Discriminative Training of Markov Logic Networks

Parag Singla **Pedro Domingos**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
{*parag, pedrod*}*@cs.washington.edu*

Abstract

Many machine learning applications require a combination of probability and first-order logic. Markov logic networks (MLNs) accomplish this by attaching weights to first-order clauses, and viewing these as templates for features of Markov networks. Model parameters (i.e., clause weights) can be learned by maximizing the likelihood of a relational database, but this can be quite costly and lead to suboptimal results for any given prediction task. In this paper we propose a discriminative approach to training MLNs, one which optimizes the conditional likelihood of the query predicates given the evidence ones, rather than the joint likelihood of all predicates. We extend Collins's (2002) voted perceptron algorithm for HMMs to MLNs by replacing the Viterbi algorithm with a weighted satisfiability solver. Experiments on entity resolution and link prediction tasks show the advantages of this approach compared to generative MLN training, as well as compared to purely probabilistic and purely logical approaches.

Introduction

AI systems must be able to learn, reason logically, and handle uncertainty. Research on combining all three has grown rapidly in recent years (Dietterich, Getoor, & Murphy 2003). For example, probabilistic relational models combine Bayesian networks with description logics (Friedman *et al.* 1999), and Bayesian logic programs combine Bayesian networks with inductive logic programming (Kersting & De Raedt 2001). The need to avoid directed cycles in these approaches causes many difficulties, which can be overcome by using Markov networks as the probabilistic representation. Relational Markov networks do this, but are limited in expressiveness to conjunctive database queries, and require exponential space in the size of the cliques (Taskar, Abbeel, & Koller 2002). Markov logic networks (MLNs) overcome these limitations by allowing the features of Markov networks to be specified by arbitrary formulas in finite first-order logic (Richardson & Domingos 2004).

Learning Markov network parameters is difficult because computing the likelihood and its gradient requires performing inference, which has worst-case exponential cost. An oft-used alternative is to optimize instead a pseudo-likelihood measure, which involves only each variable's probability given its Markov blanket (graph neighbors) in the data (Besag 1975). This is the approach used by Richardson and Domingos (2004). However, the pseudo-likelihood ignores non-local interactions between variables, and may underperform when these need to be taken into account at inference time.

Both likelihood and pseudo-likelihood are generative learning approaches in the sense that they attempt to optimize the joint distribution of all variables. In contrast, discriminative approaches maximize the conditional likelihood of a set of outputs given a set of inputs (e.g., Lafferty *et al.* (2001)). This often produces better results, because no effort is spent modeling dependencies between inputs. The voted perceptron is a discriminative algorithm for labeling sequence data, a special case in which tractable inference is possible using the Viterbi algorithm (Collins 2002). In this paper we generalize the voted perceptron to arbitrary Markov logic networks, by replacing the Viterbi algorithm with a weighted satisfiability solver. Experiments in two real-world domains show the promise of this approach.

We begin by briefly reviewing the necessary background in Markov networks, logic, and Markov logic networks. We then describe our algorithm for discriminative learning of MLNs. Finally, we report our experiments.

Markov Networks

A *Markov network* (also known as *Markov random field*) is a model for the joint distribution of a set of variables $X = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$ (Della Pietra, Della Pietra, & Lafferty 1997). It is composed of an undirected graph G and a set of potential functions ϕ_k. The graph has a node for each variable, and the model has a potential function 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

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

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 $Z = \sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$.

Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

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

$$P(X=x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x)\right) \quad (2)$$

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

Maximum *a posteriori* (MAP) inference in Markov networks involves finding the most likely state of a set of query (output) variables given the state of a set of evidence (input) variables, and is NP-hard (Roth 1996). Conditional inference involves computing the distribution of the query variables given the evidence, and is #P-complete (Roth 1996). The most widely used approximate solution to this problem is Markov chain Monte Carlo (MCMC) (Gilks, Richardson, & Spiegelhalter 1996), and in particular Gibbs sampling, which proceeds by sampling each variable in turn given its Markov blanket (i.e., its neighbors in the graph), and counting the fraction of samples that each variable is in each state.

