Online Max-Margin Weight Learning for Markov Logic Networks

Tuyen N. Huynh* Raymond J. Mooney*

Abstract

Most of the existing weight-learning algorithms for Markov Logic Networks (MLNs) use batch training which becomes computationally expensive and even infeasible for very large datasets since the training examples may not fit in main memory. To overcome this problem, previous work has used online learning algorithms to learn weights for MLNs. However, this prior work has only applied existing online algorithms, and there is no comprehensive study of online weight learning for MLNs. In this paper, we derive a new online algorithm for structured prediction using the primal-dual framework, apply it to learn weights for MLNs, and compare against existing online algorithms on three large, real-world datasets. The experimental results show that our new algorithm generally achieves better accuracy than existing methods, especially on noisy datasets.

Keywords: Online learning, structured prediction, statistical relational learning

1 Introduction

Statistical relational learning (SRL) concerns the induction of probabilistic knowledge that supports accurate prediction for multi-relational structured data [11]. These powerful SRL models have been successfully applied to a variety of real-world problems. However, the power of these models come with a cost, since they can be computationally expensive to train, in particular since most existing SRL learning methods employ batch training where the learner must repeatedly run inference over all training examples in each iteration. Training becomes even more expensive in larger datasets containing thousands of examples, and even infeasible in some cases where there is not enough main memory to fit the training data [24]. A well-known solution to this problem is online learning where the learner sequentially processes one example at a time. In this work, we look at the problem of online weight learning for Markov Logic Networks (MLNs), a recently developed SRL model that generalizes both full first-order logic and Markov networks [29, 10]. Riedel and Meza-Ruiz [31] and Mihalkova and Mooney [24] have used online learning algorithms to learn weights for MLNs. However, previous work only applied one existing online algorithm to MLNs and did not provide a comparative study of online weight learning for MLNs.

In this work, we derive a new online algorithm for structured prediction [1] from the primal-dual framework for strongly convex loss functions [16], which is the latest framework for deriving online algorithms that have low regret, and apply it to learn weights for MLNs and compare against existing online algorithms that have been used in previous work. The experimental results show that our new algorithms generally achieve better accuracy than existing algorithms on three large, real-world datasets, especially on noisy datasets.

2 Background

2.1 Notation We use lower case letters (e.g. w, λ) to denote scalars, bold face letters (e.g. $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{\lambda}$) to denote vectors, and upper case letters (e.g. W, X) to denote sets. The inner product between vectors $\boldsymbol{w}$ and $\boldsymbol{x}$ is denoted by $\langle \boldsymbol{w}, \boldsymbol{x} \rangle$.

2.2 MLNs An MLN consists of a set of weighted first-order clauses. It provides a way of softening first-order logic by making situations in which not all clauses are satisfied less likely but not impossible [29, 10]. More formally, let X be the set of all propositions describing a world (i.e. the set of all ground atoms), $\mathcal{F}$ 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 normalization constant. Then the probability of a particular truth assignment $\mathbf{x}$ to the variables in X is defined as [29]:

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

*Department of Computer Science, The University of Texas at Austin, 1616 Guadalupe, Suite 2.408 , Austin, Texas 78701, USA. {hntuyen,mooney}@cs.utexas.edu

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

There are two inference tasks in MLNs. The first one is to infer the Most Probable Explanation (MPE) or the most probable truth values for a set of unknown literals $\mathbf{y}$ given a set of known literals $\mathbf{x}$, provided as evidence. Both approximate and exact MPE methods for MLNs have been proposed [17, 30, 14]. The second inference task is computing the conditional probabilities of some unknown literals, $\mathbf{y}$, given some evidence $\mathbf{x}$. Computing these probabilities is also intractable, but there are good approximation algorithms such as MC-SAT [27] and lifted belief propagation [36].

There are two approaches to weight learning in MLNs: generative and discriminative. In generative learning, the goal is to learn a weight vector that maximizes the likelihood of all the observed data [29]. In discriminative learning, we know a priori which predicates will be used to supply evidence and which ones will be queried, and the goal is to correctly predict the latter given the former. Several discriminative weight learning methods have been proposed, most of which try to find weights that maximize the conditional log-likelihood of the data [35, 20, 13]. Recently, Huynh and Mooney [14] proposed a max-margin approach to learn weights for MLNs.

