field.

For CiteSeer, there is also an existing MLN previously mentioned in 4.4.1, which is called the isolated segmentation model (ISM). ISM is also a linear chain CRF but has more features than the simple linear chain CRF above. Like LC_0, ISM also has rules that correlate the current words with field labels. For transition rules, ISM only captures transitions within fields and also takes into account punctuation as field boundaries:

$$Next(p1, p2) \land \neg HasPunc(p1, c) \land InField(+f, p1, c) \Rightarrow InField(+f, p2, c)$$

In addition, ISM also contains rules that are specific to the citation domain such as "the first two positions of a citation are usually in the author field", "initials tend to appear in either the author or the venue field". Most of those rules are features corresponding to words that appear before or after the current tokens.

For Craigslist, previous work (Grenager et al., 2005) found that it is useful to only capture the transitions within fields and take into account the field boundaries, so we create a version of ISM for it by removing all clauses that are specific to the citation domain. Thus, the ISM MLN for Craiglist is a revised version of the LC_0 MLN. Therefore, we only ran experiments with ISM on Craigslist.

6.4.3 Methodology

To answer the questions above, we ran experiments with the following systems:

ADAGRAD_FB-LC_0: Use ADAGRAD_FB to learn weights for the LC_0 MLN.

OSL-M1-LC_0: Starting from the LC_0 MLN, this system runs a slow version of OSL where the parameter $minCountDiff$ is set to 1, i.e. all clauses whose number of true groundings in true possible worlds is greater than those in predicted possible worlds will be selected.

OSL-M2-LC_0: Starting from the LC_0 MLN, this system runs a faster version of OSL where the parameter $minCountDiff$ is set to 2.

ADAGRAD_FB-ISM: Use ADAGRAD_FB to learn weights for the ISM MLN.

OSL-M1-ISM: Like OSL-M1-LC_0, but starting from the ISM MLN.

OSL-M2-ISM: Like OSL-M2-LC_0, but starting from the ISM MLN.

OSL-M1-Empty: Like OSL-M1-LC_0, but starting from an empty MLN.

OSL-M2-Empty: Like OSL-M2-LC_0, but starting from an empty MLN.

Regarding label loss functions, we use Hamming (HM) loss described in section 4.3.2.

For inference in training and testing, we used the exact MPE inference method based on Integer Linear Programming described in section 4.3.1. For all systems, we ran one pass over the training set and used the average weight vector to predict on the test set. For Craigslist, we used the original split for training and test. For CiteSeer, we ran four-fold cross-validation (i.e. leave one topic out). The parameters λ, η, δ of ADAGRAD_FB were set to 0.001,1, and 1 respectively. For OSL, the mode declarations were set to constrain the search space of relational pathfinding to linear chain CRFs in order to make exact inference in training feasible [3]; the maximum path length *maxLen* was set to 4; the parameters λ, η, δ were set to the same values in ADAGRAD_FB. All the parameters are set based on the performance on the Craigslist development set. We used the same parameter values on CiteSeer.

Like previous work, we used F_1 to measure the performance of each system.

6.4.4 Results and Discussion

Table 6.1 shows the average F_1 with their standard deviations, average training times in minutes, and average number of non-zero clauses on CiteSeer. All results are averaged over the four folds. First, either starting from LC_0 or ISM, OSL is able to find new useful clauses that improve the F_1 scores. For LC_0, comparing to the system that only does weight learning, the fast version of OSL, OSL-M2, increases the average F_1 score by 9.4 points, from 82.62 to 92.05. The slow version of OSL, OSL-M1, further improves the average F_1 score to 94.47. For ISM, even though it is a well-developed MLN, OSL is

[3]We did try to search on a more complex space such as second-order CRFs, but it took much longer time in training with minimal improvement in the F_1 score.

Table 6.1: Experimental results on CiteSeer.

Systems	Average F_1	Average training time (minutes)	Average number of non-zero clauses
ADAGRAD_FB-LC_0	82.62 ± 2.12	10.40	$2,896$
OSL-M2-LC_0	92.05 ± 2.63	14.16	$2,150$
OSL-M1-LC_0	94.47 ± 2.04	163.17	$9,395$
ADAGRAD_FB-ISM	91.18 ± 3.82	11.20	$1,250$
OSL-M2-ISM	95.51 ± 2.07	12.93	$1,548$
OSL-M1-ISM	96.48 ± 1.72	148.98	$8,476$
OSL-M2-Empty	88.94 ± 3.96	23.18	650
OSL-M1-Empty	94.03 ± 2.62	257.26	$15,212$

Table 6.2: Experimental results on Craigslist.

Systems	F_1	Training time (minutes)	Number of non-zero clauses
ADAGRAD_FB-ISM	79.57	2.57	$2,447$
OSL-M2-ISM	77.26	3.88	$2,817$
OSL-M1-ISM	81.58	33.63	$9,575$
OSL-M2-Empty	55.28	17.64	$1,311$
OSL-M1-Empty	71.23	75.84	$17,430$

still able to enhance it. The OSL-M1-ISM achieves the best average F_1 score, 96.48, which is 2 points higher than the current best F_1 score achieved by using a complex joint segmentation model that also uses information from matching multiple citations of the same paper (Poon & Domingos, 2007). On the other hand, the results of OSL-M2-Empty and OSL-M1-Empty shows that OSL also perform very well when there is no input MLN. OSL-M1 even finds a structure that has higher predictive accuracy than that of ISM. All of the differences in F_1 score between OSL and ADAGRAD_FB are statistically significant according to a paired t-test at significance level 0.05. Regarding the training time, OSL-M2 takes on average a few more minutes than systems that only do weight learning. However, OSL-M1 takes more time to train since including more new clauses results in longer time for constructing the ground network, running inference, and computing the number of true groundings. The last column of Table 6.1 shows the average number of non-zero clauses in the final MLNs learnt by different systems. These numbers reflect the size of MLNs generated by different systems during training.

Table 6.2 shows the experimental results on Craigslist. The segmentation task in Craigslist is much harder than the one in CiteSeer due to the huge variance in the context of different ads. As a result, most words only appear once or twice in the training set. Thus the most important rules are those that correlate words with fields and those capturing the regularity that consecutive words are usually in the same field, which are already in ISM. In addition, most rules only appear once in a document. That's why OSL-M2 is not able to find useful clauses, but OSL-M1 is able to find some useful clauses that improve the F_1 score of ISM from 79.57 to 81.58. On the other hand, OSL also gives some promising results when starting from an empty MLN.

To answer question 3, we ran LSM on CiteSeer and Craigslist but the MLNs returned by LSM result in huge ground networks that make weight learning infeasible even using online weight learning. The problem is that these natural language problems have a huge vocabulary of words. Thus, failing to restrict clauses to specific words results in a blow-up in the size of the ground network. However, LSM is currently not able to learn clauses with constants. It is unclear whether it is feasible to make LSM efficiently learn clauses with constants since those constants may need to be considered individually which dramatically increases the search space. This problem also holds for other existing structure learners of MLNs (Kok & Domingos, 2005; Mihalkova & Mooney, 2007; Biba et al., 2008; Kok & Domingos, 2009). Our previous structure learner described in chapter 3 can learn clauses with constants but it can only learn non-recursive clauses. Thus, it is not suitable for the field segmentation task.