First-Order Logic

A *first-order knowledge base (KB)* is a set of sentences or formulas in first-order logic (Genesereth & Nilsson 1987). Formulas are constructed using four types of symbols: constants, variables, functions, and predicates. Constant symbols represent objects in the domain of interest (e.g., people: `Anna`, `Bob`, `Chris`, etc.). Variable symbols range over the objects in the domain. Function symbols (e.g., `MotherOf`) represent mappings from tuples of objects to objects. Predicate symbols represent relations among objects in the domain (e.g., `Friends`) or attributes of objects (e.g., `Smokes`). A *term* is any expression representing an object in the domain. It can be a constant, a variable, or a function applied to a tuple of terms. For example, `Anna`, `x`, and $\texttt{GreatestCommonDivisor}(\texttt{x}, \texttt{y})$ are terms. An *atomic formula* or *atom* is a predicate symbol applied to a tuple of terms (e.g., $\texttt{Friends}(\texttt{x}, \texttt{MotherOf}(\texttt{Anna}))$). A *ground term* is a term containing no variables. A *ground atom* or *ground predicate* is an atomic formula all of whose arguments are ground terms. Formulas are recursively constructed from atomic formulas using logical connectives and quantifiers. A *positive literal* is an atomic formula; a *negative literal* is a negated atomic formula. A KB in *clausal form* is a conjunction of *clauses*, a clause being a disjunction of literals. A *possible world* or *Herbrand interpretation* assigns a truth value to each possible ground atom. In finite domains, first-order KBs can be *propositionalized* by replacing each universally (existentially) quantified formula with a conjunction (disjunction) of all its groundings.

A central (and NP-complete) problem in logic is that of determining if a KB (usually in clausal form) is *satisfiable*, i.e., if there is an assignment of truth values to ground atoms that makes the KB true. One approach to this problem is stochastic local search, exemplified by the WalkSat solver (Selman, Kautz, & Cohen 1996). Beginning with a random truth assignment, WalkSat repeatedly flips the truth value of either (a) an atom that maximizes the number of satisfied clauses, or (b) a random atom in an unsatisfied clause. WalkSat is able to solve hard instances of satisfiability with hundreds of thousands of variables in minutes. Many first-order problems (e.g., planning) can be solved efficiently by propositionalizing them and applying a satisfiability solver. The *weighted satisfiability* problem is a variant of satisfiability where each clause has an associated weight, and the goal is to maximize the sum of the weights of satisfied clauses. MaxWalkSat is a direct extension of WalkSat to this problem (Kautz, Selman, & Jiang 1997).

Markov Logic Networks

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 MLNs 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.

Definition 1 *(Richardson & Domingos 2004) A Markov logic network 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 $C = \{c_1, c_2, \ldots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ (Equations 1 and 2) 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. From Definition 1 and Equations 1 and 2, the probability distribution over possible worlds x specified by the ground Markov network $M_{L,C}$ is given by

$$P(X=x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x)\right) \quad (3)$$

where F is the number 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 will focus on MLNs whose formulas are function-free clauses and assume domain closure, ensuring that the Markov networks generated are finite (Richardson & Domingos 2004). In this case, the groundings of a formula are formed simply by replacing its variables with constants in all possible ways. For example, if $C = \{\texttt{Anna}, \texttt{Bob}\}$, the formula $\forall \texttt{x}\ \texttt{Smokes(x)} \Rightarrow \texttt{Cancer(x)}$ in the MLN L yields the features $\texttt{Smokes(Anna)} \Rightarrow \texttt{Cancer(Anna)}$ and $\texttt{Smokes(Bob)} \Rightarrow \texttt{Cancer(Bob)}$ in the ground Markov network $M_{L,C}$ (or $\neg\texttt{Smokes(Anna)} \vee \texttt{Cancer(Anna)}$ and $\neg\texttt{Smokes(Bob)} \vee \texttt{Cancer(Bob)}$ in clausal form). See Richardson and Domingos (2004, Table II) for details.

Learning MLNs

Generative Training

MLN weights can be learned generatively by maximizing the likelihood of a relational database (Equation 3). (A closed-world assumption is made, whereby all ground atoms not in the database are assumed false.) The gradient of the log-likelihood with respect to the weights is

$$\frac{\partial}{\partial w_i} \log P_w(X=x) = n_i(x) - \sum_{x'} P_w(X=x')\, n_i(x') \quad (4)$$

