Discriminative Learning with Markov Logic Networks

Tuyen N. Huynh
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
hntuyen@cs.utexas.edu

Doctoral Dissertation Proposal

Supervising Professor: Raymond J. Mooney

Abstract

Statistical relational learning (SRL) is an emerging area of research that addresses the problem of learning from noisy structured/relational data. Markov logic networks (MLNs), sets of weighted clauses, are a simple but powerful SRL formalism that combines the expressivity of first-order logic with the flexibility of probabilistic reasoning. Most of the existing learning algorithms for MLNs are in the generative setting: they try to learn a model that maximizes the likelihood of the training data. However, most of the learning problems in relational data are discriminative. So to utilize the power of MLNs, we need discriminative learning methods that well match these discriminative tasks.

In this proposal, we present two new discriminative learning algorithms for MLNs. The first one is a discriminative structure and weight learner for MLNs with non-recursive clauses. We use a variant of ALEPH, an off-the-shelf Inductive Logic Programming (ILP) system, to learn a large set of Horn clauses from the training data, then we apply an L_1-*regularization* weight learner to select a small set of non-zero weight clauses that maximizes the conditional log-likelihood (CLL) of the training data. The experimental results show that our proposed algorithm outperforms existing learning methods for MLNs and traditional ILP systems in term of predictive accuracy, and its performance is comparable to state-of-the-art results on some ILP benchmarks. The second algorithm we present is a max-margin weight learner for MLNs. Instead of maximizing the CLL of the data like all existing discriminative weight learners for MLNs, the new weight learner tries to maximize the ratio between the probability of the correct label (the observable data) and and the closest incorrect label (among all the wrong labels, this one has the highest probability), which can be formulated as an optimization problem called "1-slack" structural SVM. This optimization problem can be solved by an efficient algorithm based on the cutting plane method. However, this cutting plane algorithm requires an efficient inference method as a subroutine. Unfortunately, exact inference in MLNs is intractable. So we develop a new approximation inference method for MLNs based on Linear Programming relaxation. Extensive experiments in two real-world MLN applications demonstrate that the proposed max-margin weight learner generally achieves higher F_1 scores than the current best discriminative weight learner for MLNs.

For future work, our short-term goal is to develop a more efficient inference algorithm and test our max-margin weight learner on more complex problems where there are complicated relationships between the input and output variables and among the outputs. In the longer-term, our plan is to develop more efficient learning algorithms through online learning and algorithms that revise both the clauses and their weights to improve predictive performance.

Contents

1 Introduction

A lot of data in the real world are in the form of relational/structured data such as 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 objects, 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 over the last ten years. Among them, Markov Logic Networks (MLNs) (Richardson & Domingos, 2006) which are sets of weighted first-order clauses are a simple 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). Therefore, in this work, we have chosen MLNs as the model for doing research.

Most of the existing learning algorithms for MLNs are in the generative setting: they try to learn a model that maximizes the likelihood of the training data. However, most of the learning problems in relational data are discriminative. For example, in the biochemical data, the goal is to learn a model that discriminates the active chemical compounds from the inactive ones based on their molecular structures. This problem is called the structure activity relationship prediction (SAR), and it is an important task in drug design and discovery (King, Sternberg, & Srinivasan, 1995). Another example is collective web-page classification. Given a set of web-pages of a department, the task is to simultaneously classify these web-pages into some pre-defined categories based on their content and the hyperlinks between them (Slattery & Craven, 1998). It is, therefore, an important research problem to develop discriminative learning algorithms for MLNs that improves its predictive performance on these discriminative tasks.

In this proposal, we present two new discriminative learning algorithms for MLNs. The first one is a discriminative structure and weight learner for MLNs with non-recursive clauses (Huynh & Mooney, 2008). We use a variant of ALEPH (Srinivasan, 2001), an off-the-shelf Inductive Logic Programming (ILP) system, to learn a large set of Horn clauses from the training data, then we apply an L_1*-regularization* weight learner to select a small set of non-zero weight clauses that maximizes the conditional log-likelihood (CLL) of the training data. The experimental results show that our proposed algorithm outperforms existing learning methods for MLNs and traditional ILP systems in term of predictive accuracy, and its performance is comparable to state-of-the-art results on some ILP benchmarks. The second algorithm we present is a max-margin weight learner for MLNs (Huynh & Mooney, 2009). Instead of maximizing the CLL of the data like all existing discriminative weight learners for MLNs, the new weight learner tries to maximize the ratio between the probability of the correct label (the observable data) and and the closest incorrect label (among all the wrong labels, this one has the highest probability), which can be formulated as an optimization problem called "1-slack" structural SVM (Joachims, Finley, & Yu, 2009). Joachims et al. (2009) presents an efficient algorithm for solving this optimization problem based on the cutting plane method. However, this cutting plane algorithm requires an efficient inference method as a subroutine. Unfortunately, exact inference in MLNs is intractable. So we develop a new approximation inference method for MLNs based on Linear Programming relaxation. One advantage of the max-margin weight learner is that it can be adapted to maximize a variety of performance metrics in addition to classification accuracy (Joachims, 2005). Extensive experiments in two real-world MLN applications demonstrate that the proposed max-margin weight learner generally achieves higher F_1 scores than the current best discriminative weight learner for MLNs.

