Discriminative Structure Learning of Markov Logic Networks

Marenglen Biba, Stefano Ferilli and Floriana Esposito

Department of Computer Science, University of Bari
Via E.Orabona, 4 - 70125, Bari, Italy
{biba,ferilli,esposito}@di.uniba.it

Abstract. Markov Logic Networks (MLNs) combine Markov networks and first-order logic by attaching weights to first-order formulas and viewing these as templates for features of Markov networks. Learning the structure of MLNs is performed by state-of-the-art methods by maximizing the likelihood of a relational database. This can lead to suboptimal results given prediction tasks. On the other hand better results in prediction problems have been achieved by discriminative learning of MLNs weights given a certain structure. In this paper we propose an algorithm for learning the structure of MLNs discriminatively by maximimizing the conditional likelihood of the query predicates instead of the joint likelihood of all predicates. The algorithm chooses the structures by maximizing conditional likelihood and sets the parameters by maximum likelihood. Experiments in two real-world domains show that the proposed algorithm improves over the state-of-the-art discriminative weight learning algorithm for MLNs in terms of conditional likelihood. We also compare the proposed algorithm with the state-of-the-art generative structure learning algorithm for MLNs and confirm the results in [22] showing that for small datasets the generative algorithm is competitive, while for larger datasets the discriminative algorithm outperfoms the generative one.

1 Introduction

Many real-world application domains are characterized by both uncertainty and complex relational structure. Statistical learning focuses on the former, and relational learning on the latter. Probabilistic Inductive Logic Programming (PILP) [7] or Statistical Relational Learning [10] aim at combining the power of both. PILP and SRL can be viewed as combining ILP principles (such as refinement operators) with statistical learning. One of the representation formalisms in this area is Markov Logic which subsumes both finite first-order logic and probabilistic graphical models as special cases [30]. Upon this formalism, Markov Logic Networks (MLNs) can be built serving as templates for constructing Markov Networks (MNs). In Markov Logic a weight is attached to each clause and learning an MLN consists in structure learning (learning the clauses) and weight learning (setting the weight of each clause).

In [30] structure learning was performed through CLAUDIEN [6] followed by a weight learning phase in which maximum pseudo-likelihood [1] weights were learned for each clause. In [14] structure is learned in a single phase using weighted pseudo-likelihood as the evaluation measure in a beam search. The algorithm performs systematic greedy search being therefore very suscetible to local optima. The state-of-the-art algorithm for generative structure learning is that in [24] which follows a bottom-up approach trying to consider fewer candidates for evaluation. This algorithm uses a propositional Markov network learning method to construct template networks that guide the construction of candidate clauses. In this way, it generates fewer clauses for evaluation.

Generative approches optimize the joint distribution of all the variables. This can lead to suboptimal results for predictive tasks because of the mismatch between the objective function used (likelihood or a function thereof) and the goal of classication (maximizing accuracy or conditional likelihood). In contrast discriminative approaches maximize the conditional likelihood of a set of outputs given a set of inputs [16] and this often produces better results for prediction problems. In [31] the voted perceptron based algorithm for discriminative weight learning of MLNs was shown to greatly outperform maximum-likelihood and pseudo-likelihood approches for two real-world prediction problems. Recently, the algorithm in [21], outperforming the voted perceptron became the state-of-the-art method for discriminative weight learning of MLNs. However, both discriminative approches to MLNs learn weights for a fixed structure, given by a domain expert or learned through another structure learning method (usually generative). Better results could be achieved if the structure could be learned in a discriminative fashion. Unfortunately, the computational cost of optimizing structure and parameters for conditional likelihood is prohibitive. In this paper we show that the simple approximation of choosing structures by maximizing conditional likelihood while setting parameters by maximum likelihood can produce better results in terms of predictive accuracy. Structures are scored through a very fast inference algorithm MC-SAT [27] whose lazy version Lazy-MC-SAT [28] greatly reduces memory requirements, while parameters are learned through a quasi-Newton optimization method like L-BFGS that has been found to be much faster [34] than iterative scaling initially used for Markov Networks' weight learning [5]. We show through experiments in two real-world domains that the proposed algorithm improves over the state-of-the-art algorithm of [21] in terms of conditional likelihood of the query predicates.

Discriminative approaches may not always provide the highest classification accuracy. An empirical and theoretical comparison of discriminative and generative classifiers (logistic regression and Naïve Bayes (NB)) is given in [22]. It is shown that for small sample sizes the generative NB classier can outperform a discriminatively trained model. This is consistent with the fact that, for the same representation, discriminative training has lower bias and higher variance than generative training, and the variance term dominates at small sample sizes [8, 9]. For the dataset sizes typically found in practice, however, the results in [12, 22, 11] all support the choice of discriminative training. An experimental comparison of discriminative and generative parameter training on both discriminatively and generatively structured Bayesian Network classifiers has been performed in [25]. In this paper we perform an experimental comparison between generative and discriminative structure learning algorithms for MLNs and confirm the results in [22] in the case of MLNs by showing that on a small dataset the generative algorithm is competitive, while on a larger dataset the discriminative algorithm outperforms the generative one in terms of conditional likelihood.