2.3 The Primal-Dual Algorithmic Framework for Online Convex Optimization In this section, we briefly review the primal-dual framework for strongly convex loss functions [16] which is the latest framework for deriving online algorithms that have low regret, the difference between the cumulative loss of the online algorithm and the cumulative loss of the optimal offline solution. Considering the following primal optimization problem:

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

where $f : W \rightarrow R_+$ is a function that measures the complexity of the weight vectors in W, $g_i : W \rightarrow R$ is a loss function, and σ is non-negative scalar. For example, if $W = R^d$, $f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2$, and $g_i(\mathbf{w}) = \max_{y \in \mathcal{Y}} [\rho(\mathbf{y}_t, \mathbf{y}) - \langle \mathbf{w}, (\boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y})) \rangle]_+$ where $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{Y} \rightarrow R^d$ is a joint feature function, then the above optimization problem is the max-margin structured classification problem [39, 41, 38]. We can rewrite the optimization problem in Eq. 2.1 as follows:

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

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

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

where each $\boldsymbol{\lambda}_t$ is a vector of Lagrange multipliers for the equality constraint $\mathbf{w}_t = \mathbf{w}_0$, and $f^*, g_1^*, ..., g_t^*$ are the Fenchel conjugate functions of $f, g_1, ..., g_t$. A Fenchel conjugate function of a function $f : W \rightarrow R$ is defined as $f^*(\boldsymbol{\theta}) = sup_{\mathbf{w} \in W}(\langle \mathbf{w}, \boldsymbol{\theta} \rangle - f(\mathbf{w}))$. See [16] for details on the steps to derive the dual problem.

From the weak duality theorem [3], we know that the dual objective is upper bounded by the optimal value of the primal problem. Thus, if an online algorithm can incrementally ascend the dual objective function in each step, then its performance is close to the performance of the best fixed weight vector that minimizes the primal objective function (the best offline learner), since by increasing the dual objective, the algorithm moves closer to the optimal primal value.

Based on this observation, Kakade et. al. [16] proposed the general online incremental dual ascent algorithm (Algorithm 1), where $\partial g_t(\mathbf{w}_t) = \{\boldsymbol{\lambda} : \forall \mathbf{w} \in W, g_t(\mathbf{w}) - g_t(\mathbf{w}_t) \geq \langle \boldsymbol{\lambda}, (\mathbf{w} - \mathbf{w}_t) \rangle\}$ is the set of subgradients of g_t at w_t. The condition $\exists \boldsymbol{\lambda}' \in \partial g_t$ s.t. $D_{t+1}(\boldsymbol{\lambda}_1^{t+1}, ..., \boldsymbol{\lambda}_t^{t+1}) \geq D_{t+1}(\boldsymbol{\lambda}_1^t, ..., \boldsymbol{\lambda}_{t-1}^t, \boldsymbol{\lambda}')$ ensures the dual objective is increased in each step. The regret of any algorithm derived from Algorithm 1 is $O(\log T)$ [16], where T is the number of examples seen so far.

A simple update rule that satisfies the condition in Algorithm 1 is to find a subgradient $\boldsymbol{\lambda}' \in \partial g_t(\mathbf{w}_t)$ and set $\boldsymbol{\lambda}_t^{t+1} = \boldsymbol{\lambda}'$ and keep all other $\boldsymbol{\lambda}_i$'s unchanged (i.e.

Algorithm 1 A general incremental dual ascent algorithm for σ-strongly convex loss function [16]

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

$\boldsymbol{\lambda}_i^{t+1} = \boldsymbol{\lambda}_i^t,\ \forall i < t$). However, the gain in the dual objective for this simple update rule is minimal. To achieve the largest gain in the dual objective, one can optimize all the $\boldsymbol{\lambda}_i$'s at each step. But this approach is usually computationally prohibitive to use since at each step, we need to solve a large optimization problem:

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

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

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

This is called the Coordinate-Dual-Ascent (CDA) update rule. If we can find a closed-form solution of the optimization problem with respect to the last variable $\boldsymbol{\lambda}_t$, then the computational complexity of the CDA update is similar to the simple update but the gain in the dual objective function is larger. Previous work [34] showed that algorithms which more aggressively ascend the dual function have better performance. In the next section, we will show that it is possible to obtain a closed-form solution of the CDA update rule for the case of structured prediction.