For future work, our short-term goal is to develop a more efficient inference algorithm and test our max-margin weight learner on more complex problems where there are complicated relationships between the input and output variables and among the outputs. In the longer-term, we plan to work on the following problems:

- *Improving the predictive performance*

  All of the current discriminative weight learners assume that the structure (the clauses) is correct, and only try to fix the weights. However, it is possible that the input structure has some errors that cannot be fixed by only modifying the weights. Hence, we plan to develop new algorithms that try to revise both the clauses and their weights at the same time. On the other hand, we also want to extend max-margin weight learner to optimize other non-linear performance metrics.

- *More efficient learning*

  Most of the existing learning algorithms for MLNs are in the batch setting. However, there are many cases where this approach becomes computationally expensive, especially when the number of training examples are huge. One efficient alternative is online learning which processes the training examples sequentially. We first plan to adapt some of the existing online max-margin weight learning algorithms to the case of MLNs. Then we plan to look at the problem of online structure learning and revision.

2 Background

2.1 MLNs and Alchemy

An MLN consists of a set of weighted first-order clauses. It provides a way of softening first-order logic by making situations in which not all clauses are satisfied less likely but not impossible (Richardson & Domingos, 2006). More formally, let X be the set of all propositions describing a world (i.e. the set of all ground atoms), $\mathscr{F}$ be the set of all clauses in the MLN, w_i be the weight associated with clause $f_i \in \mathscr{F}$, $\mathscr{G}_{f_i}$ be the set of all possible groundings of clause f_i, and Z be the normalization constant. Then the probability of a particular truth assignment $\mathbf{x}$ to the variables in X is defined as (Richardson & Domingos, 2006):

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

where $g(\mathbf{x})$ is 1 if g is satisfied and 0 otherwise, and $n_i(\mathbf{x}) = \sum_{g\in\mathscr{G}_{f_i}} g(\mathbf{x})$ is the number of groundings of f_i that are satisfied given the current truth assignment to the variables in X.

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). This task is formally defined as follows:

$$\begin{aligned}\arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) &= \arg\max_{\mathbf{y}} \frac{1}{Z_x} \exp\left(\sum_i w_i n_i(\mathbf{x},\mathbf{y})\right) \\ &= \arg\max_{\mathbf{y}} \sum_i w_i n_i(\mathbf{x},\mathbf{y}) \end{aligned} \tag{2}$$

where Z_x is the normalization constant over all possible worlds consistent with **x**, and $n_i(\mathbf{x},\mathbf{y})$ is the number of true groundings of clause f_i given the truth assignment $(\mathbf{x},\mathbf{y})$. 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 are 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 computing the conditional probabilities of some unknown query literals, **y**, given some evidence **x**. 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, Ferilli, & Esposito, 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). 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. Recently, 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.

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.

2.2 ILP and Aleph

Traditional 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 Structural Support Vector Machines

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

$$S = ((x_1,y_1),...,(x_n,y_n)) \in (\mathscr{X} \times \mathscr{Y})^n$$

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

$$h_w(x) = \arg\max_{y\in\mathscr{Y}} f_w(x,y)$$

The discriminant function $f_w(x,y)$ takes the form of a linear function:

$$f_w(x,y) = \mathbf{w}^T \Psi(x,y)$$

where $\mathbf{w} \in R^n$ is a parameter vector and $\Psi(x,y)$ is a feature vector relating an input x and output y. The features need to be designed for a given problem so that they capture the dependency structure of y and x and the relations among the outputs y . Then, the goal is to find a weight vector $\mathbf{w}$ that maximizes the margin:

$$\gamma(x_i, y_i; w) = \mathbf{w}^T \Psi(x_i, y_i) - \max_{y'_i \in \mathscr{Y} \setminus y_i} \mathbf{w}^T \Psi(x_i, y'_i)$$