where the sum is over all possible databases x', and $P_w(X = x')$ is $P(X = x')$ computed using the current weight vector $w = (w_1, \ldots, w_i, \ldots)$. In other words, the ith component of the gradient is simply the difference between the number of true groundings of the ith formula in the data and its expectation according to the current model. Unfortunately, computing these expectations requires inference over the model, which can be very expensive. Most fast numeric optimization methods (e.g., conjugate gradient with line search, L-BFGS) also require computing the likelihood itself and hence the partition function Z, which is also intractable. Although inference can be done approximately using Markov chain Monte Carlo, Richardson and Domingos (2004) found this to be too slow. Instead, they maximized the pseudo-likelihood of the data, an alternative measure widely used in areas like spatial statistics, social networks and natural language (Besag 1975). If x is a possible world (relational database) and x_l is the lth ground atom's truth value, the pseudo-log-likelihood of x given weights w is

$$\log P^*_w(X=x) = \sum_{l=1}^{n} \log P_w(X_l = x_l | MB_x(X_l)) \quad (5)$$

where $MB_x(X_l)$ is the state of X_l's Markov blanket in the data (i.e., the truth values of the ground atoms it appears in some ground formula with). Computing the pseudo-likelihood and its gradient does not require inference, and is therefore much faster. However, the pseudo-likelihood parameters may lead to poor results when inference across non-neighboring variables is required.

Discriminative Training

In many applications, we know *a priori* which predicates will be evidence and which ones will be queried, and the goal is to correctly predict the latter given the former. If we partition the ground atoms in the domain into a set of evidence atoms X and a set of query atoms Y, the *conditional likelihood* of Y given X is

$$\begin{aligned} P(y|x) &= \frac{1}{Z_x} \exp\left(\sum_{i \in F_Y} w_i n_i(x,y)\right) \\ &= \frac{1}{Z_x} \exp\left(\sum_{j \in G_Y} w_j g_j(x,y)\right) \end{aligned} \quad (6)$$

where F_Y is the set of all MLN clauses with at least one grounding involving a query atom, $n_i(x, y)$ is the number of true groundings of the ith clause involving query atoms, G_Y is the set of ground clauses in $M_{L,C}$ involving query atoms, and $g_j(x, y) = 1$ if the jth ground clause is true in the data and 0 otherwise. When some variables are "hidden" (i.e., neither query nor evidence) the conditional likelihood should be computed by summing them out, but for simplicity we treat all non-evidence variables as query variables. The gradient of the conditional log-likelihood (CLL) is

$$\begin{aligned} \frac{\partial}{\partial w_i} \log P_w(y|x) &= n_i(x,y) - \sum_{y'} P_w(y'|x) n_i(x,y') \\ &= n_i(x,y) - E_w[n_i(x,y)] \end{aligned} \quad (7)$$

As before, computing the expected counts $E_w[n_i(x, y)]$ is intractable. However, they can be approximated by the counts $n_i(x, y^*_w)$ in the MAP state $y^*_w(x)$. This will be a good approximation if most of the probability mass of $P_w(y|x)$ is concentrated around $y^*_w(x)$. Computing the gradient of the CLL now requires only MAP inference to find $y^*_w(x)$, which is much faster than the full conditional inference for $E_w[n_i(x, y)]$. (If the training database is broken up into separate examples, an MAP inference per example is performed.) This approach was used successfully by Collins (2002) in the special case of a Markov network (and hence of an MLN) where the query nodes form a linear chain. In this case, the MAP state can be found in polynomial time using the Viterbi algorithm, a form of dynamic programming (Rabiner 1989). Collins's *voted perceptron* algorithm initializes all weights to zero, performs T iterations of gradient ascent using the approximation above, and returns the parameters averaged over all iterations, $w_i = \sum_{t=1}^{T} w_{i,t}/T$. The parameter averaging helps to combat overfitting. T is chosen using a validation subset of the training data.

Generalizing this solution to arbitrary MLNs requires replacing the Viterbi algorithm with a general-purpose algorithm for MAP inference in MLNs. The form of Equation 6 suggests a solution. Since $y^*_w(x)$ is the state that maximizes the sum of the weights of the satisfied ground clauses, it can be found using the MaxWalkSat solver (see the section on logic). Given an MLN and set of evidence atoms, the KB to be passed to MaxWalkSat is formed by constructing all

groundings of clauses in the MLN involving query atoms, replacing the evidence atoms in those groundings by their truth values, and simplifying. When hidden variables are present, the algorithm of Richardson and Domingos (2004, Table III) is used. (In practice, it is often convenient to make a closed world assumption on the evidence predicates, whereby all ground predicates not explicitly listed in the evidence database are assumed false.)