The paper is organized as follows: in Section 2 we introduce MNs and MLNs, in Section 3 we describe existing generative structure learning and discriminative weight learning approaches for MLNs, Section 4 introduces the iterated local search metaheuristic, Section 5 describes a discriminative algorithm for MLNs structure learning, Section 6 presents the experiments, Section 7 describes related work and we conclude in Section 8;

2 Markov Networks and Markov Logic Networks

A Markov network (also known as Markov random field) is a model for the joint distribution of a set of variables $\mathrm{X} = (\mathrm{X}_1,\mathrm{X}_2,\ldots,\mathrm{X}_n) \in \chi$ [5]. It is composed of an undirected graph G and a set of potential functions. The graph has a node for each variable, and the model has a potential function ϕ_k for each clique in the graph. A potential function is a non-negative real-valued function of the state of the corresponding clique. The joint distribution represented by a Markov network is given by:

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$$

where $x_{\{k\}}$ is the state of the kth clique (i.e., the state of the variables that appear in that clique). Z, known as the partition function, is given by:

$$Z = \sum_{x \in \chi} \prod_k \phi_k(x_{\{k\}})$$

Markov networks are often conveniently represented as log-linear models, with each clique potential replaced by an exponentiated weighted sum of features of the state, leading to:

$$P(X = x) = \frac{1}{Z} \exp(\sum_j w_j f_j(x))$$

A feature may be any real-valued function of the state. We will focus on binary features, $f_j \in \{0, 1\}$. In the most direct translation from the potential-function form, there is one feature corresponding to each possible state x_k of each clique, with its weight being $\log(\phi(x_{\{k\}})$. This representation is exponential in the size of the cliques. However a much smaller number of features (e.g., logical functions of the state of the clique) can be specified, allowing for a more compact representation than the potential-function form, particularly when large cliques are present. MLNs take advantage of this.

A first-order KB can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea in Markov logic is to soften these constraints: when a world violates one formula in the KB it is less probable, but not impossible. The fewer formulas a world violates, the more probable it is. Each formula has an associated weight that reflects how strong a constraint it is: the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not, other things being equal.

A Markov logic network [30] L is a set of pairs $(F_i; w_i)$, where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_p\}$ it defines a Markov network $M_{L;C}$ as follows:

1. $M_{L;C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true, and 0 otherwise.

2. $M_{L;C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L. Thus there is an edge between two nodes of $M_{L;C}$ iff the corresponding ground predicates appear together in at least one grounding of one formula in L. An MLN can be viewed as a template for constructing Markov networks. The probability distribution over possible worlds x specified by the ground Markov network $M_{L;C}$ is given by

$$P(X = x) = \frac{1}{Z} \exp(\sum_{i=1}^{F} w_i n_i(x))$$

where F is the number of formulas in the MLN and $n_i(x)$ is the number of true groundings of F_i in x. As formula weights increase, an MLN increasingly resembles a purely logical KB, becoming equivalent to one in the limit of all infinite weights.

In this paper we focus on MLNs whose formulas are function-free clauses and assume domain closure (it has been proven that no expressiveness is lost), ensuring that the Markov networks generated are finite. In this case, the groundings of a formula are formed simply by replacing its variables with constants in all possible ways.

3 Structure and Parameter Learning of MLNs

3.1 Generative Structure Learning of MLNs

One of the approaches for learning Markov Network weights is iterative scaling [5]. However, maximizing the likelihood (or posterior) using a quasi-Newton optimization method like L-BFGS has recently been found to be much faster [34]. Regarding structure learning, the authors in [5] induce conjunctive features by starting with a set of atomic features (the original variables), conjoining each current feature with each atomic feature, adding to the network the conjunction that most increases likelihood, and repeating. The work in [23] extends this to the case of conditional random fields, which are Markov networks trained to maximize the conditional likelihood of a set of outputs given a set of inputs.

The first attempt to learn MLNs was that in [30], where the authors used CLAUDIEN [6] to learn the clauses of MLNs and then learned the weights by maximizing pseudo-likelihood. In [14] another method was proposed that combines ideas from ILP and feature induction of Markov networks. This algorithm, that performs a beam or shortest first search in the space of clauses guided by a weighted pseudo-log-likelihood (WPLL) measure [1], outperformed that of [30]. Recently, in [24] a bottom-up approach was proposed in order to reduce the search space. This algorithm uses a propositional Markov network learning method to construct template networks that guide the construction of candidate clauses. In this way, it generates fewer candidates for evaluation. For every candidate structure, in both [14, 24] the parameters that optimize the WPLL are set through L-BFGS that approximates the second-derivative of the WPLL by keeping a running finite-sized window of previous first-derivatives.

3.2 Discriminative Parameter Learning of MLNs

Learning MLNs in a discriminative fashion has produced for predictive tasks much better results than generative approaches as the results in [31] show. In this work the voted-perceptron algorithm was generalized to arbitrary MLNs by replacing the Viterbi algorithm with a weighted satisfiability solver. The new algorithm is essentially gradient descent with an MPE approximation to the expected sufficient statistics (true clause counts) and these can vary widely between clauses, causing the learning problem to be highly ill-conditioned, and making gradient descent very slow. In [21] a preconditioned scaled conjugate gradient approach is shown to outperform the algorithm in [31] in terms of learning time and prediction accuracy. This algorithm is based on the scaled conjugate gradient method and very good results are obtained with a simple approach: per-weight learning weights, with the weight's learning rate being the global one divided by the corresponding clause's empirical number of true groundings.

However, for both these algorithms the structure is supposed to be given by an expert or learned previously and they focus only on the parameter learning task. This can lead to suboptimal results if the clauses given by an expert do not capture the essential dependencies in the domain in order to improve classification accuracy. On the other side, since to the best of our knowledge, no attempt has been made to learn the structure of MLNs discriminatively, the clauses learned by generative structure learning algorithms tend to optimize the joint distribution of all the variables and applying discriminative weight learning after the structure has been learned generatively may lead to suboptimal results since the initial goal of the learned structure was not to discriminate query predicates.