3 Online Coordinate-Dual-Ascent Algorithms for Structured Prediction

In this section, we derive new online algorithms for structured prediction based on the algorithmic framework described in the previous section using the CDA update rule. In structured prediction [1], the label $\mathbf{y_t}$ of each example $\mathbf{x_t} \in \mathcal{X}$ belongs to some structure output space $\mathcal{Y}$. We assume that there is a joint feature function $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{Y} \to R^d$ and the prediction function takes the following form:

$$h_{\mathbf{w}}(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) \rangle$$

So in this case the weight vector $\mathbf{w}$ lies in R^d. A standard complexity function used in many tasks is $f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2$. Regarding the loss function g_t, a generalized version of the Hinge loss is widely used in max-margin structured prediction [39, 41]

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

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

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

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

The loss $l_{ML}(\mathbf{w}_t, (\mathbf{x}_t, \mathbf{y}_t))$ is the greatest loss the algorithm would suffer at step t if it used the maximal loss label $\mathbf{y}_t^{ML}$ as the prediction. On the other hand, it checks whether the max-margin constraints are satisfied since if $l_{ML}(\mathbf{w}_t, (\mathbf{x}_t, \mathbf{y}_t)) = 0$ then $\mathbf{y}_t^{ML} = \mathbf{y}_t$, and it means that the current weight vector $\mathbf{w}_t$ scores the correct label $\mathbf{y}_t$ higher than any other label $\mathbf{y}_t'$ where the difference is at least $\rho(\mathbf{y_t}, \mathbf{y}_t')$. Note that the maximal loss label $\mathbf{y}_t^{ML}$ is the input to the maximal loss (it is possible in online learning since the loss is computed after the weight vector $\mathbf{w}_t$ is chosen), therefore it does not depend on the weight vector $\mathbf{w}$ for which we want to compute the loss. So the maximal loss function only concerns the particular constraint for whether the true label $\mathbf{y}_t$ is scored higher than the maximal loss label with a margin of $\rho(\mathbf{y_t}, \mathbf{y}_t^{ML})$. This is the key difference between the maximal loss and the max-margin loss since the latter looks at the constraints of all possible labels. The main drawback of the maximal loss is that finding the maximal loss label $\mathbf{y}_t^{ML}$, which is also called the loss-augmented inference problem [38], is only feasible for some decomposable label loss functions [38] such as Hamming loss (the number of misclassified atoms) since the maximal loss label depends on the label loss function $\rho(\mathbf{y}_t, \mathbf{y}')$. This is the reason why we want to consider the second loss function, prediction-based loss, which can be used with any label loss function such as $(1 - F_1)$ loss, where F_1 is the harmonic mean of precision and recall.

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

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

Like the maximal loss, the prediction-based loss only concerns the constraint for the prediction label $\mathbf{y}_t^P$. We have $l_{PL}(\mathbf{w}_t, (\mathbf{x}_t, \mathbf{y}_t)) \leq l_{ML}(\mathbf{w}_t, (\mathbf{x}_t, \mathbf{y}_t))$ since $\mathbf{y}_t^{ML}$

is the maximal loss label for $\mathbf{w}_t$. As a result, the update based on the prediction-based loss function is less aggressive than the one based on the maximal loss function. However, the prediction-based loss function can be used with any label loss function since the predicted label $\mathbf{y}_t^P$ does not depend on the label loss function.

To apply the primal-dual algorithmic framework described in the previous section, we need to find the Fenchel conjugate function of the complexity function $f(\mathbf{w})$ and the loss function $g(\mathbf{w})$. The Fenchel conjugate function of the complexity function $f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2$ is itself, i.e. $f^*(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{\theta}||_2^2$ [3]. For the loss function, recall that the Fenchel conjugate function of the Hinge-loss $g(\mathbf{w}) = [\gamma - \langle \mathbf{w}, \mathbf{x}\rangle]_+$ is:

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

(Appendix A in [33]). We can see that both the prediction-based loss and the maximal loss have the same form as the Hinge-loss where γ is replaced by the label loss function $l(\mathbf{y}_t, \mathbf{y}_t^P)$ and $l(\mathbf{y}_t, \mathbf{y}_t^{ML})$, and $\mathbf{x}$ is replaced by $\Delta\boldsymbol{\phi}_t^{PL} = \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t^P)$ and $\Delta\boldsymbol{\phi}_t^{ML} = \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t^{ML})$ for the prediction-based loss and the maximal loss respectively. Using the result of the Hinge-loss, we have the Fenchel conjugate function of the prediction-based loss and the maximal loss as follows:

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

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

$$\text{(3.2)} \quad \text{argmax}_{\boldsymbol{\lambda}_t} - (\sigma t) f^*\left(-\frac{\boldsymbol{\lambda}_{1:(t-1)} + \boldsymbol{\lambda}_t}{(\sigma t)}\right) - g_t^*(\boldsymbol{\lambda}_t)$$

where $\boldsymbol{\lambda}_{1:(t-1)} = \sum_{i=1}^{t-1} \boldsymbol{\lambda}_i$. Substituting the conjugate function f^* and g_t^* as above in the equation 3.2, we obtain the following optimization problem:

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

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

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

Algorithm 2 Online Coordinate-Dual-Ascent Algorithms for Structured Prediction

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

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

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

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

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

The new method is summarized in Algorithm 2. Interestingly, this update formula has the same form as that of the subgradient algorithm [25] which is derived

from the simple update criterion:

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

The key difference is in the learning rate. The learning rate of the subgradient algorithm, which is equal to $1/(\sigma t)$, does not depend on the loss suffered at each step, while the learning rate of CDA is the minimization of $1/(\sigma t)$ and the loss suffered at each step. In the beginning, when t is small and therefore $1/(\sigma t)$ is large (assuming σ is small), CDA's learning rate is controlled by the loss suffered at each step. In contrast, when t is large and therefore $1/(\sigma t)$ is small, then the learning rate of CDA is driven by the quantity $1/(\sigma t)$. In other words, at the beginning, when the model is not good, CDA aggressively updates the model based on the loss suffered at each step; and later when the model is good, it updates the model less aggressively.

We can use the derived CDA algorithm to perform online weight learning for MLNs since the weight learning problem in MLNs can be cast as a max-margin structured prediction problem [14]. For MLNs, the number of true groundings of the clauses plays the role of the joint feature function $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y})$.

4 Experimental Evaluation

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

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

4.1 Datasets We ran experiments on three large, real-world datasets: the CiteSeer dataset [19] for bibliographic citation segmentation, a web search query dataset [24] obtained from Microsoft Research for query disambiguation, and the CoNLL 2005 dataset [4] for Semantic Role Labeling.

For CiteSeer, we used the version created by Poon and Domingos [28] and the simplest MLN, the isolated segmentation model, in their work.[1] The dataset contains 1,563 bibliographic citations such as:

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

The task is to segment each of these citations into three fields: *Author*, *Title* and *Venue*. The dataset has four independent subsets consisting of citations to disjoint publications in four different research areas.

For the search query disambiguation, we used the data created by Mihalkova and Mooney [24]. The dataset consists of thousands of search sessions where ambiguous queries are asked. The data are split into 3 disjoint sets: training, validation, and test. There are $4,618$ search sessions in the training set, $4,803$ sessions in the validation set, and $11,234$ sessions in the test set. In each session, the set of possible search results for a given ambiguous query is given, and the goal is to rank these results based on how likely it will be clicked by the user. A user may click on more than one result for a given query. To solve this problem, Mihalkova and Mooney [24] proposed three different MLNs which correspond to different levels of information used in disambiguating the query. We used all three MLNs in our experiments. In comparison to the Citeseer dataset, the search query dataset is larger but is much noisier since a user can click on a result because it is relevant or because the user is just doing an exploratory search.

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

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

A verb and its set of semantic roles form a proposition in the sentence, and a sentence usually contains more than one proposition. Each proposition serves as a training example. The dataset consists of three disjoint subsets: training, development, and test. The number of propositions (or examples) in the training, development, and test sets are: $90,750$; $3,248$; and $5,267$ respectively.[2] We used the MLN constructed by Riedel [30] which contains clauses that capture the features of constituents and dependencies between semantic components of the same verb.