Unlike the Viterbi algorithm, MaxWalkSat is not guaranteed to find the global MAP state. This is a potential additional source of error in the weight estimates produced. The quality of the estimates can be improved by running a Gibbs sampler starting at the state returned by MaxWalkSat, and averaging counts over the samples. If the $P_w(y|x)$ distribution has more than one mode, doing multiple runs of MaxWalkSat followed by Gibbs sampling may also be useful. In the limit, this is equivalent to computing the expected counts exactly, which gives us a straightforward way of trading off computational cost and accuracy of estimates.

MaxWalkSat assumes that all weights are positive, while learning can produce negative weights. However, it is easily shown that a formula with a negative weight in a grounded MLN is equivalent to its negation with the symmetric weight. We thus perform this conversion, if necessary, before passing a ground network to MaxWalkSat. The negation of a clause $\bigvee_{i=1}^{n} L_i$, where L_i is a literal, is $\bigwedge_{i=1}^{n} \neg L_i$, a conjunction of n unit clauses. If the original clause's weight is w, we assign a weight of $-w/n$ to each of these unit clauses.

A step of gradient ascent consists of setting $w_{i,t} = w_{i,t-1} + \eta \frac{\partial}{\partial w_i} \log P_w(y|x)|_{w_{t-1}}$. The original voted perceptron algorithm uses a learning rate of $\eta = 1$. In the generalized version, we have found it useful to set this value using a validation set, as is typically done. Weights are initialized to the corresponding clauses' log odds of being true in the data; this tends to produce faster convergence. (Because the optimization problem is convex, the initial state does not affect the solution found.)

Experiments

Databases

We carried out experiments on two publicly-available databases: the UW-CSE database used by Richardson and Domingos (2004) (available at http://www.cs.washington.edu/ai/mln), and McCallum's Cora database of computer science citations as segmented by Bilenko and Mooney (2003) (available at http://www.cs.utexas.edu/users/ml/riddle/data/cora.tar.gz).

The UW-CSE domain consists of 22 predicates and 1158 constants divided into 10 types.[1] Types include: publication, person, course, etc. Predicates include: `Student(person)`, `Professor(person)`, `AdvisedBy(person1, person2)`, `TaughtBy(course, person, quarter)`, `Publication(paper, person)` etc. Using typed variables, the total number of possible ground atoms is 4,055,575. The database contains a total of 3212 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 Richardson and Domingos (2004).

The Cora database is a collection of 1295 different citations to computer science research papers. We cleaned it up by correcting some labels and filling in missing values. This cleaned-up version contains references to 132 different research papers. We used the author, venue, and title fields. The goal is to de-duplicate citations, authors and venues (i.e., to determine which pairs of citations, author fields, and venue fields refer to the same underlying paper, author and venue, respectively). We thus defined the equality predicates `SameCitation(citation1, citation2)`, `SameAuthor(author1, author2)`, and `SameVenue(venue1, venue2)`. We also defined an equality predicate for each pair of title field values, i.e., `SameTitle(title1, title2)`. For each field, we defined six predicates testing whether the cosine TF-IDF similarity score (Salton & McGill 1983) of two field values lies in a particular range (0, 0–0.2, 0.2–0.4, etc.). For example, `TitleTFIDF.4(title1, title2)` is true if the titles `title1` and `title2` have a TF-IDF score in the range $(0.2, 0.4]$, and false otherwise. The `TFIDF` predicates can be computed directly from the data, and their groundings are the evidence atoms. At inference time, all equality predicates are unknown; we hand-labeled them for training and testing purposes (except for `SameCitation`, which was provided in the original database). Using typed variables, the total number of possible ground atoms is 401,552. The database contained a total of 82,026 tuples (true ground atoms). We hand-coded a KB for this domain, consisting of 46 clauses stating regularities like: if two citations are the same, their authors, venues, etc., are the same, and vice-versa; if two fields have a high TF-IDF score, they are the same; etc. While these are not valid as categorical logical statements, they capture important probabilistic relationships when incorporated with weights into an MLN.

Systems

We compared three versions of MLN weight learning, applied to the formulas in the hand-coded KB: the voted perceptron algorithm with MaxWalkSat inference, as described in the previous section (MLN(VP)); maximum likelihood using MC-MLE, as described in Richardson and Domingos (2004) (MLN(ML)); and pseudo-likelihood, as described in Richardson and Domingos (2004) (MLN(PL)). In addition, we compared MLN learning with a pure ILP approach (CLAUDIEN (De Raedt & Dehaspe 1997) (CL)), a

[1] We added the equality predicates that were missing from the original database.

pure knowledge-based approach (the hand-coded KB (KB)), and two pure probabilistic approaches: naive Bayes (NB) (Domingos & Pazzani 1997) and Bayesian networks (BN) (Heckerman, Geiger, & Chickering 1995).