Below are some good clauses found by OSL-M2-ISM on CiteSeer:

- If the current token is in the *Title* field and it is followed by a period then it is likely that the next token is in the *Venue* field.

$$\neg InField(Ftitle, p1, c) \vee \neg FollowBy(p1, TPERIOD, c) \vee \\ \neg Next(p1, p2) \vee InField(Fvenue, p2, c)$$

- If the next token is 'in' and it is in the *Venue* field, then the current token is likely in the *Title* field

$$\neg Next(p1, p2) \vee \neg Token(Tin, p2, c) \vee \neg InField(Fvenue, p2, c) \vee \\ InField(Ftitle, p1, c)$$

On the other hand, when starting from an empty knowledge base, OSL-M2 is able to discover the regularity that consecutive words are usually in the same fields:

$$\neg Next(p1, p2) \vee \neg InField(Fauthor, p1, c) \vee InField(Fauthor, p2, c)$$
$$\neg Next(p1, p2) \vee \neg InField(Ftitle, p1, c) \vee InField(Ftitle, p2, c)$$
$$\neg Next(p1, p2) \vee \neg InField(Fvenue, p1, c) \vee InField(Fvenue, p2, c)$$

6.5 Related Work

Our work in this chapter is related to previous work on online feature selection for Markov Random Fields (MRFs) (Perkins & Theiler, 2003; Zhu, Lao, & Xing, 2010). However, our work differs in two aspects. First, this previous work assumes all the training examples are available at the beginning and only the features are arriving online, while in our work both the examples and features (clauses) are arriving online. Second, in previous work, the new features are given, while in our work the new features are induced from each example. Thus, our work is also related to previous work on feature induction for MRFs (Della Pietra, Della Pietra, & Lafferty, 1997; McCallum, 2003), but those are batch methods.

The idea of combining relational pathfinding with mode declarations has been used in previous work (Ong, de Castro Dutra, Page, & Costa, 2005; Duboc, Paes, & Zaverucha, 2008). However, how they are used is different. In (Ong et al., 2005), mode declarations were used to transform a bottom clause into a directed hypergraph where relational pathfinding was used to find paths. Similarly, in (Duboc et al., 2008), mode declarations were used to validate paths obtained from bottom clauses. Here, mode declarations are first used to reduce the search space to paths that contain 'input' and 'output' nodes. Then they are used to test whether an hyperedge can be added to an existing path. Finally, they are used to create clauses with constants.

6.6 Chapter Summary

In this chapter, we present OSL, the first online structure learner for MLNs. In each step, OSL uses mode-guided relational pathfinding to find new clauses that fix the model's wrong predictions. Experimental results in field segmentation on two real-world datasets show that OSL is able to find new useful clauses that improve the predictive accuracies of well-developed MLNs.

Chapter 7

Automatically Selecting Hard Constraints to Enforce when Training Structured Prediction

7.1 Introduction

Many real-world applications of machine learning involve a mix of soft probabilistic constraints and hard logical constraints. For example, when extracting relations from natural language sentences, the outputs must satisfy hard constraints like "the first argument of a *live_in* relation must be a *person* entity, and the second argument must be a *location* entity," as well as many soft constraints such as "the word 'residence' frequently appears between the two arguments of a *live_in* relation." Or when segmenting bibliographic citations, a prediction must conform to the hard constraint "a *Venue* token cannot appear before a *Title* token," as well as many soft constraints such as "the word 'International' is usually a *Venue* token."

In terms of graphical models, hard constraints add new interactions between variables, which increases the computational complexity of a problem. On the other hand, from the point of view of probabilistic models, hard constraints introduce deterministic factors into the model (i.e zero out some potential function values), which causes a lot of troubles to existing inference and learning methods (Poon & Domingos, 2006). Thus, finding the most effective and efficient way to perform inference and learning for problems with a mix of hard and soft constraints is an ongoing research problem (Chang, Ratinov, Rizzolo, & Roth, 2008). Previous work has explored two different approaches to the learning problem. The first approach, called *learning plus inference* (L+I) (Punyakanok, Roth, tau Yih, & Zimak, 2005), is to completely ignore hard constraints during training and only enforce them at testing time. At first, this approach does not seem theoretically appealing since hard constraints are only used during testing and have no effect on the learning process. However, the L+I approach allows efficient modular training of individual components that are only integrated as needed for testing and has achieved significant successes in many real-world applications (Roth & Yih, 2005, 2007). For example, Punyakanok et al. (2004) and Koomen et al. (2005) trained classifiers to independently assign a semantic role to each noun phrase in a natural language sentence, and then Integer Linear Programming is used to determine the most likely set of global assignments that satisfies a set of hard linguistic constraints. The second approach, called *inference based training (IBT)* (Punyakanok et al., 2005), includes all constraints both in training and testing. This approach is theoretically more desirable since it takes into account all constraints at training time thus can ideally learn a more accurate model. Nevertheless, so far there is relatively little empirical success on real-world problems with this approach since enforcing deterministic constraints during learning typically makes the training problem significantly more complex and computationally intractable.

In this chapter, we propose a new approach to incorporating declarative hard constraints when learning probabilistic models for structured prediction. The key idea is to only include "inexpensive" constraints during training and only enforce the remaining "expensive" constraints during testing. Our new approach, which we will call *Selectively Constrained Training* (SCT), lies somewhere between the two extreme approaches reviewed above, attempting to achieve the improved accuracy of the IBT approach while retaining the training efficiency of the L+I approach.

The remainder of the chapter is organized as follows. Section 7.1 describes our heuristic for selecting which hard constraints are "inexpensive" and should be included in training. Section 7.2 presents the experimental evaluation of the proposed approach. Section 7.3 discusses related work and section 7.4 summarizes the chapter.

7.2 Heuristic for selecting hard constraints to use in training

As previously mentioned, the main problem with including hard constraints during training is that, in practice, it greatly increases the computational complexity of the learning problem. Examining the problem further, we found that the main effect of enforcing hard constraints during training is on the complexity of the inference problem. Introducing hard constraints in training usually results in a much more complex inference problem which cannot be solved efficiently in most cases. Therefore, it significantly impacts the training process since we need to solve these complex inference problems many times during training. So our idea is to only include hard constraints in training when they do not significantly increase the complexity of the underlying inference problem. We call such hard constraints "inexpensive". A standard metric for measuring the complexity of an inference problem is the tree-width of the graphical structure (Koller & Friedman, 2009). However, computing the tree-width of a graph is an NP-hard problem in general (Arnborg, Corneil, & Proskurowski, 1987). There are methods for approximating the tree width (Koller & Friedman, 2009), but it is still computationally expensive for SRL formalisms since we need to construct the ground networks and compute the approximate tree-width for each possible subset of the hard constraints. We now describe a simple and efficient heuristic for detecting "inexpensive" hard constraints.