[1] Both the dataset and the MLN can be found at `http://alchemy.cs.washington.edu/data/citeseer/`

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

Table 1: F_1 scores on CiteSeer dataset. Highest F_1 scores are shown in bold.

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

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

MM: The offline max-margin weight learner for MLNs proposed by Huynh and Mooney [14].[3]

1-best MIRA: MIRA is one of the first online learning algorithms for structured prediction proposed by McDonald et. al [22]. A simple version of MIRA, called 1-best MIRA, is widely used in practice since its update rule has a closed-form solution. 1-best MIRA has been used in previous work [31] to learn weights for MLNs. In each round, it updates the weight vectors $\mathbf{w}$ as follows:

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

Subgradient: This algorithm proposed by Ratliff et al. [25] is an extension of the Greedy Projection algorithm [42] to the case of structured prediction. Its update rule is an instance of the simple update criterion

CDA: Our newly derived online learning algorithm presented in Algorithm 2.

Regarding label loss functions, we use Hamming (HM) loss which is the standard loss function for structured prediction [39, 41]. As mentioned earlier, Hamming loss is a decomposable loss function, so it can be used with both maximal loss and prediction-based loss. Since F_1 is the standard evaluation metric for the citation segmentation task on Citeseer, we also considered the label loss function $100(1 - F_1)$ [15]. However, since this loss function is not decomposable, we can only use it with the prediction-based loss.

In training, for online learning algorithms, we use the exact MPE inference based on Integer Linear Programming (ILP) described by Huynh and Mooney [14] on Citeseer and web search query datasets, and Cutting Plane Inference [30] on the CoNLL 2005 dataset. For the offline weight learner MM, we use the approximate inference algorithm developed by Huynh and Mooney [14] since it is computationally intractable to run exact inference for all training examples at once. In testing, we use MCSAT to compute marginal probabilities for the web search query dataset since we want to rank the query results, and exact MPE inference on the other two datasets. For all online learning algorithms, we ran one pass over the training set and used the average weight vector to predict on the test set. For CiteSeer, we ran four-fold cross-validation (i.e. leave one topic out). The parameter σ of the Subgradient and CDA is set based on the performance on the validation set except Citeseer where the parameter is set based on training performance.

For testing the statistical significance between the performance of different algorithms, we use McNemar's test [9] on Citeseer and a two-sided paired t-test on the web search query. The significance level was set to 5% (p-value smaller than 0.05) for both cases.

4.3 Metrics Like previous work, for citation segmentation on Citeseer, we used F_1 at the token level to measure the performance of each algorithm; for search query disambiguation, we used MAP (Mean Average Precision) which measures how close the relevant results are to the top of the ranking; and for semantic role labeling on CoNLL 2005, we used F_1 of the predicted arguments as described in [4].

4.4 Results and Discussion Table 1 presents the F_1 scores of different algorithms on Citeseer. On this dataset, the CDA algorithm with maximal loss, CDA-ML-HM, has the best F_1 scores across four folds. These results are statistically significantly better than those of subgradient method. So aggressive update in the dual results in a better F_1 scores. The F_1 scores of CDA-ML-HM are a little bit higher than those of 1-best-MIRA, but the difference is not significant. Interestingly, with

[3] This max-margin weight learner has been shown to be comparable to other offline weight learners for MLNs [14].

Table 2: Average training time on CiteSeer dataset.

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

Table 3: MAP scores on Microsoft search query dataset. Highest MAP scores are shown in bold.