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

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

The slack variables are used to allow some errors in the training data, and the scalar C ≥ 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 (2003) 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

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

Note that, the OP1 is the OP2 with zero-one 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

$$\min_{\mathbf{w},\xi \geq 0} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$$
$$s.t.\ \forall (\bar{y}_1, ..., \bar{y}_n) \in \mathscr{Y}^n : \frac{1}{n}\mathbf{w}^T \sum_{i=1}^{n} [\Psi(x_i, y_i) - \Psi(x_i, \bar{y}_i)] \geq \frac{1}{n}\sum_{i=1}^{n} \Delta(y_i, \bar{y}_i) - \xi$$

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

```
1: **Input:** $S = ((x_1, y_1), ..., (x_n, y_n)), C, \varepsilon$
2: $\mathscr{W} \leftarrow \emptyset$
3: **repeat**
4:
       $$(\mathbf{w}, \xi) \leftarrow \min_{\mathbf{w}, \xi \geq 0} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\xi$$
       $$s.t. \quad \forall(\bar{y}_1, ..., \bar{y}_n) \in \mathscr{W} : \frac{1}{n}\mathbf{w}^T\sum_{i=1}^{n}[\Psi(x_i, y_i) - \Psi(x_i, \bar{y}_i)] \geq \frac{1}{n}\sum_{i=1}^{n}\Delta(y_i, \bar{y}_i) - \xi$$
5:     **for** $i = 1$ **to** $n$ **do**
6:         $\hat{y}_i \leftarrow \arg\max_{\hat{y} \in \mathscr{Y}}\{\Delta(y_i, \hat{y}) + \mathbf{w}^T\Psi(x_i, \hat{y})\}$
7:     **end for**
8:     $\mathscr{W} \leftarrow \mathscr{W} \cup \{(\hat{y}_1, ..., \hat{y}_n)\}$
9: **until** $\frac{1}{n}\sum_{i=1}^{n}\Delta(y_i, \hat{y}_i) - \frac{1}{n}\mathbf{w}^T\sum_{i=1}^{n}[\Psi(x_i, y_i) - \Psi(x_i, \hat{y}_i)] \leq \xi + \varepsilon$
10: return $(\mathbf{w}, \xi)$
```

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 1 solves a Quadratic Programming (QP) problem (line 4) to find the optimal weights corresponding to the current set of constraints $\mathscr{W}$ 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 $\mathscr{W}$. 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, $||\mathbf{w}||_1 = \sum_{i=1}^{n}|w_i|$, 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 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 $\Psi(x, y)$, select a loss function $\Delta(y, \bar{y})$, and design algorithms to solve the two argmax problems:

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

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

3 Discriminative structure and weight learning for MLNs with non-recursive clauses

In this section, we look at a special class of MLNs where all the clauses are non-recursive clauses which contain only one non-evidence literal. We present a new procedure for discriminatively learning both the structure and parameters for this type of MLNs. The proposed approach is a two-step process. The first step uses an off-the-shelf Inductive Logic Programming (ILP) system, ALEPH (Srinivasan, 2001), to generate a large set of potential good clauses. The second step learns the weights for these clauses, preferring to eliminate useless clauses by giving them zero weight. The weight learner in the second step tries to find a small set of non-zero weights which maximizes the conditional likelihood of the data with respect to L_1*-regularization*. We first discuss in details the proposed approach and then the experimental evaluation.

3.1 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 Discriminative Weight Learning

Compared to ALCHEMY's current 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 definite 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,

$$P(Y_j=y_j|X=x) = \frac{exp(\sum_{i\in\mathscr{F}_{Y_j}} w_i n_i(x,y_{[Y_j=y_j]}))}{exp(\sum\limits_{i\in\mathscr{F}_{Y_j}} w_i n_i(x,y_{[Y_j=0]})) + exp(\sum\limits_{i\in\mathscr{F}_{Y_j}} w_i n_i(x,y_{[Y_j=1]}))} \tag{2}$$

where $\mathscr{F}_{Y_j}$ 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:

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

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 (Singla & 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 definite 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 discriminative learning for MLNs, viz. ALCHEMY.
   (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 definite clauses instead of general approximate inference.
   (c) L_1-regularization instead of L_2.

Table 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), x_subst(B1,7,CL), x_subst(D1,6,OCH3), polar(CL,POLAR3),	less_toxic(B1,A1)
polar(OCH3,POLAR2), great_polar(POLAR3,POLAR2), size(CL,SIZE1), size(OCH3,SIZE2), alk_groups(A1,0),	less_toxic(A1,D1)
alk_groups(B1,0), alk_groups(D1,0), great_size(SIZE2,SIZE1), flex(CL,FLEX0), flex(OCH3,FLEX1)	less_toxic(B1,D1)

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).[1] 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 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 1 shows some background facts and examples from one of the datasets, and Table 2 summarizes information about all four datasets.

