Online Max-Margin Weight Learning with Markov Logic Networks

Tuyen N. Huynh and **Raymond J. Mooney**
Department of Computer Science
The University of Texas at Austin
1 University Station C0500
Austin, Texas 78712-0233, USA
{hntuyen,mooney}@cs.utexas.edu

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 new online algorithms for structured prediction using the primal-dual framework, apply them to learn weights for MLNs, and compare against existing online algorithms on two large, real-world datasets. The experimental results show that the new algorithms achieve better accuracy than existing methods.

Introduction

Statistical relational learning (SRL) concerns the induction of probabilistic knowledge that supports accurate prediction for multi-relational structured data (Getoor and Taskar 2007). 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 (Mihalkova and Mooney 2009). 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 (Richardson and Domingos 2006). Riedel and Meza-Ruiz (2008) and Mihalkova and Mooney (2009) 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 new online algorithms for structured prediction from the primal-dual framework (Shalev-Shwartz and Singer ; Shalev-Shwartz 2007; Kakade and Shalev-Shwartz 2009), which is the latest framework for deriving online algorithms that have low regret, and apply them 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 achieve better accuracy than existing algorithms on two large, real-world datasets.

Background

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 (Richardson and Domingos 2006). 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 (Richardson and Domingos 2006):

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

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 (Kautz, Selman, and Jiang 1997; Riedel 2008; Huynh and Mooney 2009). 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 (Poon and Domingos 2006) and lifted belief propagation (Singla and Domingos 2008).

There are two approaches to weight learning in MLNs: generative and discriminative. In discriminative learning, we know a priori which predicates will be used to supply evidence and which ones will be queried, and the goal is to correctly predict the latter given the former. Several discriminative weight learning methods have been proposed, most of which try to find weights that maximize the conditional log-likelihood of the data (Singla and Domingos 2005; Lowd and Domingos 2007; Huynh and Mooney 2008). Recently, Huynh and Mooney (2009) proposed a max-margin approach to learn weights for MLNs.

The Primal-Dual Algorithmic Framework for Online Convex Optimization

In this section, we briefly review the primal-dual framework (Shalev-Shwartz and Singer ; Shalev-Shwartz 2007; Kakade and Shalev-Shwartz 2009) 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. First, we look at the case of convex loss where in each step the online algorithm receives a convex loss function g_t. Considering the following optimization problem:

$$\inf_{\mathbf{w}\in\mathcal{W}} \left(\sigma f(\mathbf{w}) + \sum_{t=1}^{T} g_t(\mathbf{w})\right) \tag{1}$$

where $f : W \rightarrow R_+$ is a function that measures the complexity of the vectors in W, and σ is non-negative scalar. For example, if $W = R^d$, $f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2$, and $g_t(\mathbf{w}) = \max_{y\in\mathcal{Y}} [\rho(\mathbf{y}_t, \mathbf{y}) - \langle \mathbf{w}, (\boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t, \mathbf{y}))\rangle]_+$, then the above optimization problem becomes the same as for max-margin structured classification (Taskar, Guestrin, and Koller 2004; Tsochantaridis et al. 2004; Taskar et al. 2005). We can rewrite the optimization problem in Eq. 1 as follows:

$$\inf_{\mathbf{w}_0,\mathbf{w}_1,\ldots,\mathbf{w}_T} \left(\sigma f(\mathbf{w}_0) + \sum_{t=1}^{T} g_t(\mathbf{w}_t)\right)$$

$$\text{s.t. } \mathbf{w}_0 \in \mathcal{W}, \quad \forall t \in 1...T, \mathbf{w}_t = \mathbf{w}_0$$

where we introduce T new vectors $\mathbf{w}_1, ..., \mathbf{w}_T$ and constrain them to all be equal to $\mathbf{w}_0$. This problem is called the primal problem. The dual of this problem is:

$$\sup_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T} D(\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_T) = \sup_{\boldsymbol{\lambda}_1,\ldots,\boldsymbol{\lambda}_T} \left[-\sigma f^* \left(-\frac{1}{\sigma}\sum_{t=1}^{T} \boldsymbol{\lambda}_t\right) - \sum_{t=1}^{T} g_t^*(\boldsymbol{\lambda}_t)\right]$$

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 (Shalev-Shwartz 2007) for details on the steps to derive the dual problem.