4 Iterated Local Search

Many widely known and high-performance local search algorithms make use of randomized choice in generating or selecting candidate solutions for a given combinatorial problem instance. These algorithms are called stochastic local search (SLS) algorithms [13] and represent one of the most successful and widely used approaches for solving hard combinatorial problem. Many "simple" SLS methods come from other search methods by just randomizing the selection of the candidates during search, such as Randomized Iterative Improvement (RII), Uniformed Random Walk, etc. Many other SLS methods combine "simple" SLS methods to exploit the abilities of each of these during search. These are known as Hybrid SLS methods [13]. ILS is one of these metaheuristics because it can be easily combined with other SLS methods.

One of the simplest and most intuitive ideas for addressing the fundamental issue of escaping local optima is to use two types of SLS steps: one for reaching local optima as efficiently as possible, and the other for effectively escaping local optima. ILS methods [13, 20] exploit this key idea, and essentially use two types of search steps alternatingly to perform a walk in the space of local optima w.r.t the given evaluation function. The algorithm works as follows: The search process starts from a randomly selected element of the search space. From this initial candidate solution, a locally optimal solution is obtained by applying a subsidiary local search procedure. Then each iteration step of the algorithm consists of three major steps: first a perturbation method is applied to the current candidate solution s; this yields a modified candidate solution s' from which in the next step a subsidiary local search is performed until a local optimum s'' is obtained. In the last third step, an acceptance criterion is used to decide from which of the two local optima s or s' the search process is continued. The algorithm can terminate after some steps have not produced improvement or simply after a certain number of steps. The choice of the components of the ILS has a great impact on the performance of the algorithm.

In general, it is not straightforward to decide whether to use a systematic or SLS algorithm in a certain task. Systematic and SLS algorithms can be considered complementary to each other. SLS algorithms are advantageous in many situations, particularly if reasonably good solutions are required within a short time, if parallel processing is used and if knowledge about the problem domain is rather limited. In other cases, when time constraints are less important and some knowledge about the problem domain can be exploited, systematic search may be a better choice. Structure learning of MLNs is a hard optimization problem due to the large space to be explored, thus SLS methods are suitable for finding solutions of high quality in short time. Moreover, one of the key advantages of SLS methods is that they can greatly speed up learning through parallel processing, where speedups proportional to the number of CPUs can be achieved [13].

5 Discriminative Structure Learning of MLNs

In this section we describe our proposal for tailoring ILS metaheuristic to the problem of learning the structure of MLNs and describe how weights are set and how structures are scored. The approach we follow is similar to [12] where Bayesian Networks were learned by setting weights through maximum likelihood and choosing structures by maximizing conditional likelihood.

5.1 Search Strategy

Algorithm 1 (Discriminative Structure Learning - DSL) iteratively adds the best clause to the current MLN until two consecutive steps have not produced improvement (however other stopping criteria could be applied). It can start from an empty network or from an existing KB. Like in [30, 14] we add all unit clauses (single predicates) to the MLN. The initial weights are learned in *LearnWeights* through L-BFGS and the initial structure is scored in *ComputeCLL* through MC-SAT. The search for the best clause is performed in *SearchBestClause* described by Algorithm 2. The algorithm performs an iterated local search to find the best clause to add to the MLN. It starts by randomly choosing a unit clause CL_C in the search space. Then it performs a greedy local search to efficiently reach a local optimum CL_S. At this point, a perturbation method is applied leading to the neighbor CL'_C of CL_S and then a greedy local search is applied to CL'_C to reach another local optimum CL'_S . The *accept* function decides whether the search must continue from the previous local optimum CL_C or from the last found local optimum CL'_S (*accept* can perform random walk or iterative improvement in the space of local optima).

Careful choice of the various components of Algorithm 2 is important to achieve high performance. The clause perturbation operator (flipping the sign of literals, removing literals or adding literals) has the goal to jump in a different region of the search space where search should start with the next iteration. There can be strong or weak perturbations which means that if the jump in the search space is near to the current local optimum the subsidiary local search procedure *LocalSearch*$_{II}$ (Algorithm 3) may fall again in the same local optimum and enter regions with the same value of the objective function called *plateau*, but if the jump is too far, *LocalSearch*$_{II}$ may take too many steps to reach another good solution. In our algorithm we use only strong perturbations, i.e., we always re-start from unit clauses (in future work we intend to dynamically adapt the nature of the perturbation). Regarding the procedure *LocalSearch*$_{II}$, we decided to use an iterative improvement approach (the walk probability is set to zero and the best clause is always chosen in *step*$_{II}$) in order to balance intensification (greedily increase solution quality by exploiting the evaluation function) and diversification (randomness induced by strong perturbation to avoid search

Algorithm 1 Discriminative Structure Learning

```
Input: P:set of predicates, MLN:Markov Logic Network, RDB:Relational Database
CLS = All clauses in MLN $\cup$ P;
LearnWeights(MLN,RDB);
BestScore = ComputeCLL(MLN,RDB);
repeat
  BestClause = SearchBestClause(P,MLN,BestScore,CLS,RDB);
  if BestClause $\neq$ null then
      Add BestClause to MLN;
      BestScore = ComputeCLL(MLN,RDB);
  end if
until BestClause = null for two consecutive steps
return MLN
```

stagnation). In future work we intend to further weaken intensification by using a higher walk probability. Finally, the *accept* function always accepts the best solution found so far.

5.2 Setting Parameters through Maximum Likelihood

For every candidate structure, the parameters that optimize the WPLL are set through L-BFGS. As pointed out in [14] a potentially serious problem that arises when evaluating candidate clauses using WPLL is that the optimal (maximum WPLL) weights need to be computed for each candidate. Since this involves numerical optimization, and needs to be done millions of times, it could easily make the algorithm too slow. In [23, 5] the problem is addressed by assuming that the weights of previous features do not change when testing a new one. Surprisingly, the authors in [14] found this to be unnecessary if the very simple approach of initializing L-BFGS with the current weights (and zero weight for a new clause) is used. Although in principle all weights could change as the result of introducing or modifying a clause, in practice this is very rare. Second-order, quadratic-convergence methods like L-BFGS are known to be very fast if started near the optimum [34]. This is what happened in [14]: L-BFGS typically converges in just a few iterations, sometimes one. We use the same approach for setting the parameters that optimize the WPLL.

