In *Proceedings of the 25th International Conference on Machine Learning (ICML),*. Helsinki, Finland, July 2008.

Discriminative Structure and Parameter Learning for Markov Logic Networks

Tuyen N. Huynh HNTUYEN@CS.UTEXAS.EDU
Raymond J. Mooney MOONEY@CS.UTEXAS.EDU
Department of Computer Sciences, The University of Texas at Austin,1 University Station C0500, Austin, TX 78712, USA

Abstract

Markov logic networks (MLNs) are an expressive representation for statistical relational learning that generalizes both first-order logic and graphical models. Existing methods for learning the logical structure of an MLN are not discriminative; however, many relational learning problems involve specific target predicates that must be inferred from given background information. We found that existing MLN methods perform very poorly on several such ILP benchmark problems, and we present improved discriminative methods for learning MLN clauses and weights that outperform existing MLN and traditional ILP methods.

1. Introduction

Statistical relational learning (SRL) concerns the induction of probabilistic knowledge that supports accurate prediction for multi-relational structured data (Getoor & Taskar, 2007). Markov Logic Networks (MLNs) are a recently developed SRL model that generalizes both full first-order logic and Markov networks (Richardson & Domingos, 2006). An MLN is represented as a set of weighted clauses in first-order logic, and learning an MLN decomposes into *structure learning*, learning the logical clauses, and *parameter learning*, setting the weight of each clause. Existing structure-learning algorithms for MLNs (Kok & Domingos, 2005; Mihalkova & Mooney, 2007) are non-discriminative and attempt to learn a set of clauses that is equally capable of predicting the truth value of all predicates given an arbitrary set of evidence. However, in many learning problems, there is a specific *target predicate* that must be inferred given evidence data about other *background predicates* used to describe the input data. Most traditional *Inductive Logic Programming* (ILP) methods focus on discriminative relational learning (Dzeroski, 2007); however, they do not address the issue of uncertainty. Discriminative methods have been developed for parameter learning in MLNs (Singla & Domingos, 2005; Lowd & Domingos, 2007); however, they do not address structure learning.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008. Copyright 2008 by the author(s)/owner(s).

We have found that existing MLN structure learning methods perform very poorly when tested on several benchmark ILP problems on relating the activity of chemical compounds to their structure (King et al., 1995). This led us to develop new discriminative structure and parameter learning algorithms for MLNs whose performance on these problems surpasses that of traditional ILP methods. The overall approach is to use traditional ILP methods to construct a large number of potentially useful clauses, and then use discriminative MLN parameter learning methods to properly weight them, preferring to assign zero weights to clauses that do not contribute significantly to overall predictive accuracy, thereby eliminating them. Our structure learning component utilizes many of the clauses considered during the search conducted by a specific configuration of ALEPH (Srinivasan, 2001). Our parameter learning component utilizes an exact probabilistic inference algorithm for MLNs with only non-recursive definite clauses. Our parameter learner also uses L_1-regularization (Lee et al., 2006) instead of the normal L_2 in order to encourage assigning zero weights to clauses, thereby simplifying the theory. We present experimental results that demonstrate that all three of these enhancements contribute significantly to improving the performance of our system over existing MLN and ILP methods.

The remainder of the paper is organized as follows. Section 2 provides some background on MLNs, ALCHEMY (Kok et al., 2005), and ALEPH. Section 3 presents our new structure and parameter learning algorithms. Section 4 presents our experimental evaluation of these methods. Section 5 discusses related work, and section 6 presents our conclusions.

2. Background

2.1. 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.

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.

2.2. MLNs and Alchemy

An MLN consists of a set of weighted first-order formulae (also called clauses or rules). It provides a way of softening first-order logic by making situations in which not all formulae are satisfied less likely but not impossible (Richardson & Domingos, 2006). More formally, let X be the set of all propositions describing a world (i.e. the set of all the ground atoms[1]), $\mathcal{F}$ be the set of all clauses in the MLN, w_i be the weight associated with clause $f_i \in \mathcal{F}$, $\mathcal{G}_{f_i}$ be the set of all possible groundings of clause f_i, and $\mathcal{Z}$ be the normalizing constant. Then the probability of a particular truth assignment x to the variables in X is given by the formula (Richardson & Domingos, 2006):

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

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

[1]Ground atoms are predicates without variables, which are formed by replacing all the variables in predicates by constants.