The dual objective function $D(\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_T)$ has a nice property that makes online learning possible. Assume that for all t we have $\inf_{\mathbf{w}} g_t(\mathbf{w}) = 0$, therefore by definition of the Fenchel conjugate function we have $g_t^*(\mathbf{0}) = 0$. Then we have:

$$D(\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_t, 0, ..., 0) = D(\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_t) =$$

$$-\sigma f^* \left(-\frac{1}{\sigma}\sum_{i\leq t} \boldsymbol{\lambda}_i\right) - \sum_{i\leq t} g_i^*(\boldsymbol{\lambda}_i) \tag{2}$$

This means that if we set the $\boldsymbol{\lambda}$'s of all unseen examples to $\mathbf{0}$, then the dual objective function does not depend on the unseen cost functions $g_{t+1}, ..., g_T$, while we must know all the cost functions $g_1, ..., g_T$ to compute the primal objective value. From the weak duality theorem (Boyd and Vandenberghe 2004), 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, Shalev-Shwartz (2007) proposed the following general online incremental dual ascent algorithm (Algorithm 1):

Algorithm 1 A general incremental dual ascent algorithm for general convex loss function

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

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. $D(\boldsymbol{\lambda}_1^{t+1}, ..., \boldsymbol{\lambda}_t^{t+1}) \geq D(\boldsymbol{\lambda}_1^t, ..., \boldsymbol{\lambda}_{t-1}^t, \boldsymbol{\lambda}')$ ensures the dual objective is increased in each step. It can be shown that the regret of Algorithm 1 is $O(\sqrt{T})$.

Later, Kakade and Shalev-Shwartz (2009) extended the framework to the case of strongly convex loss functions such as the square loss or a convex loss regularized by a strongly convex function. Any strongly convex loss function can be decomposed as $l(\mathbf{w}_t) = \sigma f(\mathbf{w}_t) + g(\mathbf{w}_t)$ where f is 1-strongly convex function with respect to some norm, g is a convex function, and σ is a non-negative scalar. For the case of strongly convex loss function, the dual objective function becomes:

$$D(\boldsymbol{\lambda}_1, ..., \boldsymbol{\lambda}_t) = -(\sigma t f)^* \left(-\frac{1}{\sigma t}\sum_{i\leq t} \boldsymbol{\lambda}_i\right) - \sum_{i\leq t} g_i^*(\boldsymbol{\lambda}_i) \tag{3}$$

Algorithm 2 is the modified version of Algorithm 1 for the case of σ-strongly convex loss function. Due to the property

Algorithm 2 A general incremental dual ascent algorithm for σ-strongly convex loss function

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. $D(\boldsymbol{\lambda}_1^{t+1}, ..., \boldsymbol{\lambda}_t^{t+1}) \geq D(\boldsymbol{\lambda}_1^t, ..., \boldsymbol{\lambda}_{t-1}^t, \boldsymbol{\lambda}')$
end for

of the strongly convex loss function, the regret of Algorithm 2 is $O(\log T)$ which is much lower than that of Algorithm 1.

A simple update rule that satisfies the condition in Algorithm 1 and 2 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. $\boldsymbol{\lambda}_i^{t+1} = \boldsymbol{\lambda}_i^t,\ \forall i < t$). 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 λ_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:

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

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

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

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 (Shalev-Shwartz and Singer 2007) 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.

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, 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 $\boldsymbol{\phi}(\mathbf{x}, \mathbf{y}) : \mathcal{X} \times \mathcal{Y} \rightarrow R^d$ 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 structured prediction 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 (Taskar, Guestrin, and Koller 2004; Tsochantaridis et al. 2004)

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

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 problems 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 (Taskar et al. 2005), is only feasible for some decomposable label loss functions such as Hamming loss 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.

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. $f^*(\boldsymbol{\theta}) = \frac{1}{2}||\boldsymbol{\theta}||_2^2$ (Boyd and Vandenberghe 2004). 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 (Shalev-Shwartz and Singer)). Since both the prediction based loss and the maximal loss are generalizations of the Hinge-loss to the structure output, they 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. For the case of regularized convex loss

function, we need to solve the following optimization:

$$\text{argmax}_{\boldsymbol{\lambda}_t} \; -(\sigma t) f^*\left(-\frac{\boldsymbol{\lambda}_{1:(t-1)}+\boldsymbol{\lambda}_t}{(\sigma t)}\right) - g_t^*(\boldsymbol{\lambda}_t) \quad (4)$$

Substituting the conjugate function f^* and g_t^* as above in the equation 4, we obtain the following optimization:

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

where $\boldsymbol{\lambda}_{1:(t-1)} = \sum_{i=1}^{t-1}\boldsymbol{\lambda}_i$. This is a function of α only and in fact it is a concave parabola whose maximum achieves at the point:

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

If $\alpha^* \in [0,1]$, then α^* is the maximizer of the optimization. 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:

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

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

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

Using the same procedure, we obtain the following update for the case of the unregularized loss function:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \min\left\{\frac{1}{\sigma}, \frac{\left[\rho(\mathbf{y}_t,\mathbf{y}_t^{P|ML}) - \left\langle\mathbf{w}_t,\Delta\boldsymbol{\phi}_t^{PL|ML}\right\rangle\right]_+}{||\Delta\boldsymbol{\phi}_t^{PL|ML}||_2^2}\right\}\Delta\boldsymbol{\phi}_t^{PL|ML} \quad (5)$$

The new algorithms are presented in Algorithm 3 where CDA1, derived from Algorithm 1, uses an unregularized convex loss function, and CDA2, derived from Algorithm 2, employs a regularized convex loss function. Note that CDA1 is similar to the Passive-Aggressive (PA) algorithm 1 of Crammer et al.(2006), but PA1 was derived using a different formulation where at each step t the weight vector $\mathbf{w}_{t+1}$ is set to the solution of the following optimization problem:

$$\min_{\mathbf{w},\xi\geq 0} \quad \frac{1}{2}||\mathbf{w}-\mathbf{w}_t||_2^2 + C\xi$$

$$\text{s.t.} \quad \mathbf{w}\cdot[\boldsymbol{\phi}(\mathbf{x}_t,\mathbf{y}_t) - \boldsymbol{\phi}(\mathbf{x}_t,\mathbf{y}_t^{P|ML})] \geq \sqrt{\rho(\mathbf{y}_t,\mathbf{y}_t^{P|ML})} - \xi$$

If we set $C = 1/\sigma$, then the solution to this optimization has the same form as the update rule of CDA1 (Eq. 5). In addition, a variant of the algorithm PA1 which is identical to CDA1 with maximal loss was derived by Keshet et al.(2007). This connection shows that the algorithm PA1 is a special case of the online Coordinate-Dual-Ascend algorithm when $f(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||_2^2$ is used as the complexity function. It also gives a new interpretation for the update rule

Algorithm 3 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 (CDA1): $l_t = \left[\rho(\mathbf{y}_t,\mathbf{y}_t^P) - \langle\mathbf{w}_t,\Delta\boldsymbol{\phi}_t\rangle\right]_+$
  ML (CDA1): $l_t = \left[\rho(\mathbf{y}_t,\mathbf{y}_t^{ML}) - \langle\mathbf{w}_t,\Delta\boldsymbol{\phi}_t\rangle\right]_+$
  PL (CDA2): $l_t = \left[\rho(\mathbf{y}_t,\mathbf{y}_t^P) - \frac{t-1}{t}\langle\mathbf{w}_t,\Delta\boldsymbol{\phi}_t\rangle\right]_+$
  ML (CDA2): $l_t = \left[\rho(\mathbf{y}_t,\mathbf{y}_t^{ML}) - \frac{t-1}{t}\langle\mathbf{w}_t,\Delta\boldsymbol{\phi}_t\rangle\right]_+$
  Compute λ_t:
  CDA1: $\lambda_t = \min\{1/\sigma, \frac{l_t}{||\Delta\boldsymbol{\phi}||_2^2}\}$
  CDA2: $\lambda_t = \min\{1/(\sigma t), \frac{l_t}{||\Delta\boldsymbol{\phi}||_2^2}\}$
  Update:
  CDA1: $\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda_t\Delta\boldsymbol{\phi}_t$
  CDA2: $\mathbf{w}_{t+1} = \frac{t-1}{t}\mathbf{w}_t + \lambda_t\Delta\boldsymbol{\phi}_t$
end for

of CDA1, which is the solution of an optimization problem that tries to find a weight vector that not only scores the true label higher than the other label (the maximal loss or prediction loss label) with a high margin, but is also close (in squared distance) to the current weight vector $\mathbf{w}_t$.

We can use algorithms CDA1 and CDA2 to perform online weight learning for MLNs since the weight learning problem in MLNs can be cast as a max-margin structured prediction problem (Huynh and Mooney 2009). For MLNs, the number of true groundings of the clauses, $\mathbf{n}(\mathbf{x},\mathbf{y})$, plays the role of the joint feature function $\vec{\phi}(\mathbf{x},\mathbf{y})$.

Experimental Evaluation

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

1. How do our new online learning algorithms compare to existing online max-margin learning methods?
2. How do our new online learning algorithms compare to existing batch max-margin weight learning methods?
3. Is CDA2 better than CDA1 in practice?
4. How well does the the prediction based loss compare to the maximal loss in practice?