5.3 Efficient Structure Scoring

In order to score MLN structures, we need to perform inference over the network. A very fast algorithm for inference in MLNs is MC-SAT [27]. Since probabilistic inference methods like MCMC or belief propagation tend to give poor results when deterministic or near-deterministic dependencies are present, and logical ones like satisfiability testing are inapplicable to probabilistic dependencies, MC-SAT combines ideas from both MCMC and satisfiability to handle probabilistic, deterministic and near-deterministic dependencies that are typical of statistical

Algorithm 2 SearchBestClause

```
Input: P: set of predicates, MLN: Markov Logic Network, BestScore: current best
score, CLS: List of clauses, RDB: Relational Database)
$CL_C$ = Random Pick a clause in CLS $\cup$ P;
$CL_S$ = $LocalSearch_{II}(CL_S)$;
BestClause = $CL_S$;
repeat
  $CL'_C$ = $Perturb(CL_S)$;
  $CL'_S$ = $LocalSearch_{II}(CL'_C$,MLN,BestScore);
  if ComputeCLL(BestClause,MLN,RDB) $\leq$ ComputeCLL($CL'_S$,MLN,RDB)
  then
    BestClause = $CL'_S$;
    Add BestClause to MLN;
    BestScore = ComputeCLL($CL'_S$,MLN,RDB)
  end if
  $CL_S$ = accept($CL_S$,$CL'_S$);
until two consecutive steps have not produced improvement
Return BestClause
```

relational learning. MC-SAT was shown to greatly outperform Gibbs sampling and simulated tempering in two real-world datasets regarding entity resolution and collective classification.

Even though MC-SAT is a very fast inference algorithm, scoring candidate structures at each step can be potentially very expensive since inference has to be performed for each candidate clause added to the current structure. One problem that arises is that fully instantiating a finite first-order theory requires memory in the order of the number of constants raised to the length of the clauses, which signicantly limits the size of domains where the problem can still be tractable. To avoid this problem, we used a lazy version of MC-SAT, Lazy-MC-SAT [28] which reduces memory and time by orders of magnitude compared to MC-SAT. Before Lazy-MC-SAT was introduced, the LazySat algorithm [33] was shown to greatly reduce memory requirements by exploiting the sparseness of relational domains (i.e., only a small fraction of ground atoms are true, and most clauses are trivially satisfied). The authors in [28] generalize the ideas in [33] by proposing a general method for applying lazy inference to a broad class of algorithms such as other SAT solvers or MCMC methods. Another problem is that even though Lazy-MC-SAT makes memory requirements tractable, it can take too much time to construct the Markov random field in the first step of MC-SAT for every candidate structure.

To make the execution of Lazy-MC-SAT tractable for every candidate structure, we use the following simple heuristics: 1) We score through Lazy-MC-SAT only those candidates that produce an improvement in WPLL. Once the parameters are set through L-BFGS, it is straightforward to compute the gain in WPLL for each candidate. This reduces the number of candidates to be scored through Lazy-MC-SAT for a gain in CLL. 2) We pose a memory limit for Lazy-MC-SAT on the clause activation phase and this greatly speeds up the whole inference

Algorithm 3 LocalSearch$_{II}$

```
Input: (CL$_C$: current clause)
wp: walk probability, the probability of performing an improvement step or a
random step
repeat
   NBHD = Neighborhood of CL$_C$ constructed using the clause construction operators;
   CL$_S$ = $Step_{RII}$(CL$_C$,NBHD,wp);
   CL$_C$ = CL$_S$;
until two consecutive steps do not produce improvement
Return CL$_S$;

$Step_{RII}$(CL$_C$,NBHD,wp)
U = random(]0,1]); random number using a Uniform Probability Distribution
if ($U \leq wp$) then
   CL$_S$ = step$_{URW}$(CL$_C$,NBHD)
   Uninformed Random Walk: randomly choose a neighbor from NBHD
else
   CL$_S$ = step$_{II}$(CL$_C$,NBHD)
   Iterative Improvement: choose the best among the improving neighbours in NBHD.
   If there is no improving neighbor choose the minimally worsening one
end if
Return CL$_S$
```

task. Although in principle this limit can reduce the accuracy of inference, we found that in most cases the memory limit is never reached making the overall inference task very fast. 3) We pose a time limit in the clause activation phase in order to avoid those rare cases where the step takes a very long time to be completed. For most candidate structures such a time limit is never reached and for those rare cases where time limit is reached, inference is performed using the activated clauses within the limit.

We found that these simple approximations greatly speed up the scoring of each structure at each step. Filtering the potential candidates through the gain in WPLL can in principle exclude good candidates due to the mismatch between the optimization of WPLL and that of CLL. However, we empirically found that most candidates not improving WPLL, did not improve CLL. Further investigation on this issue may help to select better or more candidates to be scored through Lazy-MC-SAT.

6 Experiments

Through experimental evaluation we want to answer the following questions:

(Q1) Is the DSL algorithm competitive with state-of-the-art discriminative training algorithms of MLNs?

(Q2) Is the DSL algorithm competitive with the state-of-the-art generative algorithm for structure learning of MLNs?

(Q3) Is the DSL algorithm competitive with pure probabilistic approaches such as Naïve Bayes and Bayesian Networks?