Table 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 (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.

[1]Since the current ALCHEMY does not support real valued variables, we could not test our approach on the other standard ILP benchmark data sets in molecular biology.

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.

ALEPHExactL2 : 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.

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 (structured prediction) 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 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) for all probabilistic systems by using the AUCCalculator package (Davis & Goadrich, 2006).

3.3.3 Results and Discussion

Tables 3 and 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 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). ALCHEMY's existing non-discriminative structure learners find only a few (3–5) simple clauses. Two of them are unit clauses

Table 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 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, 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 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 definite 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 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 6 compares the average number of clauses learned (after zero-weight clauses are removed) for L_1 and L_2

Table 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 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

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 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

Table 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**

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 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 have presented a new approach to constructing MLNs that discriminatively learns both their structure and parameters to optimize predictive accuracy for a stated target predicate when given evidence specified with a defined set of background predicates. It 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.

4 Max-Margin Weight Learning for MLNs

In Section 3, 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 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). In this section, we present a max-margin approach to weight learning in MLNs based on the general framework for max-margin training of structured models (Tsochantaridis et al., 2005; Joachims et al., 2009).

4.1 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:

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

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 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. For example, if we apply it to a CRF (Lafferty, McCallum, & Pereira, 2001) then the result model is an M3N (Taskar et al., 2003). In fact, this translation is the connection between log-linear models and linear classifiers (Collins, 2004).

In turn, the max-margin problem above can be formulated as a "1-slack" structural SVM as described in section 2.3:

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

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

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 $\Psi(x,y)$ 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 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 2.1. 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.

4.2 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, & 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 factor, i.e. infinite potential function. 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 (ILP) 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 ILP 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 ILP 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 ILP problem:
$$\begin{aligned} 1 - y_i \geq z_j, &\quad i \in I_j^+ \\ y_i \geq z_j, &\quad i \in I_j^- \end{aligned}$$

The final ILP has the following form:

Optimization Problem 5 (OP5):