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

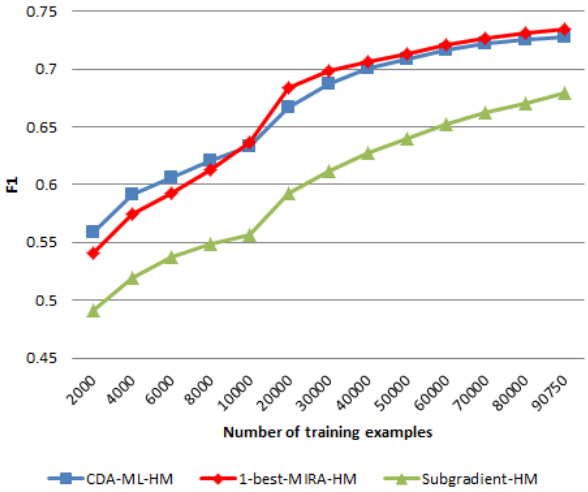

Figure 1: Learning curve on CoNLL 2005

the possibility of using exact inference in training, CDA is a little bit more accurate than the batch max-margin algorithm (MM) since the batch learner can only afford to use approximate inference in training. Other advantages of online algorithms are in terms of training time and memory. Table 2 shows the average training time of different algorithms on this dataset. All online learning algorithms took on average about 12-13 minutes for training while the batch one took an hour and a half on the same machine. In addition, since online algorithms process one example at a time, they use much less memory than batch methods. On the other hand, the running time results also confirm that the new algorithm, CDA, has the same computational complexity as other existing online methods. Regarding the comparison between maximal loss and prediction-based loss, the former is better than the latter on this dataset due to its more aggressive updates. For prediction-based loss function, there is not much difference between using different label loss functions in this case.

Table 3 shows the MAP scores of different algorithms on the Microsoft web search query dataset. The first row in the table is from Mihalkova and Mooney [24] who used a variant of the structured perceptron [5] called Contrastive Divergence (CD) [12] to do online weight learning for MLNs. It is clear that the CDA algorithm has better MAP scores than CD. For this dataset, we were unable to run offline weight learning since the large amount of training data exhausted memory during training. The 1-best MIRA has the worst MAP scores on this dataset. This behavior can be explained as follows. From the update rule of the 1-best MIRA algorithm, we can see that it aggressively updates the weight vector according to the loss incurred in each round. Since this dataset is noisy, this update rule leads to overfitting. This also explains why the subgradient algorithm has good performance on this data since its update rule does not depend on the loss incurred in each round. The MAP scores of the CDA algorithms are not significantly better than that of the subgradient method, but their performance is more consistent across the three MLNs. Regarding the loss function, the MAP scores of CDA-PL and CDA-ML are almost the same.

Figure 1 shows the learning curve of three online learning algorithms: CDA, 1-best MIRA and subgradient on the CoNLL 2005 dataset. In general, the relative accuracy of three algorithms is similar to what we have seen on Citeseer. CDA outperforms the subgradient method across the whole learning curve. In particular, at $30,000$ training examples, about $1/3$ of the training set, the F_1 score of CDA is already better than the that of the subgradient method trained on the whole training set. The performance of CDA and 1-best MIRA are comparable to each other, except on the early part of the learning curve (less than $10,000$ examples) where the F_1 scores of CDA are about 1 to 2 percentage points higher than those of 1-best MIRA.

The CoNLL 2005 dataset was carefully annotated by experts [26], which is a time consuming and expensive process. Nowadays, a faster and cheaper way to obtain this type of annotation is using crowdsourcing

services such as Amazon Mechanical Turk,[4] which is possible to assign annotation jobs to thousands of people and get results back in a few hours [37]. However, a downside of this approach is the big variance in the quality of labels obtained from different annotators. As a result, there is a lot of noise in the annotated data. To simulate this type of noisy labeled data, we introduce random noise to the CoNLL 2005 dataset. At p percent noise, there is probability p that an argument in a proposition is swapped with another argument in the same proposition. For example, an argument with role "A0" may be swapped to an argument with role "A1" and vice versa. Figure 2 shows the F_1 scores of the above three online learning algorithms on noisy CoNLL 2005 dataset at various levels of noise. With the presence of noise, CDA is the most accurate and also the most robust to noise among the three algorithms. For 10% noise and higher, CDA is significantly better than the other two methods. The F_1 score of CDA at a noise level of 50% is 8.5% higher than that of 1-best MIRA and 12.6% higher than that of the subgradient method. On the other hand, comparing with the F_1 score on the clean dataset, the F_1 score of CDA at 50% of noise only drops 8.4 points while those of 1-best MIRA and subgradient drop about 17.6 and 16.1 respectively. In addition, the F_1 score of CDA at 50% noise is higher than the F_1 score of 1-best MIRA at 35% noise and comparable to the F_1 score of subgradient method at 20% noise.