In MLN(VP) training, we used a single run of MaxWalkSat during each learning iteration. In the UW-CSE domain we used $\eta = 0.001$ and $T = 200$. In Cora we used $\eta = 5 \times 10^{-7}$ and $T = 100$. (Cora has a much larger number of query predicates than UW-CSE, necessitating the use of a much smaller learning rate.) Following Richardson and Domingos (2004), we trained MLN(PL) and MLN(ML) using L-BFGS with a zero-centered Gaussian prior. We used Gibbs sampling to approximate the function value and gradient for MLN(ML). Running it to convergence was found to be too slow for the learning to complete in any practical amount of time. A single inference did not satisfy the convergence criteria even when run for more than 24 hours. To obtain the estimates in a reasonable time, we limited each Gibbs sampling inference to at most 50,000 passes for burn-in and 500,000 passes for mixing. The total learning time was limited to a maximum of three days. For inference in all the MLN systems, we used a single run of MaxWalkSat to compute the MAP truth values of non-evidence atoms, followed by Gibbs sampling to compute their conditional probabilities given the evidence.

We used the same settings for CLAUDIEN as Richardson and Domingos, and let CLAUDIEN run for 24 hours on a Sun Blade 1000 workstation.[2]

In the UW-CSE domain, we used the algorithm of Richardson and Domingos (2004) to construct attributes for the naive Bayes and Bayesian network learners. Following Richardson and Domingos, we tried using order-1 and order-1+2 attributes, and report the best results. In the Cora domain, we used the evidence predicates as attributes (i.e., we predicted whether two fields values are the same from their TF-IDF similarity scores; for the citations, evidence values from all the corresponding fields were used as attributes.).

Methodology

In the UW-CSE domain, we used the same leave-one-area-out methodology as Richardson and Domingos (2004). In the Cora domain, we performed eight-fold cross-validation, ensuring that no true set of matching records was split between folds, to avoid train-test set contamination. For each system on each test set, we measured the conditional log-likelihood (CLL) and area under the precision-recall curve (AUC) for the query predicate. 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 atom's log-probability given the evidence. The precision-recall curve for a predicate is computed by varying the threshold CLL above which a ground atom is predicted to be true. We computed the standard deviations of the AUCs using the method of Richardson and Domingos (2004). To obtain probabilities from CL and KB (required to compute CLLs and AUCs) we treated CLAUDIEN's output and the hand-coded KBs as MLNs with all equal infinite weights. We smoothed all probabilities for all systems as in Richardson and Domingos (2004).

[2]CLAUDIEN only runs on Solaris machines.

Table 1: Experimental results on the UW-CSE database.

System	CLL	AUC
MLN(VP)	-0.033 ± 0.003	0.295 ± 0.022
MLN(ML)	-0.063 ± 0.004	0.077 ± 0.011
MLN(PL)	-0.034 ± 0.003	0.232 ± 0.024
KB	-0.053 ± 0.004	0.114 ± 0.004
CL	-0.693 ± 0.000	0.006 ± 0.000
NB	-1.237 ± 0.035	0.065 ± 0.000
BN	-0.046 ± 0.002	0.020 ± 0.000

Results

The results on the UW-CSE domain are shown in Table 1. MLNs with discriminative training clearly outperform all the other approaches. The poor results of MLN(ML) are attributable to the fact that learning using non-converged chains of Gibbs sampling produces parameters which are far from optimal. MLN(PL) is only marginally worse than MLN(VP) on CLL but loses substantially on AUC. Purely logical and purely probabilistic approaches all give poor results.

The results on Cora are shown in Table 2. Again, MLNs with discriminative training outperform the other approaches. The performance of MLN(ML) is very poor due to the non-convergence of Gibbs sampling. MLN(PL) performs much worse than MLN(VP) on both CLL and AUC. The purely logical KB performs very poorly. CL performs better than KB, but still much worse than MLN(VP). Among the probabilistic approaches, BN performs marginally better than MLN(VP) on CLL on predicting citations, but significantly worse on AUC. On authors and venues, MLN(VP) outperforms all other approaches on both CLL and AUC.

Conclusion

Markov logic networks are a powerful combination of logic and probability. In this paper we developed an algorithm for discriminative learning of MLN parameters by combining the voted perceptron with a weighted satisfiability solver, and we verified experimentally that this approach outperforms generative learning.