From the point of view of the resulting graphical model, each hard constraint defines a graphical structure among the output variables. So we define an "inexpensive" hard constraint in terms of how its addition to the graphical model affects the efficiency of inference. Since all graphical models can be converted into factor graphs, we analyze the complexity of a constraint based on its factor graph representation (Kschischang, Frey, & Loeliger, 2001). From the perspective of factor graphs, a hard constraint introduces new factors into the original problem. Intuitively, the denser the factor graph is, the harder the inference problem is since a dense factor graph tends to have high tree-width. For a given problem, the number of random variables, i.e. the number of nodes in its factor graph, is a fixed number, so the denseness of the factor graph is determined by the number of factors and the number of edges. Therefore, we measure the complexity of a hard constraint based on the number of factors added by the constraint and the degree (the number of edges connecting to nodes or the number of involved variables) of the created factors. A well-known result is that if the graph is a linear chain then inference is efficient (Sutton & McCallum, 2007). Looking at the factor graph of a linear chain, we see that it has n nodes (one for each output variable), $n-1$ factors (one for each pair of adjacent nodes), and each factor has a degree of two. So for linear chains, the relationship between the created factors and number of nodes is linear, $2*(n-1)$ and n. Based on this observation, we propose the following heuristic for detecting "inexpensive" hard constraints:

Definition. *An "inexpensive" hard constraint is one that creates a graphical structure in which the number of factors times the degree of each factor is linear in its number of nodes.*

For an MLN, this heuristic is easily implemented. The number of nodes created by a clause is the total number of unique ground literals of its query predicates. The number of factors that a clause creates is its number of unique ground clauses. The degree of each created factor is the number of appearances of query predicates in the clause. Therefore, based on examining these quantities, it can be automatically decided whether or not a hard clause is "inexpensive." Below are examples of "inexpensive" and "expensive" hard constraints in MLNs for the task of citation segmentation:

- **An "inexpensive" constraint**: a *Venue* token cannot appear right after an *Author* token

  $$Next(p1, p2) \wedge InField(Fauthor, p1, c) \Rightarrow \neg InField(Fvenue, p2, c).$$

  This constraint satisfies the above criterion since the number of generated ground clauses is $n-1$, there are two appearances of the query predicate *InField* in each ground clause, and the total number of unique ground literals is $2n$ where n is the number of possible positions (i.e. tokens) in the citation to be segmented.

- **An "expensive" constraint**: a *Venue* token cannot appear before an *Author* token

  $$LessThan(p1, p2) \wedge InField(Fauthor, p2, c) \Rightarrow \\ \neg InField(Fvenue, p1, c).$$

  This constraint does not satisfy the above criterion since the relationship between the number of generated ground clauses and the total number of unique ground literals is quadratic ($n * (n-1)/2$ ground clauses and $2n$ ground literals.)

In the SCT approach, only the inexpensive constraints are used when training the weights of the soft clauses in the model. This ensures that inference, and therefore training, is reasonably efficient. During testing, all hard clauses and soft clauses with learned weights are used when making predictions. This helps ensures accurate predictions for test cases.

7.3 Experimental Evaluation

7.3.1 Data

In order to empirically evaluate SCT and compare it to L+I and IBT, we ran experiments on two standard bibliographic citation datasets: CiteSeer described in 4.4.1 and Cora (Bilenko & Mooney, 2003).

The task is to segment each citation into three fields: *Author*, *Title* and *Venue*. There are 1, 563 and 1, 295 citations in CiteSeer and Cora respectively. The CiteSeer dataset has four independent subsets consisting of citations in four different research areas. There are 3 disjoint subsets of citations in Cora.

7.3.2 Hard Constraints

We used the MLN for isolated segmentation model described in 6.4.2 as the base MLN. It has one hard clause imposing the constraint that a token can only be part of at most one field (mutual exclusivity):

$$InField(f1, p, c) \wedge (f1! = f2) \Rightarrow \neg InField(f2, p, c).$$

When analyzing the prediction errors of the isolated segmentation model including this constraint, we noticed these types of recurring errors:

- Violations of the purity of each field. For example, we found *Title* or *Venue* tokens between two *Author* tokens in many examples.
- Violations of the typical order of field appearances such as the *Venue* field appearing between the *Author* and *Title* field.

To correct these errors, we introduced the following new hard constraints:

- Continuity constraint (C1): any tokens between two tokens of the same field must also belong to that field.

$$LessThan(p1, p2) \wedge LessThan(p2, p3) \wedge InField(+f, p1, c) \wedge$$
$$InField(+f, p3, c) \Rightarrow InField(+f, p2, c).$$

- Constraints on the order of field appearance.
  - C2: a *Title* token cannot appear before an *Author* token

$$LessThan(p1, p2) \wedge InField(Fauthor, p2, c) \Rightarrow$$
$$\neg InField(Ftitle, p1, c).$$

  - C3: a *Venue* token cannot appear before an *Author* token

$$LessThan(p1, p2) \wedge InField(Fauthor, p2, c) \Rightarrow$$
$$\neg InField(Fvenue, p1, c).$$

  - C4: a *Venue* token cannot appear before a *Title* token

$$LessThan(p1, p2) \wedge InField(Ftitle, p2, c) \Rightarrow$$
$$\neg InField(Fvenue, p1, c).$$

– C5: a *Venue* token cannot appear right after an *Author* token

$$Next(p1, p2) \wedge InField(Fauthor, p1, c) \Rightarrow$$
$$\neg InField(Fvenue, p2, c).$$

According to the heuristic described in previous section, only the mutually exclusive constraint and constraint C5 are "inexpensive" constraints, the rest are "expensive" ones.

7.3.3 Methodology

We evaluated the following systems:

No constraints : Train the base MLN, which is the isolated segmentation model without the mutual exclusivity constraint, and use the learned MLN for testing.

L+I : Like the previous system, but include all of the hard constraints above during testing.

IBT : Train a model that includes clauses from the base MLN as well as all of the hard constraints above, and then use the learned MLN with all the constraints for testing.

IBT-Approx : Like IBT but use approximate inference instead of exact inference in training.

SCT : Use the heuristic described in previous section to automatically select the "inexpensive" hard constraints from the hard constraints above and only include them when training, and then include all constraints during testing.

SCT$-$: Like SCT but does not include "expensive" constraints in testing. In other words, this is IBT without "expensive" constraints.

To train all of the systems, we used the CDA-ML algorithm described in section 5.2. We used Hamming loss as the label loss function and set the value of σ to 0.0001 for CiteSeer and 0.002 for Cora.[1] We ran one pass over the training set and used the average weight vector for making predictions on the test set. For inference, we used the procedure described in section 4.3.1 to translate the MPE inference problem into an ILP. An ILP solver was then used for exact inference, and the ILP was relaxed to a Linear Program (LP) and an LP solver used for approximate inference.[2] Only exact inference guarantees that inference results satisfy all hard constraints. For all systems, we ran 4-fold cross-validation on CiteSeeer and 3-fold cross-validation on Cora.

Like previous work, we used the F_1 at the token level, F_1-token, to measure the segmentation accuracy of all systems. However, the F_1-token only captures the local performance, which is not the best metric for measuring the effect of hard constraints which enforce global properties of the complete segmentation. To better measure the effect of hard constraints, we also computed the F_1 at the field level, F_1-field, (all tokens in a field must be

[1]The value of σ was set based on predictive performance on the training set.