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

5 Related Work

Online learning for max-margin structured prediction has been studied in several pieces of previous work. In addition to those mentioned earlier, a family of online algorithms similar to the 1-best MIRA, called passive-aggressive algorithms, was presented in [7]. Another piece of related work is the exponentiated gradient algorithm [2, 6] which also performs updates based on the dual of the primal problem. However, the dual problem in [2, 6] is more complicated and expensive to solve since it was derived based on the max-margin loss, l_{MM}. As a result, to efficiently solve the problem, the authors assume that each label $\mathbf{y}$ is a set of parts and both the joint feature and the label loss function can be decomposed into a sum over those

[4] https://www.mturk.com/mturk/

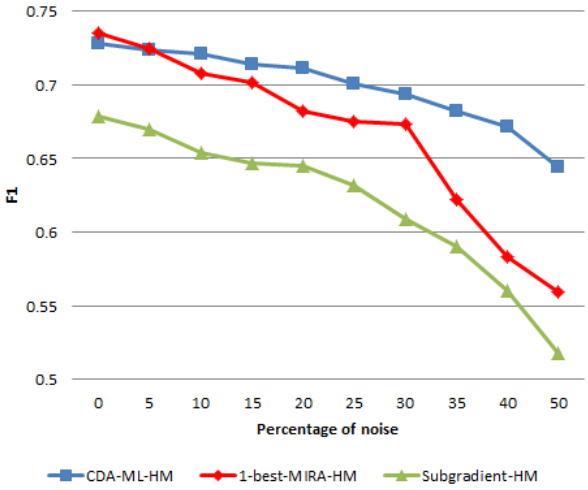

Figure 2: F_1 scores on noisy CoNLL 2005

for the individual parts. Even under this assumption, efficiently computing the marginal values of the part variables is still a challenging problem.

In the context of online weight learning for MLNs, one related algorithm is SampleRank [8] which uses a sampling algorithm to generate samples from a given training example and updates the weight vector whenever it misranks a pair of samples. So unlike traditional online learning algorithms that perform one update per example, SampleRank performs multiple updates per example. However, the performance of SampleRank highly depends on the sampling algorithm, and which sampling algorithms are best is an open research question.

The issue of prediction-based loss versus maximal loss has been discussed previously [7, 32], but no experiments have been conducted to compare them on real-world datasets.

6 Future Work

In this work, we applied our derived online learning algorithm to MLNs, but it can be used for any structured prediction model. So it would be interesting to apply the same method to other structured prediction models such as M3Ns [39], Structural SVMs [41], RMNs [40], and FACTORIE [21]. On the other hand, like most online learners, our algorithm assumes that the model's structure (e.g. the set of clauses in an MLN) is correct, and only updates the model parameters (e.g. the weights of an MLN). However, in practice, the input structure is usually not optimal, so it should be also revised. A number of methods for learning and revis-

ing MLN structure have been developed [23, 18]; however, they are all batch algorithms that do not scale adequately to very large training sets. We are currently developing a new algorithm that performs both online parameter *and* structure learning.

7 Conclusions

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

Acknowledgments

The authors thank Sebastian Riedel for providing the MLN for the semantic role labeling task on CoNLL 2005. We also thank IBM for the free academic license of CPLEX. This research is supported by a gift from Microsoft Research and by ARO MURI grant W911NF-08-1-0242. Most of the experiments were run on the Mastodon Cluster, provided by NSF Grant EIA-0303609. The first author also thanks the Vietnam Education Foundation (VEF) for its sponsorship.

References

[1] Bakir, G.H., Hofmann, T., Schölkopf, B., Smola, A.J., Taskar, B., Vishwanathan, S.V.N. (eds.): Predicting Structured Data. The MIT Press (2007)

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

[3] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)

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

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

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

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

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

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

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

[11] Getoor, L., Taskar, B. (eds.): Introduction to Statistical Relational Learning. MIT Press, Cambridge, MA (2007)

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

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

[14] Huynh, T.N., Mooney, R.J.: Max-margin weight learning for Markov logic networks. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2009), Part I. pp. 564–579 (2009)

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

[16] Kakade, S.M., Shalev-Shwartz, S.: Mind the duality gap: Logarithmic regret algorithms for online optimization. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) Advances in Neural Information Processing Systems 21, Vancouver, British Columbia, Canada, December 8-11, 2008. pp. 1457–1464. MIT Press (2009)