(Q4) Is the DSL algorithm competitive with state-of-the-art ILP systems for the task of structure learning of MLNs?

(Q5) Does the DSL algorithm always perform better than BUSL for classification tasks? If not, are there any regimes in which each algorithm performs better?

Regarding question **(Q1)** we have to compare DSL with Preconditioned Scaled Conjugate Gradient (PSCG), the state-of-the-art discriminative training algorithm for MLNs proposed in [21]. It must be noted that this algorithm takes in input a fixed structure, and with the clausal knowledge base we use in our experiments for Cora (each dataset comes with a hand-coded knowledge base), PSCG has achieved the best published results. We also exclude the approach of adapting the rule set and then learning weights with PSCG, since it would be computationally very expensive.

To answer question **(Q2)** we have to perform experimental comparison with the Bottom-Up Structure Learning (BUSL) algorithm [24] which is the state-of-the-art algorithm for this task. Since in principle, the MLNs structure can be learned using any ILP technique it would be interesting to know how the DSL compares to ILP approaches. In [14], the proposed algorithm based on beam search (BS) was shown to outperform FOIL and the state-of-the-art ILP system Aleph for the task of learning MLNs structure. Moreover, BS outperformed both Naïve Bayes and Bayesian Networks in terms of CLL and AUC. Since in [24] was shown that BUSL outperforms the BS algorithm of [14], our baseline for questions **(Q2)**, **(Q3)** and **(Q4)** is again BUSL. Regarding question **(Q5)**, we compare DSL and BUSL on two datasets with the goal of discovering regimes in which each one can perform better. We will use two datasets, one of which can be considered of small size and the other one of much larger size.

6.1 Datasets

We carried out experiments on two publicly-available databases: the UW-CSE database (available at http://alchemy.cs.washington.edu/data/uw-cse) used by [14, 30, 24] and the Cora dataset originally labeled by Andrew McCallum. Both represent standard relational datasets and are used for two important relational tasks: Cora for entity resolution and UW-CSE for social network analysis. For Cora we used a cleaned version from [32], with five splits for cross-validation.

The published UW-CSE dataset consists of 15 predicates divided into 10 types. Types include: publication, person, course, etc. Predicates include: Student(person), Professor(person), AdvisedBy(person1, person2),TaughtBy(course, person, quarter), Publication (paper, person) etc. The dataset contains a total of 2673 tuples (true ground atoms, with the remainder assumed false). We used the hand-coded knowledge base provided with it, which includes 94 formulas stating regularities like: each student has at most one advisor; if a student is an author of a paper, so is her advisor; etc. Notice that these statements are not always true, but are typically true. The task is to predict who is whose advisor from information about coauthorships, classes taught, etc. More precisely, the query atoms are all groundings of AdvisedBy(person1, person2), and the evidence atoms are all groundings of all other predicates except Student(person) and Professor(person), corresponding to the Partial Information scenario in [30].

The Cora dataset consists of 1295 citations of 132 different computer science papers, drawn from the Cora Computer Science Research Paper Engine. The task is to predict which citations refer to the same paper, given the words in their author, title, and venue fields. The labeled data also specify which pairs of author, title, and venue fields refer to the same entities. We performed experiments for each field in order to evaluate the ability of the model to deduplicate fields as well as citations. Since the number of possible equivalences is very large, like the authors did in [21] we used the canopies found in [32] to make this problem tractable. The dataset contains a total of 70367 tuples (true and false ground atoms, with the remainder assumed false).

6.2 Systems and Methodology

We implemented the DSL algorithm in the Alchemy package [15]. We used the implementation of L-BFGS and Lazy-MC-SAT in Alchemy to learn maximum WPLL weights and compute CLL during clause search.

Regarding parameter learning, we compared our algorithm performance with the state-of-the-art algorithm PSCG of [21] for discriminative weight learning of MLNs. This algorithm takes as input an MLN and the evidence (groundings of non-query predicates) and discriminatively trains the MLN to optimize the CLL of the query predicates given evidence. DSL and PSCG both optimize the CLL of the query predicates and a comparison between these algorithms would be useful to understand if learning automatically the clauses from scratch can improve over hand-coded MLN structures in terms of classification accuracy of the query predicates given evidence. We performed all the experiments on a 2.13 GHz Intel Core2 Duo CPU. For the UW-CSE dataset we trained PSCG on the hand-coded knowledge base provided with the dataset. We used the implementation of PSCG in the Alchemy package and ran this algorithm with the default parameters for 10 hours. For the Cora dataset, for PSCG we report the results obtained in [21]. For both datasets, for the DSL algorithm we used the following parameters: The mean and variance of the Gaussian prior were set to 0 and 100, respectively; maximum variables per clause = 4; maximum predicates per clause = 4; penalization of weighted pseudo-likelihood = 0.01 for UW-CSE and 0.001 for Cora. For L-BFGS we used the following parameters: maximum iterations = 10,000 (tight) and 10 (loose); convergence threshold = 10^{-5} (tight) and 10^{-4} (loose). For Lazy-MC-SAT during learning we used the following parameters: memory limit = 200MB, maximum number of steps for Gibbs sampling = 100, simulated annealing temperature = 0.5.

Regarding BUSL, for both datasets, we used the following parameters: The mean and variance of the Gaussian prior were set to 0 and 100, respectively;

maximum variables per clause: 5 for UW-CSE and 6 for Cora; maximum predicates per clause = 6; penalization of WPLL: 0.01 for UW-CSE and 0.001 for Cora. minWeight 0.5 for UW-CSE and 0.01 for Cora; For L-BFGS we used the following parameters: maximum iterations = 10,000 (tight) and 10 (loose); convergence threshold = 10^{-5} (tight) and 10^{-4} (loose).