In order to perform inference for an MLN, L, one needs to produce its corresponding ground Markov network. As described by Richardson and Domingos (2006), this is done by including a node for every possible grounding of the predicates in L and an edge between two nodes if they appear together in a grounding of a clause in L. The nodes appearing together in a ground clause form a clique. In general, exact inference in MLNs is intractable, so to calculate the probability that a ground atom or a set of them has a particular truth assignment given some evidence, one needs to run an approximate inference algorithm such as MCMC on this ground Markov network. MC-SAT (Poon & Domingos, 2006) is currently the best inference method for MLNs.

As previously mentioned, learning an MLN consists of two tasks: structure learning and weight learning. The weight learning component is independent of the structure learning one. It can learn weights for clauses produced by structure learning or written by a human expert. There are two approaches to weight learning: generative and discriminative. The former is used when there is no specific target predicate, and the latter when we know which predicate is to be queried. The current state-of-the-art discriminative weight learner is *preconditioner scaled conjugate gradient* (PSCG) (Lowd & Domingos, 2007). This algorithm uses samples from MC-SAT to approximate the intractable expected counts of satisfied clauses w.r.t. the current model. These counts are needed to compute the gradient and Hessian of the conditional log-likelihood (CLL) of an MLN. The inverse diagonal Hessian is used as the preconditioner in this method.

Regarding structure learning, there are currently two algorithms for learning clauses for MLNs. The first algorithm was proposed by Kok and Domingos (2005). This algorithm uses a top-down approach and can perform either beam-search or shortest-first search over the space of clauses. The candidate clauses are scored using *weighted pseudo log-likelihood* (WPLL) (Kok & Domingos, 2005). In contrast, the second algorithm, BUSL (Mihalkova & Mooney, 2007) follows a more bottom-up approach. 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. Both of these algorithms can be constrained to only learn clauses that contain a given target predicate by setting their "ne" (non-evidence) parameter to that predicate. However, they are not designed for discriminative learning since they try to find a set of clauses which maximizes WPLL, a non-discriminative measure.

ALCHEMY (Kok et al., 2005) is an open source software system for MLNs. It has implementations of all the existing algorithms for structure learning, generative weight learning, discriminative weight learning, and inference for MLNs. It is also a framework for developing new algorithms for MLNs. Our proposed algorithm is implemented in this framework.

3. Discriminative MLN Algorithms

In this section, we describe our two-step process for discriminatively learning both the structure and parameters of an MLN. The first step uses ALEPH to learn a large set of potential clauses. The second step learns the weights for these clauses, preferring to eliminate useless clauses by giving them zero weight.

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 CLL of each of the query atoms. 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 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 their weights instead of the approximate inference algorithm that ALCHEMY uses to handle the general case. Since these clauses are non-recursive definite clauses in which the target predicate only appears once, multiple query atoms will not appear together in any grounding of any clause. For MLNs, this means that the Markov blanket of a query atom only contains evidence atoms. Consequently, the query atoms are independent given the evidence. Let Y be the set of query atoms and X be the set of evidence atoms, the conditional log likelihood of Y given X in this case is:

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

and,

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

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

$$\frac{\partial}{\partial w_i} \log P(Y=y|X=x) =$$

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

$$-P(Y_j=1|X=x)n_i(x, y_{[Y_j=1]})]$$

Notice that the sum of the last two terms in the gradient is the expected count of the number of true grounding of the i'th formula. In general, computing this expected count requires performing approximate inference under the model. For example, Singla and Domingos (2005) ran MAP inference and used the counts in the MAP 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 predicate appears 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 et al. (2006) 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., 2006). 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 et al., 2007), our experiments confirm that it results in simpler and more accurate learned models compared to L_2-regularization.

4. 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 et al., 2006), TFOIL (Landwehr et al., 2007), and RUMBLE (Rückert & Kramer, 2008).
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.

4.1. 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).[2] 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

4.2. 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.