Datasets

We ran experiments on two large, real-world datasets: the CiteSeer dataset (Lawrence, Giles, and Bollacker 1999) for bibliographic citation segmentation, and a web search query dataset obtained from Microsoft Research for query disambiguation.

For CiteSeer, we used the dataset and the MLN of Poon and Domingos(2007). The dataset has 1,563 citations and

each of them is segmented into three fields: *Author*, *Title* and *Venue*. The dataset has four disjoint subsets corresponding to four different research topics. We used the simplest MLN, the isolated segmentation model, of Poon and Domingos(2007) for learning.

For the search query disambiguation, we used the data created by Mihalkova and Mooney(2009). 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(2009) 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.

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(2009).

1-best MIRA: MIRA is one of the first online learning algorithm for structured prediction proposed by Crammer, McDonald, and Pereira (2005). A simple version of MIRA, called 1-best MIRA, is widely used in practice since its update rule has a closed-form solution. Riedel and Meza-Ruiz(2008) used 1-best MIRA to learn weights for MLNs. The update rule of 1-best MIRA is similar to that of the CDA1 with prediction based loss. 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, Bagnell, and Zinkevich (2007) is an extension of the Greedy Projection algorithm (Zinkevich 2003) to the case of structured prediction. The update rule of this algorithm is similar to that of the CDA2 with maximal loss. Using the learning rate $\alpha_t = 1/(\sigma t)$, the algorithm updates the weight vectors $\mathbf{w}$ at each step as follows:

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

CDA1 and CDA2: The two derived online learning algorithms presented in Algorithm 3.

In training, we used the exact MPE inference based on Integer Linear Programming described by Huynh and Mooney (2009) for online learning algorithms since exact inference on each example is feasible on these datasets. For the offline weight learner MM, we used the approximate inference algorithm developed by Huynh and Mooney (2009) since it is computationally intractable to run exact inference for all training examples at once. In testing, we used exact MPE inference for Citeseer and used MCSAT to compute marginal probabilities for the web search query dataset since we want to rank the query results. The parameter σ of the Subgradient, CDA1 and CDA2 is set based on the performance on the validation set. All experiments used Hamming loss as the label loss function. 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).

Table 1: Average F_1 and training time on CiteSeer dataset

Algorithms	Average F_1	Average training time
MM-HM	93.433 ± 1.41	90.282 min.
1-best-MIRA-HM	93.056 ± 3.18	11.772 min.
Subgradient-HM	91.910 ± 2.37	12.655 min.
CDA1-PL-HM	92.945 ± 2.94	11.824 min.
CDA1-ML-HM	94.131 ± 2.25	12.738 min.
CDA2-PL-HM	92.836 ± 2.55	11.869 min.
CDA2-ML-HM	94.375 ± 2.13	12.887 min.

Table 2: MAP scores on Microsoft search query dataset

Algorithms	MLN1	MLN2	MLN3
CD	0.375	0.386	0.366
1-best-MIRA	0.366	0.375	0.379
Subgradient	0.374	0.397	0.396
CDA1-PL	0.380	0.395	0.396
CDA1-ML	0.379	0.389	0.385
CDA2-PL	0.382	0.397	0.398
CDA2-ML	0.380	0.397	0.397

Metrics

Like previous work, we used F_1, the harmonic mean of recall and precision, to measure the performance of each algorithm on Citeseer, and for the web query we used the MAP (Mean Average Precision) which measures how close the relevant results are to the top of the ranking.

Results and Discussion

Table 1 presents the average F_1 scores of different algorithms on Citeseer. On this dataset, CDA algorithms with maximal loss, CDA1-ML and CDA2-ML, have the best average F_1 scores, and these results are better than those of 1-best MIRA and the subgradient method (the differences with respect to the subgradient method are statistically significant). The F_1 scores of CDA1-ML and CDA2-ML are also better than that of the batch max-margin algorithm (MM) since the batch learner can only uses approximate inference in training. Other advantages of online algorithms are in terms of training time and memory. On this dataset, the 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. Since online algorithms process one example at a time, they use much less memory than batch methods. Between CDA1 and CDA2, CDA2-ML is a little bit better but the difference is not significant. Regarding the comparison between maximal loss and prediction-based loss, the former is significantly better than latter on this dataset.