[2]We used lp_solve (`http://lpsolve.sourceforge.net`) for ILP and Mosek (`http://www.mosek.com`) for LP.

Table 7.1: Performance of different systems on CiteSeer. Results are averaged over 4 folds. Results of the proposed approach are shown in bold.

	F_1-token	F_1-field	Citation accuracy	Average trainining time (minutes)
No constraints	93.53 ± 2.63	84.41 ± 8.63	37.25 ± 12.50	12.1
L+I	95.10 ± 2.51	85.85 ± 9.61	63.30 ± 22.59	12.1
IBT	95.53 ± 2.04	88.00 ± 6.75	66.83 ± 17.95	322.8
IBT-Approx	90.37 ± 3.73	69.48 ± 15.55	20.45 ± 15.04	64.4
SCT	**95.64 ± 2.05**	**87.97 ± 6.85**	**66.72 ± 18.72**	**37.4**
SCT−	94.61 ± 1.95	84.70 ± 7.91	43.24 ± 10.37	37.4

Table 7.2: Performance of different systems on Cora. Results are averaged over 3 folds. Results of the proposed approach are shown in bold.

	F_1-token	F_1-field	Citation accuracy	Average training time (minutes)
No constraints	96.46 ± 0.45	92.58 ± 3.08	55.91 ± 3.96	2.6
L+I	98.88 ± 0.21	93.74 ± 0.86	81.30 ± 1.86	2.6
IBT	98.92 ± 0.40	95.63 ± 1.74	84.45 ± 7.50	266.3
IBT-Approx	87.06 ± 1.98	62.12 ± 4.69	13.61 ± 5.15	87.2
SCT	**98.98 ± 0.49**	**96.04 ± 1.83**	**85.39 ± 7.08**	**22**
SCT−	98.10 ± 0.90	94.73 ± 2.08	71.40 ± 9.51	22

correctly assigned the right labels in order to count as a correct field) and *citation accuracy*: the proportion of citations in which *all* tokens are assigned the correct labels, i.e. the percentage of citations segmented completely correctly.

7.3.4 Results and Discussion

Table 7.1 and Table 7.2 show the performance of difference systems on CiteSeer and Cora respectively. The first three columns show how adding hard constraints improves segmentation accuracy. All of the systems with constraints except IBT-Approx are significantly better than the one without them, especially in terms of citation accuracy. For example, the citation accuracies of IBT and SCT are 30% higher than that of “No constraints” on average. The reason for the huge jump in the citation accuracy is due to the fact that without constraints there are many examples having only a few mislabeled tokens. Among the systems using constraints, the ones that are *trained* using constraints except IBT-Approx have higher segmentation accuracy than the one that only enforces constraints during testing (L+I). On the other hand, the significant difference between the citation accuracy of SCT− and that of SCT shows the usefulness of the expensive hard constraints (C1 to C4).

Therefore, including hard constraints during learning improves the accuracy of the learned model with exact inference. However, the last column of Table 7.1 and 7.2 show that naively including all hard constraints in training results in a huge increase in training time. Training for the IBT approach takes 26.7 times longer than the L+I approach on CiteSeer, and an order of magnitude longer on Cora. IBT-Approx results show that using approximate inference in training reduces training time but also significantly decreases accuracy. However, using the heuristic described in section 7.2, SCT takes only 3 times longer to train than L+I and is 8.6 times faster than IBT on CiteSeer, and 8.5 times longer than L+I and 12 times faster than IBT on Cora, while still matching IBT’s accuracy.

IBT-Approx has the worst accuracy, which may be due to the combination of online learning and approximate inference, since the same approximate inference method has shown good performance in chapter 4. The performance of IBT-Approx may be improved by using the latest methods for learning with approximate inference (Meshi, Sontag, Jaakkola, & Globerson, 2010; Martins, Smith, Xing, Aguiar, & Figueiredo, 2010; Koo, Rush, Collins, Jaakkola, & Sontag, 2010).

7.4 Related Work

Learning and inference with constraints has been studied in several previous papers (Punyakanok et al., 2005; Tromble & Eisner, 2006; Chang et al., 2008). The first comprehensive study of the issue of learning and inference over constrained output was conducted by Punyakanok et al. (2005). The L+I and IBT approaches were defined in that work, and experiments on both synthetic and real-world data were conducted to compare their predictive performance. The authors also presented some theoretical results on the predictive performance of the L+I and IBT approaches. However, they did not look at the case of only enforcing *some* constraints during training. In followup work, Chang et al. (2008) proposed Constrained Conditional Models (CCMs), an extension of linear models for combining probabilistic models with declarative constraints. A CCM has two weight vectors: one for features and one for constraints. Unlike MLNs, CCMs separate constraints from features, thus their weights are learned in different manners. When all the constraints are hard constraints, then a CCM can be represented by an MLN where the soft clauses in the MLN are the features and the hard clauses are the constraints in the corresponding CCM. Like Punyakanok et al. (2005), the authors only looked at two ways to train a CCM: the L+I and IBT approaches.

By only enforcing “inexpensive” constraints in training, our work shares the goal of piecewise training (Sutton & McCallum, 2009), which is “to perform less inference at training time than at test time.” The key idea of piecewise training is to break the original complex problem into smaller pieces, train these pieces independently, and then combine them at testing for prediction. In this sense, piecewise training is similar to an extreme version of the L+I approach (Chang et al., 2008) where no interaction between output variables is learned during training and constraints are only used in testing to capture those relationships. So far, piecewise training has only been applied in training structured prediction models without hard constraints. It would be interesting to see how well it performs when there are hard constraints in the models.

Another line of research on speeding up the learning process for complex problems is learning with approximate inference (Martins, Smith, & Xing, 2009; Meshi et al., 2010; Martins et al., 2010; Koo et al., 2010). It would be interesting to see how well these methods help improve the accuracy of IBT-Approx.

7.5 Chapter Summary

We have presented a new approach to incorporating declarative hard constraints into learning models for structured prediction. The idea is to only enforce “inexpensive” hard constraints during training that do not inordinately increase the computational complexity of inference. We also proposed a simple heuristic for detecting “inexpensive” hard constraints in Statistical Relational Learning frameworks like Markov Logic that represent models as templates for constructing graphical models. The proposed approach was applied to MLNs on the task of bibliographic citation segmentation. Experimental results show that the new approach achieves the best predictive accuracy while still allowing for efficient training.

Chapter 8

Future Work

This chapter discusses some ways in which the contributions of this thesis can be extended.

8.1 Online Max-Margin Weight Learning

As mentioned earlier, the CDA algorithm developed in chapter 5 can be applied to other structured prediction models. So it would be interesting to apply CDA to models such as M3Ns (Taskar et al., 2004). On the other hand, currently CDA assumes that the data are fully observable. However, there are problems in which some information is not observable. For example, in plan recognition, we usually only observe the actions and the top-level plans not the intermediate plans (Blaylock & Allen, 2005). Existing work has developed a max-margin approach for learing with partially observable data (Yu & Joachims, 2009), but the method is for a batch setting. So it would be interesting to extend CDA to the case of learing with partially observable data. Another venue for future work is to derive Coordinate-Dual-Ascent algorithms for l_1-regularized max-margin structured prediction.