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

Algorithm 2 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 $\mathbf{y}$
```

where C^+ and C^- are the set of clauses with positive and negative weights respectively. This ILP 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 ILP 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 ILP 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 ILP problem. In our case, the original ILP 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 ILP 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 2), which we found to work well. In each step, 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 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 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 (Slattery & Craven, 1998) 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.

$Has(+word, page) => PageClass(+class, page)$
$!Has(+word, page) => PageClass(+class, page)$
$PageClass(+c1, p1) \wedge Linked(p1, p2) => PageClass(+c2, p2)$

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 , we used the dataset and MLN used in (Poon & Domingos, 2007). The dataset has 1,563 citations and each of them is segmented into three fields: *Author*, *Title* and *Venue*. The dataset has four disconnected segments corresponding to four different research topics. We used the simplest MLN in (Poon & Domingos, 2007), which is the isolated segmentation model. Despite its simplicity, after grounding, this model results in a large network with more than 30,000 query atoms and 110,000 ground clauses.

All the datasets and MLNs can be found at the Alchemy website.[2]

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 1 and LP problems in the LP-relaxation MPE inference, we used the MOSEK [3] 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, the harmonic mean of recall and precision, to measure the performance of each algorithm on the WebKB and CiteSeer datasets. This is the standard evaluation metric in multi-class text categorization and information extraction.

4.4.3 Results and Discussion

Table 8 and 10 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

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

Table 8: 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 9: 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

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 8 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 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 current best discriminative weight learner for MLNs, PSCG-MCSAT.

Table 9 compares the performance of the proposed LP-relaxation MPE inference algorithm against MC-SAT 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 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., 2003), 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 10, the performance of max-margin methods are very close to those of PSCG. However, its performance is much more stable. Table 11 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

Table 10: 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 11: 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

drastically on both training and testing after a certain number of iterations. For example, from Table 11 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 12 and 13 compares the performance of the MM weight learners against the some of the systems described in section 3 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 section. 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.

Table 12: 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 13: 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

4.5 Related Work

Our work is related to various previous projects. Among them, M3N (Taskar et al., 2003) is probably the most related. It is a special case of structural SVMs where the feature function $\Psi(x,y)$ 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., 2003). 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., 2003), extragradient methods (Taskar, Lacoste-Julien, & Jordan, 2006), or exponentiated gradient methods (Collins, Globerson, Koo, Carreras, & Bartlett, 2008). As mentioned in (Taskar et al., 2003), 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; Huynh & Mooney, 2008; Biba et al., 2008). We have mentioned some of them (Singla & Domingos, 2005; Lowd & Domingos, 2007; Huynh & Mooney, 2008) 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. Extending the max-margin framework to structure learning is an area for future work.

4.6 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 current best discriminative MLN weight learner.

5 Proposed Research

5.1 Improving the predictive performance

5.1.1 Revising MLNs

In the previous section, we looked at the problem of learning weights for a given set of clauses provided as the input. We assumed that the input clauses are correct and only learnt weights for them. However, these clauses usually provided by domain experts may be too general or too specific, thus they are not good for prediction. In fact, in many cases, when we look at the learnt MLNs, there are clauses whose weights are very small (nearly zero), which do not have any predictive power. Therefore, it would be better if we could revise these clauses. First-order theory revision is a well-studied problem (Wrobel, 1996). However, revising a first-order probabilistic model like MLNs is a new problem and only a few work have looked at this problem (Revoredo & Zaverucha, 2002; Paes, Revoredo, Zaverucha, & Costa, 2005; Mihalkova, Huynh, & Mooney, 2007). All of the existing work is based on FORTE (Richards & Mooney, 1995), a successful first-order theory revision system. Among them, (Mihalkova et al., 2007) is the one that deals with revising MLNs in the context of transfer learning. Based on this work, we propose the following procedure to revise an MLN in a discriminative manner:

1. **Step 1**: This step is similar to the Self-Diagnosis step of the RTAMAR (Mihalkova et al., 2007) to generate revision points for the next steps. However, we make some modifications. First, instead of transferring the weights from the source MLN to the target MLN, we learn the weights for the input MLN by using a discriminative weight learner. The clauses with very small weights are the candidates for revision. This is different from RTAMAR, where all clauses are inspected by the algorithm. Then, we set the values of all of the groundings of the query predicates to unknown, and run inference with the current MLN to find where it makes wrong predictions. For a wrong prediction of a query atom X, all ground clauses containing X of the candidate clauses will be inspected and classified into two bins: **[Relevant]** and **[Irrelevant]**. The **[Relevant]** bin consists of clauses whose premises are satisfied and the **[Irrelevant]** bin contains clauses whose premises are not satisfied. In comparison to RTAMAR, we combine the **[Relevant; Good]** and **[Relevant; Bad]** into the **[Relevant]** bin and the same for the other two bins **[Irrelevant; Good]** and **[Irrelevant; Bad]**. The reason for that is because assigning a negative weight to a clause in the bad bin will make it become a good one and vice versa. So, in term of predictive power, the clauses in the good bin and bad bin are the same, and it is the job of the weight learner to assign the correct weights to these clauses. The clauses in the **[Relevant]** bin are the one that too general since the premises are satisfied but the weights are nearly zero, and the ones in the **[Irrelevant]** bin are too specific since their premises are not satisfied. Then we count how many times a candidate clause falls into one of the two bins by diagnosing all the wrong predictions. Finally, if a candidate clause is placed in the **[Relevant]** or **[Irrelevant]** bin more than *p* percent of the time, it is marked for lengthening or shortening respectively. The role of the threshold *p* is to ignore the random errors introduced by the inference algorithm. The value of *p* will be determined in experiment.

2. **Step 2**: To revise the marked clauses, we can use the top-down beam search approach in RTAMAR, a stochastic local search (Paes, Zaverucha, & Costa, 2007) or a bottom-up approach (Duboc, Paes, & Zaverucha, 2008). The weights of the candidate clauses can be learnt in an inexpensive way by keeping the weights of other clauses fixed. Then we can run inference and score the candidate clauses by some discriminative metrics such as accuracy or CLL. This step will be terminated when the algorithm cannot find any new clause that improves the score of the model.

5.1.2 Optimizing non-linear performance metrics

In section 4.3, we have shown how to optimize a linearly decomposable loss function such as the Hamming loss. Though, there are other popular performance metrics (e.g. F_1, ROCArea) that are not linearly decomposable. Thus, the loss functions corresponding to these metrics (for example the F_1 loss = $1 - F_1$) are also non-linear. Finding the most violated constraint for these loss functions is a much harder problem. One simple approximation is to find the MPE solution or N-best MPE solution (Yanover & Weiss, 2003) then check to see whether any of them violates the constraint. If no violated constraint is found, then the current weight vector is a good solution. This approach will not guarantee to find the optimal weights but hopefully it can find a good one.

Regarding the F_1 loss which can be written as:

$$\Delta_{F_1}(\mathbf{y^T}, \mathbf{y}) = 1 - \frac{2TP}{2TP + FP + FN} = \frac{FP + FN}{2TP + FP + FN}$$

where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives. The quantities TP, FP, and FN can be represent as linear functions of the query variables y_i's:

$$\begin{aligned} TP &= \sum_{i:y_i^T=1} y_i \\ FP &= \sum_{i:y_i^T=0} y_i \\ FN &= \sum_{i:y_i^T=1} (1 - y_i) \end{aligned}$$

Thus, the F_1 loss is a linear-fractional function of the variables y_i's. Adding this linear-fractional loss to the objective function of the MPE problem (OP5), we have a fractional programming problem. So we may use techniques in fractional programming (Stancu-Minasian, 1997) such as Dinkelbach's algorithm (Dinkelbach, 1967) to find the most violated constraint for the F_1 loss.

5.2 More efficient learning algorithm

5.2.1 Online learning

So far, we have presented algorithms for learning in the batch setting. However, there are many cases where this approach becomes computationally expensive, especially when the number of training examples are huge. For example, considering the problem of extracting information from text document, each document is an example. To get a high accuracy model, we usually train on a huge corpus containing thousands of documents. This makes batch learning costly and inefficient since we need to keep all of the training examples in memory and run inference on thousands of training examples in each iteration. One efficient alternative is online learning which processes the training examples sequentially.

Some of the existing weight learning algorithms for MLNs already have the ability to do online learning. The structured perceptron algorithm (Singla & Domingos, 2005) and its variant, contrastive divergence (Lowd & Domingos, 2007), can operate in an online fashion by processing one example at a time and updating the weights whenever it makes a wrong prediction. To handle overfitting, we can use some variants of the structured perceptron such as voted perceptron or averaged perceptron (Collins, 2002). However, these simple perceptron-based algorithm do not look for a solution that has a large margin. There are some existing work on margin-based online learning for structured prediction (e.g., Crammer, McDonald, & Pereira, 2005;

Crammer, Dekel, Keshet, Shalev-Shwartz, & Singer, 2006; Keshet, Shalev-Shwartz, Singer, & Chazan, 2007; Shalev-Shwartz, 2007; Nathan Ratliff & Zinkevich, 2007). We plan to adapt these algorithms to the case of MLNs. For example, we can rewrite the M3LNs optimization problem (the OP4) as follows:

Optimization Problem 6 (OP6): Max-Margin Markov Logic Networks

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

This optimization problem can be cast as an unconstrained optimization problem:

Optimization Problem 7 (OP7):

$$\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C[\max_{\bar{\mathbf{y}}\in Y}\{\Delta(\mathbf{y},\bar{\mathbf{y}}) + \mathbf{w}^T\mathbf{n}(\mathbf{x},\bar{\mathbf{y}})\} - \mathbf{w}^T\mathbf{n}(\mathbf{x},\mathbf{y})]$$

since the inequality constraint in OP6 becomes an equality at the optimal solution. Then we can apply the online subgradient algorithm for structured prediction (Nathan Ratliff & Zinkevich, 2007) to solve the OP7. To find the subgradients for this problem, we need to solve the loss-augmented inference problem. We have developed an approximation algorithm for solving this problem in section 4.3.