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.

ALEPH**PSCG:** Uses the discriminative weight learner PSCG to learn MLN weights for the clauses in the final theory returned by ALEPH. Note that PSCG also uses L_2-regularization.

ALEPH**ExactL2** : Uses the limited-memory BFGS algorithm (Liu & Nocedal, 1989) implemented in ALCHEMY to learn discriminative MLN weights for the clauses in the final theory returned by ALEPH. The objective function is CLL with L_2 regularization. The CLL is computed exactly as described in Section 3.2.

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).

[2]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.

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

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 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

4.3. 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 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. Due to space constraints, we do not report the corresponding AUC-ROC results, which are qualitatively similar.

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 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

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**

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**

that have a well-defined possible worlds semantics.

5. 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-descriminative 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 et al., 2005) used ALEPH to propose candidate features for a Bayesian network classifier. Muggleton(2000) used PROGOL, another popular ILP system, to learn clauses for Stochastic Logic Programs (SLPs).

When restricted to learning non-recursive clauses for classification, our approach is equivalent to using ALEPH to construct features for use by L_1-regularized logistic regression. Under this view, our approach is closely related to MACCENT (Dehaspe, 1997), which uses a greedy approach to induce clausal constraints that are used as features for maximum-entropy classification. One difference between our approach and MACCENT is that we use a two-step process instead of greedily adding one feature at a time. In addition, our clauses are induced in a bottom-up manner while MACCENT uses top-down search; and our weight learner employs L_1-regularization which makes it less prone to overfitting. Unfortunately, we could not compare experimentally to MACCENT since "only an implementation of a propositional version of MACCENT is available, which only handles data in attribute-value (vector) format" (Landwehr et al., 2007). Additionally, MLNs are a more expressive formalism that also allows for structured prediction, as demonstrated by our results that include a transitivity constraint on the target relation.

6. Conclusions

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.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. We also thank Niels Landwehr for helping us set up the experiment with ALEPH. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and managed by the Air Force Research Laboratory (AFRL) under contract FA8750-05-2-0283. 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. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of DARPA, AFRL, VEF, or the United States Government.

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**

References

Andrew, G., & Gao, J. (2007). Scalable training of L_1-regularized log-linear models. *Proc. of 24th Intl. Conf. on Machine Learning (ICML-2007)* (pp. 33–40). Corvallis, OR: Omnipress.

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

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

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

Dehaspe, L. (1997). Maximum entropy modeling with clausal constraints. *Proc. of the 7th Intl. Workshop on Inductive Logic Programming* (pp. 109–124). Springer-Verlag.

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 L. Getoor and B. Taskar (Eds.), *Introduction to statistical relational learning*, 57–92. Cambridge, MA: MIT Press.

Getoor, L., & Taskar, B. (Eds.). (2007). *Statistical relational learning*. Cambridge, MA: MIT Press.

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

Kok, S., & Domingos, P. (2005). Learning the structure of Markov logic networks. *Proc. of 22nd Intl. Conf. on Machine Learning (ICML-2005)*. Bonn,Germany.

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

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. *Proc. of the Twenty-First Natl. Conf. on Artificial Intelligence (AAAI-2006)*. Boston, MA: AAAI Press.

Lee, S., Ganapathi, V., & Koller, D. (2006). Efficient structure learning of Markov networks using L_1-regularization. *Advances in Neural Information Processing Systems 18*.

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

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

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

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

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

Ng, A. Y. (2004). Feature selection, L_1 vs. L_2 regularization, and rotational invariance. *Proc. of 21st Intl. Conf. on Machine Learning (ICML-2004)* (pp. 78–85). Banff, Alberta, Canada: ACM.

Poon, H., & Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. *Proc. of the Twenty-First Natl. Conf. on Artificial Intelligence (AAAI-2006)*. Boston, MA.

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

Rückert, U., & Kramer, S. (2008). Margin-based first-order rule learning. *Machine Learning*, *70*, 189–206.

Singla, P., & Domingos, P. (2005). Discriminative training of Markov logic networks. *Proc. of 20th Natl. Conf. on Artificial Intelligence (AAAI-2005)* (pp. 868–873).

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