8.2 Online Structure Learning

In chapter 6, we presented OSL, the first online structure learning for MLNs. Since this is the initial step, there is still a lot of room for improvement. First, the mode declarations provide a simple way to restrict the search space for paths but there are some types of constraints that cannot be expressed by mode declarations such as the constraint that the predicate P1 and predicate P2 should not appear in the same path. So it may be useful to use some other forms of language biases that are more expressive. In addition, OSL currently does not use clauses in the existing MLN to restrict the search space. So it would be useful to exploit this information. Second, OSL, especially OSL-M1, currently adds a lot of new clauses at each step, which significantly increases the computational cost. So it would be useful to develop a better criterion for selecting fewer but useful clauses at each step. On the other hand, besides l_1-regularization, there are also other methods for inducing sparse models such as greedy methods (Zhang, 2009). Thus, it would be interesting to explore those methods in order to reduce the number of clauses learnt by OSL. Finally, relational pathfinding is just one way to learn clauses from data, and there are some types of clauses that cannot be learnt by relational pathfinding such as clauses containing non-relational literals. Hence, it may be useful to combine relational pathfinding with other search methods.

8.3 Other Issues in Online Learning

In this thesis, we only considered one extreme setting in online learning where the learner only uses the current example for updating the model at each step and only runs one pass over examples. So it would be interesting to explore other settings such as running multiple passes (or epochs) over examples or maintaining a window of examples. It would also be useful to study the sensitivity of the online algorithms developed in chapter 5 and 6 to the order of examples.

8.4 Discriminative Learning with Large Mega-Examples

In this thesis, we addressed one dimension of the issue of scalability in discriminative learning for MLNs when the number of examples is increasing. However, the issue of scalability also arises when the size of an example is getting bigger. For instance, in social network analysis (Backstrom & Leskovec, 2011), each example is a huge network. In order to use the max-margin weight learning methods developed in chapters 4 and 5 on these large mega-examples, one needs to develop efficient MPE inference methods for large graphs. For instance, one can use the approach described by Singla and Domingos (2008) to lift the max-product algorithm (Pearl, 1988), a widely used approximation algorithm for MPE inference in Markov network. For discriminative structure learning on large mega-examples, one can adapt the motif identification step in LSM to only search for motifs that contain the query predicates, then uses the mode-guided relational pathfinding developed in chapter 6 to search for paths in each motif. In addition, another important line of research is to develop online learning methods for the case where the example is a single huge network that is changing over time.

8.5 Improving Scalability through Parallelism

In this thesis, we improve the scalability of discriminative learning methods for MLNs through online learning. However, another way to speed up existing discriminative learning methods for MLNs is to make them parallel. For instance, one may use GraphLab (Low, Gonzalez, Kyrola, Bickson, Guestrin, & Hellerstein, 2010), a recently proposed parallel framework for machine learning algorithms, to speed up existing batch learning methods for MLNs. In addition, the online algorithms presented in chapter 5 and 6 can also be sped up by computing the subgradients and performing the weight's updates in parallel.

8.6 Learning with Hard Constraints

In the previous chapter, we only experimentally evaluated the SCT approach with one learning algorithm. So, it would be useful to test the SCT approach with other learning algorithms. Additionally, given the encouraging results on citation segmentation, an obvious area for future research is evaluating the selective enforcement of constraints when training models for other real-world applications that involve a lot of hard constraints such as entity and relation extraction. On the other hand, SCT could be applied to select inexpensive constraints in other SRL formalisms that use expressions in logical, relational-database, or object-oriented languages as templates for constructing graphical models, such as BLPs (Kersting & De Raedt, 2001), PRMs (Getoor, Friedman, Koller, & Pfeffer, 2001), RMNs (Taskar et al., 2002) and FACTORIE (McCallum, Schultz, & Singh, 2009). Given the success with MLNs, it would be interesting to see how well SCT perform on those formalisms.

Currently, SCT only addresses the computational aspect of hard constraints. However, another important aspect of hard constraints is whether they are helpful or not. So it would be useful to extend SCT to take into account the usefulness of hard constraints.

8.7 Other Applications

In previous chapters, we have applied our new algorithms to some real-world structured prediction problems that involve data with thousands of examples such as natural language field segmentation, semantic role labeling, and web search. There are many more real-world problems that have similar characteristics. For examples, many problems in computer vision involve data with thousands of images where each image is an example. The task may be to segment an image into different regions or to recognize all the objects and their interactions in a given image, etc. Thus, it would be interesting to apply the methods developed in this thesis to those problems.

In addition, the online learning algorithms developed in chapter 5 and 6 fit nicely to many problems in social media where the data arrive in a streaming order. So it would be interesting to apply those methods to problems with streaming data.

Chapter 9

Conclusion

The research presented in this thesis addresses two important issues in discriminative learning for MLNs: accuracy and scalability.

We first presented a new method that discriminatively learns both the structure and parameters for a special class of MLNs where all the clauses are non-recursive ones which allow efficient exact inference. The proposed approach is a two-step process. The first step uses ALEPH to generate a large set of potential clauses. The second step learns the weights for these clauses, preferring to eliminate useless clauses by giving them zero weight by using l_1-regularization. The new method outperforms existing MLN and ILP methods and achieves state-of-the-art accuracies on the Alzheimer's-drug benchmarks.

To further improve the predictive accuracy, we proposed a new approach to learning weights for MLNs, which aims to maximize the separation margin instead of the conditional likelihood of the training data. In order to solve the max-margin optimization problem, we developed a new approximate algorithm for loss-augmented MPE inference in MLN based on LP-relaxation. The max-margin weight learner generally has better or equally good but more stable predictive accuracy than existing discriminative weight learning methods for MLNs.

Then, to make the max-margin method more scalable, we derived CDA, an online algorithm for max-margin structured prediction, from the primal-dual framework. We applied CDA to learn weights for MLNs on problems with thousand of examples where existing batch learning methods for MLNs cannot. Experimental results on several large-scale real-world problems show that CDA generally achieves better accuracy than existing online methods for structured prediction. In particular, CDA is more accurate and robust on noisy datasets.

However, like other existing online algorithms, CDA assumes the input MLN's structure is complete and only updates the weights. But, the input structure is usually incomplete in practice, so it should be also updated. To address this issue, we developed OSL, the first algorithm that performs both online structure and parameter learning. To find new clauses at each step, we introduced mode-guided relational pathfinding which use mode declarations to constrain the search of relational pathfinding in a novel way. Experimental results in field segmentation on two real-world datasets show that OSL is able to find new useful clauses that improve the predictive accuracies of well-developed MLNs.

In the final part of the thesis, we addressed the problem of learning with a mix of hard and soft constraints which arises in many real-world problems. Based on first-order logic, MLNs provides a convenient way to encode both soft and hard constraints. However, the training problem becomes more computational expensive due to the complexity introduced by the hard constraints. To address this issue, we proposed SCT, a simple heuristic for automatically selecting which hard constraints should be included during training. On the task of bibliographic citation segmentation, SCT achieves better accuracy than existing methods for learning with hard constraints, while still allows efficient training.