Thus far, we have only discussed online learning of weights, but a complete online learning system must also be able to do online structure learning and revision since the errors may come from the structure. Thus only fixing the weights is not enough. So we also plan to look at the problem of online structure learning and revision. In online learning, we need to do both structure learning and revision together since to fix prediction errors on a given example we may need to either revise the current model or learn new clauses from the example or do both. This problem is related to the problem of incremental theory refinement (Mooney, 1992). As pointed out in (Mooney, 1992), an incremental/online learning system may run into the problem of "snowballing": based on a small amount of data, the system makes bad initial changes to the model, these bad changes may not be fixed in later steps, and result in a over-complicated model which hurts the general performance. So the challenge in online structure learning and revision is to be able to make good decisions based on only a small amount of data, the current example or a subset of examples seen so far. However, there is a good news for MLNs. Since MLNs define a probability over possible worlds or interpretations (De Raedt & Kersting, 2008), its training examples are Herbrand interpretations which contain more information than positive/negative examples used in traditional ILP systems.

5.2.2 Efficient MPE and loss-augmented MPE inference algorithms

The major weakness of the LP-relaxation inference algorithm presented in section 4.2 is that it operates on the ground Markov network, i.e. the whole MLN must be fully grounded. Fully instantiating an MLN takes a lot of time, requires a lot of memory, and becomes impossible when there are many query atoms and the model contains complex relationships among them, for example entity resolution and joint learning problem (Singla & Domingos, 2006; Poon, Domingos, & Sumner, 2008). There is some existing work on efficient inference methods for MLNs that try not to fully ground the whole network such as lazy inference (Singla & Domingos, 2006; Poon et al., 2008), cutting plane inference (CPI) (Riedel, 2008), and lifted inference (Singla & Domingos, 2008). These algorithms exploit different aspects of relational domains and structures of the ground Markov network. Lazy inference takes advantage of the sparsity of relational domains: most query atoms are false. CPI utilizes the redundancy of the ground Markov network: predictions based on local information already satisfy the global constraints. Lifted inference exploits the symmetry of the ground Markov network: some structures appear multiple times in the network. So we plan to combine the advantages of these algorithms into a more efficient inference algorithm. For example, we can reduce the size of the network constructed by the CPI method by taking into account the the sparsity of relational domains. Since most query atoms are false, initially we only need to ground clauses that may make a query atom become true. These are ground clauses that are unsatisfied (or satisfied if the weight is negative) assuming that all the query atoms are false. So instead of initializing the partial network with all the groundings of non-recursive clauses (Riedel, 2008), we only need to consider a subset of them. In the case that the partial network is still a large one, we can apply the methods in lifted inference to construct a compressed factor graph of the network. Then we can run the LP-relaxation inference algorithm on the compressed factor graph.

The above method can also be used for solving the loss-augmented inference problem if the loss function is decomposable into a set of ground clauses. For example, the Hamming loss can be represented by adding a unit clause with weight 1 or -1 for every false or true grounding of the query predicates respectively.

5.3 Experiments on additional problems

We plan to apply M3LNs to more complex problems where there are complicated relationships between the input and output variables and among the output ones. One such problem is joint learning in natural language processing. For example, considering the problem of jointly recognizing entities and relations in sentences (Roth & Yih, 2002), first-order logic provides a natural way to express the patterns for identifying entities and relations from local information such as lexical and syntactical information and also the global relationships between entity types and relations, etc. Then we can use the max-margin weight learner to learn weights for these clauses. Another interesting joint learning problem is "scene understanding" in computer vision(Li & Li, 2007; Heitz, Gould, Saxena, & Koller, 2008; Li, Socher, & Fei-Fei, 2009), where the key problem is to simultaneously recognize the overall scene and the component objects of a given image. To achieve a high performance, besides the visual features, ones need to take into account the relationships between objects and between objects and scenes. Learning these types of relationships is what a statistical relational learning model like MLNs is good for. The challenge of joint learning is to be able to handle a large amount of complicated data. The online learning and efficient inference algorithm described in previous sections will help to solve this challenge. Besides, we also want to look at the activity recognition problem (Tran & Davis, 2008) which also requires the ability to handle complicated relations and interactions between objects.

6 Conclusions

Learning from noisy structured/relational data is one of the key problems in machine learning. Markov logic networks, a formalism that combines the expressivity of first-order logic with the flexibility of probabilistic reasoning, are a powerful model to handle such kind of data. Discriminative learning is an important research problem in MLNs since most of learning problems in relational data are discriminative. In this proposal, we have presented two new discriminative learning algorithms for MLNs. The first algorithm is a discriminative structure and weight learner for MLNs with non-recursive clauses, and the second one is a max-margin weight learner for MLNs. For future work, our short-term goal is to develop a more efficient MPE inference algorithm for MLNs and apply our max-margin weight learner to more complex problems which contain complicated relationships between input and output variables and among the ouputs such as joint learning problem in natural language processing. In the longer-term, our plan is to develop more efficient learning algorithms through online learning and algorithms that revise both the clauses and their weights to improve predictive performance.

Acknowledgments