In the UW-CSE domain, we used the same leave-one-area-out methodology as in [30]. In the Cora domain, we performed 5-fold cross-validation. For each train/test split, one of the training folds is used as tuning set for computing the CLL. For each system on each test set, we measured the CLL and the area under the precision-recall curve (AUC) for the query predicates. The advantage of the CLL is that it directly measures the quality of the probability estimates produced. The advantage of the AUC is that it is insensitive to the large number of true negatives (i.e., ground atoms that are false and predicted to be false). The CLL of a query predicate is the average over all its groundings of the ground atoms log-probability given evidence. The precision-recall curve for a predicate is computed by varying the CLL threshold above which a ground atom is predicted to be true; i.e. the predicates whose probability of being true is greater than the threshold are positive and the rest are negative. For the computation of AUC we used the package of [3].

It must be noted also that since the goal of the experimental study was to verify that DSL is competitive to other state-of-the-art techniques, and not to boost performance, we did not try to optimize any parameter.

6.3 Results

After learning the structure discriminatively, we performed inference on the test fold for both datasets by using MC-SAT with number of steps = 10000 and simulated annealing temperature = 0.5. For each experiment, on the test fold all the groundings of the query predicates were commented: advisedBy for the UW-CSE dataset (professor and student are also commented) and sameBib, sameTitle, sameAuthor and sameVenue for Cora. MC-SAT produces probability outputs for every grounding of the query predicate on the test fold. We used these values to compute the average CLL over all the groundings.

The results for all algorithms on the UW-CSE dataset are reported in Table 1 where CLL is averaged over all the groundings of the predicate advisedBy in the test fold. Regarding DSL and PSCG, in this domain DSL performs better in terms of CLL in all folds of the dataset and in two folds in terms of AUC. Overall, DSL performs better in terms of CLL and worse in terms of AUC. For the Cora dataset the results are reported in Table 2 where for each query predicate we report the average of CLL of its groundings over the test fold (for each predicate, training is performed on four folds and testing on the remaining one in a 5-fold cross-validation). For Cora, DSL performs better in terms of CLL but worse in terms of AUC for all the query predicates. We observed empirically on each fold that the performances in terms of CLL and AUC were always balanced, a slightly better performance in CLL always resulted in a slightly worse performance in terms of AUC and vice versa. Since CLL determines the quality of the probability

Table 1. Results for DSL, PSCG and BUSL for the query predicate advisedBy in the UW-CSE domain

	DSL		PSCG		BUSL	
area	CLL	AUC	CLL	AUC	CLL	AUC
language	-0.036±0.012	0.032	-0.049±0.016	0.011	-0.024±0.008	0.115
graphics	-0.016±0.002	0.009	-0.023±0.005	0.005	-0.014±0.002	0.007
systems	-0.021±0.004	0.023	-0.026±0.005	0.069	-0.295±0.000	0.007
theory	-0.019±0.005	0.031	-0.028±0.007	0.101	-0.013±0.003	0.032
ai	-0.021±0.002	0.014	-0.032±0.005	0.034	-0.019±0.003	0.013
Overall	-0.023±0.005	0.022	-0.032±0.008	0.044	-0.073±0.003	0.035

Table 2. Results for DSL, PSCG and BUSL for all query predicates in the Cora domain

	DSL		PSCG		BUSL	
predicate	CLL	AUC	CLL	AUC	CLL	AUC
sameBib	-0.116±0.001	0.495	-0.291±0.003	0.990	-0.566±0.001	0.138
sameTitle	-0.076±0.005	0.430	-0.231±0.014	0.953	-0.100±0.004	0.419
sameAuthor	-0.126±0.007	0.590	-0.182±0.013	0.999	-0.834±0.009	0.323
sameVenue	-0.100±0.003	0.247	-0.444±0.012	0.823	-0.232±0.005	0.218
Overall	-0.105±0.004	0.441	-0.287±0.000	0.941	-0.433±0.005	0.275

predictions output by the algorithm, DSL outperforms PSCG in terms of the ability to predict correctly the query predicates given evidence. However, since AUC is useful to predict the few positives in the data, PSCG produces better results for only positive examples. Hence, these results answer question **(Q1)**. The worse performance of DSL in terms of AUC can be due to the approach of DSL of optimizing the conditional likelihood during structure learning that may lead to overfitting. This issue deserves further investigation. However, it must be noted that PSCG has achieved the best published results on Cora in terms of AUC [21]. Moreover, since we did not try to optimize parameters, it is interesting to know if the results would change with a higher number of variables and literals per clause. For DSL, we plan to perform more extensive experiments with larger search parameters in order to boost performance.

Regarding DSL against BUSL, the results show that DSL performs better than BUSL in terms of CLL on both datasets. It must be noted, however, that for UW-CSE, BUSL performed generally better than DSL, but produced very low results in one fold. In terms of AUC, BUSL performs slightly better on the UW-CSE dataset while in the Cora dataset DSL outperforms BUSL. Therefore, questions **(Q2)**, **(Q3)** and **(Q4)** can be answered affirmatively. DSL is competitive with BUSL even though for BUSL, in the UW-CSE domain, we used optimized parameters taken from [24] in terms of number of variables and literals per clause, while for DSL we did not optimize any parameter.

Regarding question **(Q5)**, the goal was whether previous results of [22] carry on to MLNs, that on small datasets generative approaches can perform better than discriminative ones. The UW-CSE dataset with a total of 2673 tuples can be considered of much smaller size compared to Cora that has 70367 tuples. The results of Table 1 show that on the UW-CSE dataset, the generative algorithm

BUSL performs better in terms of AUC and is competitive in terms of CLL since it underperforms DSL only because of the low results in the *systems* fold of the dataset. Thus we can answer question **(Q5)** confirming the results in [22] that on small datasets generative approaches can perform better than discriminative ones, while for larger datasets discriminative approches outperform generative ones.