Overall, the work in this thesis have led to progress on discriminative learning for MLNs. Since many real-world problem are discriminative and involve noisy structured data with a lot of examples, our work have provided more accurate and scalable methods for solving those problems.

Bibliography

Andrew, G., & Gao, J. (2007). Scalable training of L_1-regularized log-linear models. In Ghahramani, Z. (Ed.), *Proceedings of 24th International Conference on Machine Learning (ICML-2007)*, pp. 33–40, Corvallis, OR.

Anguelov, D., Taskar, B., Chatalbashev, V., Koller, D., Gupta, D., Heitz, G., & Ng, A. (2005). Discriminative learning of Markov random fields for segmentation of 3D scan data. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pp. 169–176.

Arnborg, S., Corneil, D. G., & Proskurowski, A. (1987). Complexity of finding embeddings in a k-tree. *SIAM J. Algebraic Discrete Methods*, *8*, 277–284.

Asano, T. (2006). An improved analysis of Goemans and Williamson's LP-relaxation for MAX SAT. *Theoretical Computer Science*, *354*(3), 339–353.

Asano, T., & Williamson, D. P. (2002). Improved approximation algorithms for MAX SAT. *Journal of Algorithms*, *42*(1), 173–202.

Backstrom, L., & Leskovec, J. (2011). Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the 4th International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12*, pp. 635–644.

Bakir, G., Hoffman, T., Schölkopf, B., Smola, A. J., Taskar, B., & Vishwanathan, S. V. N. (Eds.). (2007). *Predicting Structured Data*. MIT Press, Cambridge, MA.

Bartlett, P. L., Collins, M., Taskar, B., & McAllester, D. A. (2005). Exponentiated gradient algorithms for large-margin structured classification. In *Advances in Neural Information Processing Systems 17, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada.*

Biba, M., Ferilli, S., & Esposito, F. (2008). Discriminative structure learning of Markov logic networks. In *Proceedings of the 18th international conference on Inductive Logic Programming (ILP'08)*, pp. 59–76, Prague, Czech Republic. Springer-Verlag.

Bilenko, M., & Mooney, R. J. (2003). Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pp. 39–48, Washington, DC.

Blaylock, N., & Allen, J. (2005). Generating artificial corpora for plan recognition. In *International Conference on User Modeling (UM-05)*. Springer.

Boros, E., & Hammer, P. L. (2002). Pseudo-Boolean optimization. *Discrete Applied Mathematics*, *123*(1-3), 155–225.

Boyd, S., & Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bradley, P. S., & Mangasarian, O. L. (1998). Feature selection via concave minimization and support vector machines. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pp. 82–90, Madison, Wisconsin, USA. Morgan Kaufmann Publishers Inc.

Carreras, X., & Màrquez, L. (2005). Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pp. 152–164, Ann Arbor, MI.

Chang, M.-W., Ratinov, L.-A., Rizzolo, N., & Roth, D. (2008). Learning and inference with constraints. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*, pp. 1513–1518.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, Philadelphia, PA.

Collins, M. (2004). Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In Harry Bunt, J. C., & Satta, G. (Eds.), *New Developments in Parsing Technology*. Kluwer.

Collins, M., Globerson, A., Koo, T., Carreras, X., & Bartlett, P. L. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *Journal of Machine Learning Research*, *9*, 1775–1822.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, *7*, 551–585.

Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Culotta, A. (2008). *Learning and inference in weighted logic with application to natural language processing*. Ph.D. thesis, University of Massachusetts.

Cussens, J. (2007). Logic-based formalisms for statistical relational learning.. In Getoor, L., & Taskar, B. (Eds.), *Introduction to Statistical Relational Learning*, pp. 269–290. MIT Press, Cambridge, MA.

Davis, J., Burnside, E. S., de Castro Dutra, I., Page, D., & Costa, V. S. (2005). An integrated approach to learning Bayesian networks of rules. In *Proceedings of the 16th European Conference on Machine Learning (ECML-05)*, pp. 84–95.

Davis, J., & Goadrich, M. (2006). The relationship between precision-recall and ROC curves. In *Proceedings of 23rd International Conference on Machine Learning (ICML-2006)*, pp. 233–240.

Dehaspe, L. (1997). Maximum entropy modeling with clausal constraints. In Džeroski, S., & Lavrač, N. (Eds.), *Proceedings of the 7th International Workshop on Inductive Logic Programming*, pp. 109–124.

Della Pietra, S., Della Pietra, V. J., & Lafferty, J. D. (1997). Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(4), 380–393.

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, *10*(7), 1895–1923.

Domingos, P., & Lowd, D. (2009). *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Duboc, A. L., Paes, A., & Zaverucha, G. (2008). Using the bottom clause and mode declarations on FOL theory revision from examples. In *Proceedings of the 18th International Conference on Inductive Logic Programming (ILP-2008)*, pp. 91–106.

Duchi, J., Hazan, E., & Singer, Y. (2010). Adaptive subgradient methods for online learning and stochastic optimization. Tech. rep. UCB/EECS-2010-24, EECS Department, University of California, Berkeley.

Dudík, M., Phillips, S. J., & Schapire, R. E. (2007). Maximum entropy density estimation with generalized regularization and an application to species distribution modeling. *Journal of Machine Learning Research*, *8*, 1217–1260.

Džeroski, S. (1991). Handling noise in inductive logic programming. Master's thesis, Faculty of Electrical Engineering and Computer Science, University of Ljubljana.

Dzeroski, S. (2007). Inductive logic programming in a nutshell.. In Getoor, L., & Taskar, B. (Eds.), *Introduction to Statistical Relational Learning*, pp. 57–92. MIT Press, Cambridge, MA.

Finley, T., & Joachims, T. (2008). Training structural SVMs when exact inference is intractable. In *Proceedings of 25th International Conference on Machine Learning (ICML-2008)*, pp. 304–311, Helsinki,Finland.

Fung, G. M., & Mangasarian, O. L. (2004). A feature selection Newton method for support vector machine classification. *Computational Optimization and Applications*, *28*(2), 185–202.

Genesereth, M. R., & Nilsson, N. J. (1987). *Logical foundations of artificial intelligence*. Morgan Kaufmann.

Getoor, L., & Taskar, B. (Eds.). (2007). *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA.

Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. (2001). Learning probabilistic relational models. In Džeroski, S., & Lavrač, N. (Eds.), *Relational Data Mining*.

Grenager, T., Klein, D., & Manning, C. D. (2005). Unsupervised learning of field segmentation models for information extraction. In *Proceedings of* *the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-05)*.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*(8), 1771–1800.

Huynh, T. N., & Mooney, R. J. (2009). Max-margin weight learning for Markov logic networks. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-09)*, pp. 564–579, Bled, Slovenia.

Huynh, T. N., & Mooney, R. J. (2011). Online max-margin weight learning with Markov Logic Networks. In *Proceedings of the 2011 SIAM International Conference on Data Mining (SDM-2011)*, Meza, Arizona, USA.

Huynh, T. N., & Mooney, R. J. (2008). Discriminative structure and parameter learning for Markov logic networks.. pp. 416–423, Helsinki, Finland.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *Proceedings of 22nd International Conference on Machine Learning (ICML-2005)*, pp. 377–384.