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

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

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

[20] Lowd, D., Domingos, P.: Efficient weight learning

for Markov logic networks. In: Proceedings of 7th European Conference of Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD-2007). pp. 200–211 (2007)
[21] McCallum, A., Schultz, K., Singh, S.: FACTORIE: Probabilistic Programming via Imperatively Defined Factor Graphs. In: Advances in Neural Information Processing Systems 22 (NIPS-2009). pp. 1249–1257 (2009)
[22] McDonald, R., Crammer, K., Pereira, F.: Online large-margin training of dependency parsers. In: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL-05). pp. 91–98. Association for Computational Linguistics, Morristown, NJ, USA (2005)
[23] Mihalkova, L., Mooney, R.J.: Bottom-up learning of Markov logic network structure. In: Proceedings of 24th International Conference on Machine Learning (ICML-2007). Corvallis, OR (June 2007)
[24] Mihalkova, L., Mooney, R.J.: Learning to disambiguate search queries from short sessions. In: Buntine, W.L., Grobelnik, M., Mladenic, D., Shawe-Taylor, J. (eds.) Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD 2009), Part II. pp. 111–127 (2009)
[25] Nathan Ratliff, J.A.D.B., Zinkevich, M.: (Online) subgradient methods for structured prediction. In: Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AIStats) (2007)
[26] Palmer, M., Gildea, D., Kingsbury, P.: The Proposition Bank: An Annotated Corpus of Semantic Roles. Computational Linguistics 31(1), 71–106 (2005)
[27] Poon, H., Domingos, P.: Sound and efficient inference with probabilistic and deterministic dependencies. In: Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI-06). Boston, MA (July 2006)
[28] Poon, H., Domingos, P.: Joint inference in information extraction. In: Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI-07). pp. 913–918. Vancouver, British Columbia, Canada (2007)
[29] Richardson, M., Domingos, P.: Markov logic networks. Machine Learning 62, 107–136 (2006)
[30] Riedel, S.: Improving the accuracy and efficiency of MAP inference for Markov logic. In: Proceedings of 24th Conference on Uncertainty in Artificial Intelligence (UAI-2008). pp. 468–475. Helsinki, Finland (2008)
[31] Riedel, S., Meza-Ruiz, I.: Collective semantic role labelling with Markov logic. In: Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL'08) (2008)
[32] Shalev-Shwartz, S.: Online Learning: Theory, Algorithms, and Applications. Ph.D. thesis, The Hebrew University of Jerusalem (2007)
[33] Shalev-Shwartz, S., Singer, Y.: Convex repeated games and Fenchel duality. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in Neural Information Processing Systems 19, pp. 1265–1272. MIT Press (2007)
[34] Shalev-Shwartz, S., Singer, Y.: A unified algorithmic approach for efficient online label ranking. In: Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics (AIStats) (2007)
[35] Singla, P., Domingos, P.: Discriminative training of Markov logic networks. In: Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05). pp. 868–873 (2005)
[36] Singla, P., Domingos, P.: Lifted first-order belief propagation. In: Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI-08). pp. 1094–1099. Chicago, Illinois, USA (2008)
[37] Snow, R., O'Connor, B., Jurafsky, D., Ng, A.Y.: Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-2008). pp. 254–263. Association for Computational Linguistics, Morristown, NJ, USA (2008)
[38] Taskar, B., Chatalbashev, V., Koller, D., Guestrin, C.: Learning structured prediction models: a large margin approach. In: Proceedings of 22nd International Conference on Machine Learning (ICML-2005). pp. 896–903. ACM, Bonn, Germany (2005)
[39] Taskar, B., Guestrin, C., Koller, D.: Max-margin Markov networks. In: Advances in Neural Information Processing Systems 16 (NIPS 2003) (2003)
[40] Taskar, B., Abbeel, P., Koller, D.: Discriminative probabilistic models for relational data. In: Proceedings of 18th Conference on Uncertainty in Artificial Intelligence (UAI-2002). pp. 485–492. Edmonton, Canada (2002)
[41] Tsochantaridis, I., Joachims, T., Hofmann, T., Altun, Y.: Support vector machine learning for interdependent and structured output spaces. In: Proceedings of 21st International Conference on Machine Learning (ICML-2004). pp. 104–112. Banff, Canada (July 2004)
[42] Zinkevich, M.: Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In: Proceedings of 20th International Conference on Machine Learning (ICML-2003). pp. 928–936 (2003)