Finally, we give examples of clauses from MLN structures learned for both datasets (we omit the relative weights). For the UW-CSE dataset examples of learned clauses are:

$$\mathit{advisedBy}(a1, a2) \vee \mathit{inPhase}(a1, a3) \vee \mathit{inPhase}(a2, a4)$$

$$\neg \mathit{student}(a1) \vee \mathit{position}(a2, a3) \vee \mathit{advisedBy}(a1, a2)$$

These clauses model the relation *advisedBy* between students and professors. In the first clause, *a1* and *a2* are variables that denote persons (students or professors) and the predicate *inPhase* states for each of these persons the phase of their university career. In the second clause, the predicate *position* relates the person denoted by *a2* (only professors have a position) to his university position.

For Cora, examples of learned clauses are the following:

$\mathit{sameAuthor(a1,a2)} \vee \neg \mathit{hasWordAuthor(a1,a3)} \vee \neg \mathit{hasWordAuthor(a2,a3)} \vee a1 = a2$

$$\neg \mathit{title}(a1, a2) \vee \neg \mathit{title}(a3, a2) \vee \neg \mathit{sameBib}(a3, a1)$$

In the first clause, *a1* and *a2* denote author fields while the predicate *has-WordAuthor* relates author fields to words contained in these fields. In the second rule the predicate *title* relates titles to their respective citations and the predicate *sameBib* is true if both its arguments denote the same citation.

7 Related Work

Many works in the SRL or PILP area have addressed classification tasks. Our discriminative method falls among those approaches that tightly integrate ILP and statistical learning in a single step for structure learning. The earlier works in this direction are those in [4, 26] that employ statistical models such as maximum entropy modeling in [4] and logistic regression in [26]. These approaches can be computationally very expensive. A simpler approach that integrates FOIL [29] and Naïve Bayes is nFOIL proposed in [17]. This approach interleaves the steps of generating rules and scoring them through CLL. In another work [2] these steps are coupled by scoring the clauses through the improvement in classification accuracy. This algorithm incrementally builds a Bayes net during rule learning and each candidate rule is introduced in the network and scored by whether it improves the performance of the classifier. In a recent approach [19], the kFOIL system integrates ILP and support vector learning. kFOIL constructs the feature space by leveraging FOIL search for a set of relevant clauses. The search is driven by the performance obtained by a support vector machine based on the resulting kernel. The authors showed that kFOIL improves over nFOIL. Recently, in TFOIL [18], Tree Augmented Naïve Bayes, a generalization of Naïve Bayes was integrated with FOIL and it was shown that TFOIL outperforms nFOIL.

The most closely related approach to the DSL algorithm is nFOIL (and TFOIL as an extension) which is the first system in literature to tightly integrate feature construction and Naïve Bayes. Such a dynamic propositionalization was shown to be superior compared to static propositionalization approaches that use Naïve Bayes only to post-process the rule set. The approach is different from ours in that nFOIL selects features and parameters that jointly optimize a probabilistic score on the training set, while DSL maximizes the likelihood on the training data but selects the clauses based on the tuning set. This approach is similar to SAYU [2] that uses the tuning set to compute the score in terms of classification accuracy or AUC, with the difference that DSL uses CLL as score instead of AUC. Another difference with nFOIL and SAYU is that DSL, to perform inference for the computation of CLL, uses MC-SAT that is able to handle probabilistic, deterministic and near-deterministic dependencies that are typical of statistical relational learning. Moreover, the lazy version Lazy-MC-SAT reduces memory and time by orders of magnitude as the results in [28] show. This makes it possible to apply the proposed algorithm to very large domains.

8 Conclusions and Future Work

Markov Logic Networks are a powerful representation that combines first-order logic and probability by attaching weights to first-order formulas and viewing these as templates for features of Markov networks. In this paper we have introduced an algorithm that learns discriminatively first-order clauses and their weights. The algorithm scores the candidate structures by maximizing conditional likelihood while setting the parameters by maximum pseudo-likelihood. Empirical evaluation with real-world data in two domains show the promise of our approach improving over the state-of-the-art discriminative weight learning algorithm for MLNs in terms of conditional log-likelihood of the query predicates given evidence. We have also compared the proposed algorithm with the state-of-the-art generative structure learning algorithm and shown that on small datasets the generative approach is competitive, while on larger datasets the discriminative approach outperforms the generative one.

Directions for future work include speeding up the counting of the number of true groundings of a first-order clause which is the most expensive step for parameters' setting; learning the structure discriminatively and then applying a weight learning algorithm such as PSCG; developing methods to avoid overfitting; studying the relationship between the performance in terms of CLL and AUC and try to improve the accuracy in predicting the positives; scoring structures through AUC instead of CLL; adapting dynamically the nature of perturbation in the iterated local search procedure; performing more experiments with optimized parameters regarding the number of variables and literals per clause; finding heuristics to select, among clauses that do not improve WPLL, potential candidates that can improve CLL.

Acknowledgements We would like to thank Pedro Domingos for helpful discussions, Daniel Lowd for help on PSCG, Marc Sumner for help on Alchemy and Lilyana Mihalkova per help on BUSL. We also thank the anonymous reviewers for their comments.