Directions for future work include discriminative learning of MLN structure (cf. (McCallum 2003)), maximum-margin learning of MLN parameters (cf. (Taskar *et al.* 2004)), extensions of MaxWalkSat, and further application of MLNs to entity resolution, link prediction and other problems.

Acknowledgments

This research was partly supported by ONR grant N00014-02-1-0408 and by a Sloan Fellowship awarded to the second author.

Table 2: Experimental results on the Cora database.

	Citation		Author		Venue	
System	CLL	AUC	CLL	AUC	CLL	AUC
MLN(VP)	−0.069±0.001	0.973±0.000	−0.069±0.008	0.969±0.001	−0.232±0.006	0.771±0.003
MLN(ML)	−13.261±0.013	0.111±0.000	−12.973±0.086	0.162±0.000	−13.380±0.034	0.061±0.000
MLN(PL)	−0.699±0.000	0.722±0.001	−3.062±0.046	0.323±0.001	−0.708±0.002	0.342±0.005
KB	−8.629±0.009	0.149±0.000	−8.096±0.062	0.180±0.000	−8.475±0.022	0.096±0.000
CL	−0.461±0.000	0.187±0.000	−2.375±0.069	0.090±0.000	−1.261±0.022	0.047±0.000
NB	−0.082±0.000	0.945±0.000	−0.203±0.008	0.734±0.006	−0.233±0.003	0.339±0.001
BN	−0.067±0.000	0.951±0.000	−0.203±0.008	0.734±0.006	−0.233±0.003	0.339±0.001

References

Besag, J. 1975. Statistical analysis of non-lattice data. *The Statistician* 24:179–195.

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

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*, 1–8. Philadelphia, PA: ACL.

De Raedt, L., and Dehaspe, L. 1997. Clausal discovery. *Machine Learning* 26:99–146.

Della Pietra, S.; Della Pietra, V.; and Lafferty, J. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19:380–392.

Dietterich, T.; Getoor, L.; and Murphy, K., eds. 2003. *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*. Banff, Canada: IMLS.

Domingos, P., and Pazzani, M. 1997. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning* 29:103–130.

Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1300–1307. Stockholm, Sweden: Morgan Kaufmann.

Genesereth, M. R., and Nilsson, N. J. 1987. *Logical Foundations of Artificial Intelligence*. San Mateo, CA: Morgan Kaufmann.

Gilks, W. R.; Richardson, S.; and Spiegelhalter, D. J., eds. 1996. *Markov Chain Monte Carlo in Practice*. London, UK: Chapman and Hall.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20:197–243.

Kautz, H.; Selman, B.; and Jiang, Y. 1997. A general stochastic approach to solving problems with hard and soft constraints. In Gu, D.; Du, J.; and Pardalos, P., eds., *The Satisfiability Problem: Theory and Applications*. New York, NY: American Mathematical Society. 573–586.

Kersting, K., and De Raedt, L. 2001. Towards combining inductive logic programming with Bayesian networks. In *Proceedings of the Eleventh International Conference on Inductive Logic Programming*, 118–131. Strasbourg, France: Springer.

Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 282–289. Williamstown, MA: Morgan Kaufmann.

McCallum, A. 2003. Efficiently inducing features of conditional random fields. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, 403–410. Acapulco, Mexico: Morgan Kaufmann.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77:257–286.

Richardson, M., and Domingos, P. 2004. Markov logic networks. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://www.cs.washington.edu/homes/pedrod/mln.pdf.

Roth, D. 1996. On the hardness of approximate reasoning. *Artificial Intelligence* 82:273–302.

Salton, G., and McGill, M. J. 1983. *Introduction to Modern Information Retrieval*. New York, NY: McGraw-Hill.

Selman, B.; Kautz, H.; and Cohen, B. 1996. Local search strategies for satisfiability testing. In Johnson, D. S., and Trick, M. A., eds., *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. Washington, DC: American Mathematical Society. 521–532.

Taskar, B.; Abbeel, P.; and Koller, D. 2002. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, 485–492. Edmonton, Canada: Morgan Kaufmann.

Taskar, B.; Wong, M. F.; Abbeel, P.; and Koller, D. 2004. Max-margin Markov networks. In Thrun, S.; Saul, L.; and Schölkopf, B., eds., *Advances in Neural Information Processing Systems 16*. Cambridge, MA: MIT Press.