Table 2 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(2009) who used a variant of the structured perceptron (Collins 2002) called Contrastive Divergence (CD) (Hinton 2002) to do online weight learning for MLNs. It is clear that the CDA algorithms have 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 CDA1 and CDA2 are not significantly better than that of the subgradient method, but their performance is more consistent across the three MLNs. CDA2 is better than CDA1 on this dataset since CDA2-ML is significantly better than CDA1-ML and CDA2-PL is comparable to CDA1-PL. Regarding the loss function, in contrast to the results on Citeseer, the CDA algorithms with prediction-based loss are a little better than the ones with maximal loss especially for the case of CDA1, since aggressive update can lead to overfitting on noisy datasets.

In summary, our new online learning algorithms CDA1 and CDA2 have generally better performance 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. CDA2 is a little better than CDA1 especially on noisy datasets. Between the maximal loss and the prediction-based loss, the maximal loss is better than the prediction-based loss on clean datasets, but the latter is a bit better than the former on noisy datasets.

Conclusions and Future Work

We have presented a comprehensive study of online weight learning for MLNs. Based on the primal-dual framework, we derived new online algorithms for structured prediction and applied them to learn weights for MLNs and compared them to existing online methods. Our new algorithms generally achieved better accuracy than existing online methods on two large, real-world datasets. Since the new algorithms are not specific to MLNs, it would be interesting to apply them to other structured prediction models.

Acknowledgments

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.

References

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. Cambridge University Press.

Collins, M. 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In *EMNLP'02*, 1–8.

Crammer, K.; Dekel, O.; Keshet, J.; Shalev-Shwartz, S.; and Singer, Y. 2006. Online passive-aggressive algorithms. *JMLR* 7:551–585.

Crammer, K.; McDonald, R.; and Pereira, F. 2005. Scalable large-margin online learning for structured classification. Technical report, Dep. of Computer and Information Science, Uni. of Pennsylvania.

Getoor, L., and Taskar, B., eds. 2007. *Statistical Relational Learning*. Cambridge, MA: MIT Press.

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

Huynh, T. N., and Mooney, R. J. 2008. Discriminative structure and parameter learning for Markov logic networks. In *ICML-08*, 416–423.

Huynh, T. N., and Mooney, R. J. 2009. Max-margin weight learning for Markov logic networks. In *ECML PKDD 2009, Part I*, 564–579.

Kakade, S. M., and Shalev-Shwartz, S. 2009. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *NIPS-08*, 1457–1464.

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

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

Lawrence, S.; Giles, C. L.; and Bollacker, K. D. 1999. Autonomous citation matching. In *Proceedings of the third annual conference on Autonomous Agents*.

Lowd, D., and Domingos, P. 2007. Efficient weight learning for Markov logic networks. In *PKDD-2007*, 200–211.

Mihalkova, L., and Mooney, R. J. 2009. Learning to disambiguate search queries from short sessions. In *ECML PKDD 2009, Part II*, 111–127.

Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI-2006*.

Poon, H., and Domingos, P. 2007. Joint inference in information extraction. In *AAAI-07*, 913–918.

Ratliff, N.; Bagnell, J. A.; and Zinkevich, M. 2007. (Online) subgradient methods for structured prediction. In *AISTATS-07*.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *MLJ* 62:107–136.

Riedel, S., and Meza-Ruiz, I. 2008. Collective semantic role labelling with Markov logic. In *CoNLL'08*.

Riedel, S. 2008. Improving the accuracy and efficiency of MAP inference for Markov logic. In *UAI-08*, 468–475.

Shalev-Shwartz, S., and Singer, Y. Convex repeated games and Fenchel duality. In *NISP-06*.

Shalev-Shwartz, S., and Singer, Y. 2007. A unified algorithmic approach for efficient online label ranking. In *AISTATS-07*.

Shalev-Shwartz, S. 2007. *Online Learning: Theory, Algorithms, and Applications*. Ph.D. Dissertation, The Hebrew Uni. of Jerusalem.

Singla, P., and Domingos, P. 2005. Discriminative training of Markov logic networks. In *AAAI-05*, 868–873.

Singla, P., and Domingos, P. 2008. Lifted first-order belief propagation. In *AAAI-08*, 1094–1099.

Taskar, B.; Chatalbashev, V.; Koller, D.; and Guestrin, C. 2005. Learning structured prediction models: a large margin approach. In *ICML-05*, 896–903.

Taskar, B.; Guestrin, C.; and Koller, D. 2004. Max-margin Markov networks. In *NISP-03*.

Tsochantaridis, I.; Hofmann, T.; Joachims, T.; and Altun, Y. 2004. Support vector machine learning for interdependent and structured output spaces. In *ICML-04*, 104–112.

Zinkevich, M. 2003. Online Convex Programming and Generalized Infinitesimal Gradient Ascent. In *ICML-03*, 928–936.