References

1. Besag, J. Statistical analysis of non-lattice data. *Statistician*, 24:179–195, 1975.
2. Davis, J., Burnside, E., de Castro Dutra, I., Page, D., and Santos Costa, Vito. An integrated approach to learning Bayesian networks of rules. In Proc. 16th European Conference on Machine Learning, volume 3720 of LNCS, pages 84–95, Springer, 2005.
3. Davis, J., and Goadrich, M. The relationship between precision-recall and ROC curves. Proc. of 23rd Intl. Conf. on Machine Learning, pages 233–240, 2006.
4. Dehaspe, L. Maximum entropy modeling with clausal constraints. In *Proc. of ILP-97*, pages 109–124,1997.
5. Della Pietra, S., Pietra, V. D., and Laferty, J. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:380–392, 1997.
6. De Raedt, L., and Dehaspe, L. Clausal discovery. *Machine Learning*, 26:99–146, 1997.
7. De Raedt, Luc., Frasconi, Paolo., Kersting, Kristian., and Muggleton, Stephen. *Probabilistic Inductive Logic Programming - Theory and Applications.* Springer 2008.
8. Domingos, P., and Pazzani, M. On the optimality of the simple Bayesian classier under zero-one loss. *Machine Learning*, 29:103–130, 1997.
9. Friedman, J. H. On bias, variance, 0/1 - loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, 1:55–77, 1997.
10. Getoor, L., and Taskar, B. *Introduction to statistical relational learning.* MIT Press, 2007.
11. Greiner, R., Su, X., Shen, S., and Zhou, W. Structural extension to logistic regression: Discriminative parameter learning of belief net classiers. *Machine Learning*, 59:297–322, 2005.
12. Grossman, D., and Domingos, P. Learning bayesian network classiers by maximizing conditional likelihood. In *Proc. 21st Int'l Conf. on Machine Learning*, pages 361–368, Banf, Canada: ACM Press, 2004.
13. Hoos, H. H., and Stutzle, T. *Stochastic local search: Foundations and applications.* Morgan Kaufmann, San Francisco, 2005.
14. Kok, S., and Domingos, P. Learning the structure of markov logic networks. In *Proc, 22nd Int'l Conf. on Machine Learning*, pages 441–448, 2005.
15. Kok, S., Singla, P., Richardson, M., and Domingos, P.: The alchemy system for statistical relational ai (Technical Report). Department of Computer Science and Engineering, University of Washington, Seattle, WA, http://alchemy.cs.washington.edu/, 2005.

16. Laferty, J., McCallum, A., and Pereira, F. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th Int'l Conf. on Machine Learning*, pages 282–289, 2001.
17. Landwehr, N., Kersting, K., and De Raedt, L. nFOIL: Integrating Naive Bayes and FOIL. In *Proc. 20th Nat'l Conf. on Articial Intelligence*, pages 795–800, AAAI Press, pages 795–800, 2005.
18. Landwehr, N., Kersting, K., and De Raedt, L. Integrating Naive Bayes and FOIL. In: *Journal of Machine Learning Research*, pp. 481-507. 2007.
19. Landwehr, N., Passerini, A., De Raedt L., and Frasconi, P. kFOIL: Learning Simple Relational Kernels. In *Proc. 21st Nat'l Conf. on Articial Intelligence*, AAAI Press, 2006.
20. Loureno, H.R., Martin, O., and Stutzle, T. Iterated local search. In *Handbook of Metaheuristics*, pages 321–353, Kluwer Academic Publishers, 2002.
21. Lowd, D., and Domingos, P. Efficient weight learning for markov logic networks. In *Proc. of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages, 200–211, 2007.
22. Ng, A. Y. and Jordan, M. I. On discriminative vs. generative: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems*, 14, Cambridge, MA: MIT Press, 841–848, 2002.
23. McCallum, A. Efficiently inducing features of conditional random fields. In *Proc. 19th Conf. on Uncertainty in Artificial Intelligence*, pages 403–410, 2003.
24. Mihalkova, L., and Mooney, R. J. Bottom-up learning of markov logic network structure. In *Proc. 24th Int'l Conf. on Machine Learning*, pages 625–632, 2007.
25. Pernkopf, F., and Bilmes, J. Discriminative versus generative parameter and structure learning of Bayesian network classiers. In *Proc, 22nd Int'l Conf. on Machine Learning*, pages 657–664, 2005.
26. Popescul, A., Ungar, L., Lawrence, S., and Pennock, D. Statistical Relational Learning for Document Mining. In *Proc. 3rd Int'l Conf. on Data Mining*, 275–282, 2003.
27. Poon, H., and Domingos, P. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proc. 21st Nat'l Conf. on Articial Intelligence*, pages 458–463, AAAI Press, 2006.
28. Poon, H., Domingos, P., and Sumner, M. A General Method for Reducing the Complexity of Relational Inference and its Application to MCMC. In *Proc. 23rd Nat'l Conf. on Articial Intelligence*, 2008, Chicago, IL: AAAI Press. To appear.
29. Quinlan, J. R. Learning logical denitions from relations. *Machine Learning*, 5:239–266, 1990.
30. Richardson, M., and Domingos, P. Markov logic networks. *Machine Learning*, 62:107–236, 2006.
31. Singla, P., and Domingos, P. Discriminative training of markov logic networks. In *Proc. 20th Nat'l Conf. on Articial Intelligence*, pages 868–873, AAAI Press, 2005.
32. Singla, P., and Domingos, P. Entity resolution with markov logic. In *Proc. 6th Int'l Conf. on Data Mining*, pages 572–582, IEEE Computer Society Press, 2006.
33. Singla, P., and Domingos, P. Memory-efficient inference in relational domains. In *Proc. 21st Nat'l Conf. on Articial Intelligence*, pages 488–493, AAAI Press, 2006.
34. Sha, F., and Pereira, F. Shallow parsing with conditional random fields. In *Proc. HLT-NAACL*, pages 134–141, 2003.