Joachims, T., Finley, T., & Yu, C.-N. (2009). Cutting-plane training of structural SVMs. *Machine Learning*. `http://www.springerlink.com/content/h557723w88185170`.

Kakade, S. M., & Shalev-Shwartz, S. (2009). Mind the duality gap: Logarithmic regret algorithms for online optimization. In Koller, D., Schuurmans, D., Bengio, Y., & Bottou, L. (Eds.), *Advances in Neural Information*

Processing Systems 21, Vancouver, British Columbia, Canada, December 8-11, 2008, pp. 1457–1464. MIT Press.

Kautz, H., Selman, B., & Jiang, Y. (1997). A general stochastic approach to solving problems with hard and soft constraints. In Dingzhu Gu, J. D., & Pardalos, P. (Eds.), *The Satisfiability Problem: Theory and Applications*, pp. 573–586. American Mathematical Society.

Kersting, K., & De Raedt, L. (2001). Towards combining inductive logic programming with Bayesian networks. In *Proceedings of the 11th International Conference on Inductive Logic Programming (ILP-2001)*, pp. 118–131, Strasbourg, France.

King, R. D., Sternberg, M. J. E., & Srinivasan, A. (1995). Relating chemical activity to structure: An examination of ILP successes. *New Generation Computing*, *13*(3,4), 411–433.

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. In *Proceedings of 22nd International Conference on Machine Learning (ICML-2005)*, Bonn,Germany.

Kok, S., & Domingos, P. (2008). Extracting semantic networks from text via relational clustering. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML-PKDD 2008), Antwerp, Belgium*, pp. 624–639. Springer.

Kok, S., & Domingos, P. (2009). Learning Markov logic network structure via hypergraph lifting.. pp. 505–512, Montreal, Quebec, Canada.

Kok, S., & Domingos, P. (2010). Learning Markov logic networks using structural motifs. In Fürnkranz, J., & Joachims, T. (Eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 551–558, Haifa, Israel.

Kok, S., Singla, P., Richardson, M., & Domingos, P. (2005). The Alchemy system for statistical relational AI. Tech. rep., Department of Computer Science and Engineering, University of Washington. `http://www.cs.washington.edu/ai/alchemy`.

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Koller, D., & Pfeffer, A. (1998). Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pp. 580–587, Madison, WI. AAAI Press / The MIT Press.

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., & Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2010)*, pp. 1288–1298, Morristown, NJ, USA.

Koomen, P., Punyakanok, V., Roth, D., & Yih, W. (2005). Generalized inference with multiple semantic role labeling systems. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pp. 181–184, Ann Arbor, MI.

Kschischang, F. R., Frey, B., & Loeliger, H.-A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*(2), 498–519.

Kumar, M. P., Kolmogorov, V., & Torr, P. H. S. (2009). An analysis of convex relaxations for MAP estimation of discrete MRFs. *Journal of Machine Learning Research*, *10*(Jan), 71–106.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of 18th International Conference on Machine Learning (ICML-2001)*, pp. 282–289, Williamstown, MA.

Landwehr, N., Kersting, K., & Raedt, L. D. (2007). Integrating Naive Bayes and FOIL. *Journal of Machine Learning Research*, *8*, 481–507.

Landwehr, N., Passerini, A., Raedt, L. D., & Frasconi, P. (2006). kFOIL: Learning simple relational kernels. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*.

Lawrence, S., Giles, C. L., & Bollacker, K. D. (1999). Autonomous citation matching. In *Proceedings of the Third Annual Conference on Autonomous Agents*.

Lee, S., Ganapathi, V., & Koller, D. (2007). Efficient structure learning of Markov networks using L_1-regularization. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pp. 817–824.

Liu, D. C., & Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematic Programming*, *45*(3), 503–528.

Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., & Hellerstein, J. M. (2010). GraphLab: A new parallel framework for machine learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI-2010)*, Catalina Island, California.

Lowd, D., & Domingos, P. (2007). Efficient weight learning for Markov logic networks. In *Proceedings of 7th European Conference of Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD-2007)*, pp. 200–211.

Martins, A., Smith, N., & Xing, E. (2009). Polyhedral outer approximations with application to natural language parsing.. pp. 713–720, Montreal. Omnipress.

Martins, A. F. T., Smith, N. A., Xing, E. P., Aguiar, P. M. Q., & Figueiredo, M. A. T. (2010). Turbo parsers: dependency parsing by approximate variational inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2010)*, pp. 34–44, Morristown, NJ, USA.

McCallum, A. (2003). Efficiently inducing features of conditional random fields. In *Proceedings of 19th Conference on Uncertainty in Artificial Intelligence (UAI-2003)*, pp. 403–410, Acapulco, Mexico.

McCallum, A., Schultz, K., & Singh, S. (2009). FACTORIE: Probabilistic Programming via Imperatively Defined Factor Graphs. In *Advances in Neural Information Processing Systems 22 (NIPS-2009)*, pp. 1249–1257.

McDonald, R., Crammer, K., & Pereira, F. (2005). Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL-05)*, pp. 91–98, Morristown, NJ, USA. Association for Computational Linguistics.

Meshi, O., Sontag, D., Jaakkola, T., & Globerson, A. (2010). Learning efficiently with approximate inference via dual losses.. pp. 783–790, Haifa, Israel. Omnipress.

Mihalkova, L., Huynh, T., & Mooney, R. J. (2007). Mapping and revising Markov logic networks for transfer learning. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, pp. 608–614, Vancouver, BC.

Mihalkova, L., & Mooney, R. J. (2009). Learning to disambiguate search queries from short sessions. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD-09)*, pp. 111–127, Bled, Slovenia.

Mihalkova, L., & Mooney, R. J. (2007). Bottom-up learning of Markov logic network structure. In *Proceedings of 24th International Conference on Machine Learning (ICML-2007)*, Corvallis, OR.

Muggleton, S. (2000). Learning stochastic logic programs. In *Proceedings of the AAAI2000 Workshop on Learning Statistical Models from Relational Data*.

Muggleton, S. (1995). Inverse entailment and Progol. *New Generation Computing*, *13*, 245–286.

Nathan Ratliff, J. A. D. B., & Zinkevich, M. (2007). (Online) subgradient methods for structured prediction. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*.

Ng, A. Y. (2004). Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *Proceedings of 21st International Conference on Machine Learning (ICML-2004)*, pp. 78–85, Banff, Alberta, Canada.

Ong, I. M., de Castro Dutra, I., Page, D., & Costa, V. S. (2005). Mode directed path finding. In *Proceedings of the 16th European Conference on Machine Learning (ECML-2005), Porto, Portugal,*, pp. 673–681. Springer.

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, *31*(1), 71–106.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo,CA.

Perkins, S., & Theiler, J. (2003). Online feature selection using grafting. In *Proceedings of 20th International Conference on Machine Learning (ICML-2003)*, pp. 592–599.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, Boston, MA.

Poon, H., & Domingos, P. (2007). Joint inference in information extraction. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, pp. 913–918, Vancouver, British Columbia, Canada.

Provost, F. J., Fawcett, T., & Kohavi, R. (1998). The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the 15th International Conference on Machine Learning (ICML 1998), Madison, Wisconson, USA, July 24-27*, pp. 445–453.