We thank Niels Landwehr for helping us set up the experiment with ALEPH. We also thank Daniel Lowd and Hoifung Poon for useful discussions and helping with the experiments. This research is sponsored by DARPA and managed by AFRL under contract FA8750-05-2-0283. The project is also partly support by ARO grant W911NF-08-1-0242. Most of the experiments were run on the Mastodon Cluster, provided by NSF Grant EIA-0303609. The first author also thanks the Vietnam Education Foundation (VEF) for its sponsorship.

References

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.

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.

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.

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

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.

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.

Crammer, K., McDonald, R., & Pereira, F. (2005). Scalable large-margin online learning for structured classification. Tech. rep., Department of Computer and Information Science, University of Pennsylvania.

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

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.

De Raedt, L., & Kersting, K. (2008). Probabilistic inductive logic programming. In Raedt, L. D., Frasconi, P., Kersting, K., & Muggleton, S. (Eds.), *Probabilistic Inductive Logic Programming*, Vol. 4911 of *Lecture Notes in Computer Science*, pp. 1–27. Springer.

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.

Dinkelbach, W. (1967). On nonlinear fractional programming. *Management Science*, *13*(7), 492–498.

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.

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.

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

Heitz, G., Gould, S., Saxena, A., & Koller, D. (2008). Cascaded classification models: Combining models for holistic scene understanding. In Koller, D., Schuurmans, D., Bengio, Y., & Bottou, L. (Eds.), *Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008*, pp. 641–648. MIT Press.

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

Huynh, T. N., & Mooney, R. J. (2008). Discriminative structure and parameter learning for Markov logic networks. In *Proceedings of 25th International Conference on Machine Learning (ICML-2008)*, pp. 416–423, Helsinki, Finland.

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 Knowledge Discovery in Databases (ECML PKDD 2009), Part I*, pp. 564–579.

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.

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.

Keshet, J., Shalev-Shwartz, S., Singer, Y., & Chazan, D. (2007). A large margin algorithm for speech-to-phoneme and music-to-score alignment. *IEEE Transactions on Audio, Speech & Language Processing*, *15*(8), 2373–2382.

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. (2009). Learning Markov logic network structure via hypergraph lifting. In *Proceedings of the 26th International Conference on Machine Learning (ICML-2009)*, pp. 505–512, Montreal, Quebec, Canada.

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., & 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.

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)*.

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.

Li, L.-J., & Li, F.-F. (2007). What, where and who? Classifying events by scene and object recognition. In *Proceedings of the 11th International Conference on Computer Vision (ICCV-2007)*, pp. 1–8.

Li, L.-J., Socher, R., & Fei-Fei, L. (2009). Towards total scene understanding:classification, annotation and segmentation in an automatic framework. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*.

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

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.

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

Mooney, R. (1992). Batch versus incremental theory refinement. In *Proceedings of the 1992 AAAI Spring Symposium on Knowledge Assimilation*.

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.

Paes, A., Revoredo, K., Zaverucha, G., & Costa, V. S. (2005). Probabilistic first-order theory revision from examples. In *Proceedings of the 15th International Conference on Inductive Logic Programming (ILP-2005)*, pp. 295–311, Bonn, Germany.

Paes, A., Zaverucha, G., & Costa, V. S. (2007). Revising first-order logic theories from examples through stochastic local search. In *Proceedings of the 17th International Conference on Inductive Logic Programming (ILP-2007)*, pp. 200–210.

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.

Poon, H., Domingos, P., & Sumner, M. (2008). A general method for reducing the complexity of relational inference and its application to MCMC. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08)*, pp. 1075–1080.

Revoredo, K., & Zaverucha, G. (2002). Revision of first-order Bayesian classifiers. In *Proceedings of the 12th International Conference on Inductive Logic Programming (ILP-2002)*, pp. 223–237.

Richards, B. L., & Mooney, R. J. (1995). Automated refinement of first-order Horn-clause domain theories. *Machine Learning*, *19*(2), 95–131.

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.

Roth, D., & Yih, W.-t. (2002). Probabilistic reasoning for entity & relation recognition. In *Proceedings of the 19th international conference on Computational linguistics*, pp. 1–7, Taipei, Taiwan.

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.

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. (2006). Memory-efficient inference in relational domains. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06)*.

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.

Srinivasan, A. (2001). *The Aleph manual*. `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/`.

Stancu-Minasian, I. (1997). *Fractional Programming: Theory, Methods and Applications*. Kluwer Academic Publishers.

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. (2003). Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16 (NIPS 2003)*.

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.

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.

Wrobel, S. (1996). First order theory refinement. In De Raedt, L. (Ed.), *Advances in Inductive Logic Programming*, pp. 14–33. IOS Press, Amsterdam.

Yanover, C., & Weiss, Y. (2003). Finding the M most probable configurations in arbitrary graphical models. In Thrun, S., Saul, L. K., & Schölkopf, B. (Eds.), *Advances in Neural Information Processing Systems 16 (NIPS 2003)*. MIT Press.

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., & Xing, E. P. (2009). On primal and dual sparsity of Markov networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML-2009)*, pp. 1265–1272, Montreal, Quebec, Canada.