Punyakanok, Roth, D., Yih, W., Zimak, D., & Tu, Y. (2004). Semantic role labeling via generalized inference over classifiers. In *Proceedings of the Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, Boston, MA.

Punyakanok, V., Roth, D., tau Yih, W., & Zimak, D. (2005). Learning and inference over constrained output. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI'05)*, pp. 1124–1129, Edinburgh, Scotland, UK.

Richards, B. L., & Mooney, R. J. (1992). Learning relations by pathfinding. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 50–55, San Jose, CA.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning*, *62*, 107–136.

Riedel, S. (2008). Improving the accuracy and efficiency of MAP inference for Markov logic. In *Proceedings of 24th Conference on Uncertainty in Artificial Intelligence (UAI-2008)*, pp. 468–475, Helsinki, Finland.

Riedel, S., & Meza-Ruiz, I. (2008). Collective semantic role labelling with Markov logic. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL'08)*.

Roth, D., & Yih, W. (2005). Integer linear programming inference for conditional random fields. In *Proceedings of 22nd International Conference on Machine Learning (ICML-2005)*, pp. 737–744.

Roth, D., & Yih, W. (2007). Global inference for entity and relation identification via a linear programming formulation. In Getoor, L., & Taskar, B. (Eds.), *Introduction to Statistical Relational Learning*, pp. 553–580. MIT Press.

Rückert, U., & Kramer, S. (2007). Margin-based first-order rule learning. *Machine Learning*, *70*(2-3), 189–206.

Shalev-Shwartz, S. (2007). *Online Learning: Theory, Algorithms, and Applications*. Ph.D. thesis, The Hebrew University of Jerusalem.

Shalev-Shwartz, S., & Singer, Y. (2007a). Convex repeated games and Fenchel duality. In Schölkopf, B., Platt, J., & Hoffman, T. (Eds.), *Advances in Neural Information Processing Systems 19*, pp. 1265–1272. MIT Press.

Shalev-Shwartz, S., & Singer, Y. (2007b). A unified algorithmic approach for efficient online label ranking. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AIStats)*.

Singla, P., & Domingos, P. (2005). Discriminative training of Markov logic networks. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, pp. 868–873.

Singla, P., & Domingos, P. (2008). Lifted first-order belief propagation. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*, pp. 1094–1099, Chicago, Illinois, USA.

Slattery, S., & Craven, M. (1998). Combining statistical and relational methods for learning in hypertext domains. In Page, D. (Ed.), *Proceedings of the 8th International Workshop on Inductive Logic Programming (ILP-98)*, pp. 38–52. Springer, Berlin.

Snow, R., O'Connor, B., Jurafsky, D., & Ng, A. Y. (2008). Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2008)*, pp. 254–263, Morristown, NJ, USA. Association for Computational Linguistics.

Sutton, C., & McCallum, A. (2007). An introduction to conditional random fields for relational learning. In Getoor, L., & Taskar, B. (Eds.), *Introduction to Statistical Relational Learning*, pp. 93–127. MIT Press.

Sutton, C. A., & McCallum, A. (2009). Piecewise training for structured prediction. *Machine Learning*, *77*(2-3), 165–194.

Szummer, M., Kohli, P., & Hoiem, D. (2008). Learning CRFs using graph cuts. In *Proceedings of the 10th European Conference on Computer Vision (ECCV'08)*, pp. 582–595, Marseille, France. Springer-Verlag.

Taskar, B., Chatalbashev, V., Koller, D., & Guestrin, C. (2005). Learning structured prediction models: a large margin approach. In *Proceedings of 22nd International Conference on Machine Learning (ICML-2005)*, pp. 896–903, Bonn, Germany. ACM.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin Markov networks. In Thrun, S., Saul, L., & Schölkopf, B. (Eds.), *Advances in Neural Information Processing Systems 16*. MIT Press.

Taskar, B., Lacoste-Julien, S., & Jordan, M. I. (2006). Structured prediction, dual extragradient and Bregman projections. *Journal of Machine Learning Research*, *7*, 1627–1653.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *Proceedings of 18th Conference on Uncertainty in Artificial Intelligence (UAI-2002)*, pp. 485–492, Edmonton, Canada.

Tran, S. D., & Davis, L. S. (2008). Event modeling and recognition using markov logic networks. In *Proceedings of the 10th European Conference on Computer Vision (ECCV), Marseille, France, October 12-18*, pp. 610–623.

Tromble, R. W., & Eisner, J. (2006). A fast finite-state relaxation method for enforcing global constraints on sequence decoding. In Moore, R. C., Bilmes, J. A., Chu-Carroll, J., & Sanderson, M. (Eds.), *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, June 4-9, 2006, New York*, pp. 423–430.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. In *Proceedings of 21st International Conference on Machine Learning (ICML-2004)*, pp. 104–112, Banff, Canada.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, *6*, 1453–1484.

Werner, T. (2008). High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF). In *Proceedings of the 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008)*. IEEE Computer Society.

Yu, C.-N. J., & Joachims, T. (2009). Learning structural SVMs with latent variables. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML-2009)*, pp. 1169–1176.

Zelle, J. M., Thompson, C. A., Califf, M. E., & Mooney, R. J. (1995). Inducing logic programs without explicit negative examples. In *Proceedings of the*

Fifth International Workshop on Inductive Logic Programming (ILP-95), pp. 403–416, Leuven, Belgium.

Zhang, T. (2009). On the consistency of feature selection using greedy least squares regression. *Journal of Machine Learning Research*, *10*, 555–568.

Zhu, J., Rosset, S., Hastie, T., & Tibshirani, R. (2003). 1-norm support vector machines. In Thrun, S., Saul, L. K., & Schölkopf, B. (Eds.), *Advances in Neural Information Processing Systems 16 (NIPS 2003)*, pp. 49–56. MIT Press.

Zhu, J., Lao, N., & Xing, E. P. (2010). Grafting-light: fast, incremental feature selection and structure learning of Markov random fields. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2010), Washington, DC, USA*, pp. 303–312. ACM.

Zhu, J., & Xing, E. P. (2009). On primal and dual sparsity of Markov networks.. pp. 1265–1272, Montreal, Quebec, Canada.

Zinkevich, M. (2003). Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *Proceedings of 20th International Conference on Machine Learning (ICML-2003)*, pp. 928–936.

Vita

Tuyen Ngoc Huynh (Huynh Ngoc Tuyen in Vietnamese order) was born in 1981 in Quang Ngai, a small province in the central of Vietnam. After finishing high school at the High School for Gifted Students in 1999, he went to study Computer Engineering at the Ho Chi Minh City University of Technology, where he obtained a Bachelor of Engineering degree and was awarded a University Silver Medal in 2004. In 2005, Tuyen received a fellowship from the Vietnam Education Foundation to pursue the Master program in Computer Science at the University of Texas at Austin. After finishing the Master program in Spring 2007, he continued his doctoral study at the Department of Computer Science, University of Texas at Austin. After graduation, Tuyen will be joining SRI International in Menlo Park as a computer scientist.

Permanent address: hntuyen@gmail.com

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.