Learning Structural SVMs with Latent Variables

Chun-Nam John Yu CNYU@CS.CORNELL.EDU
Thorsten Joachims TJ@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14850 USA

Abstract

We present a large-margin formulation and algorithm for structured output prediction that allows the use of latent variables. Our proposal covers a large range of application problems, with an optimization problem that can be solved efficiently using Concave-Convex Programming. The generality and performance of the approach is demonstrated through three applications including motif-finding, noun-phrase coreference resolution, and optimizing precision at k in information retrieval.

1. Introduction

In many structured prediction tasks, there is useful modeling information that is not available as part of the training data $(x_1, y_1), ..., (x_n, y_n)$. In noun phrase coreference resolution, for example, one is typically given the clustering y of noun-phrases for a training document x, but not the set of informative links that connects the noun phrases together into clusters. Similarly, in machine translation, one may be given the translation y of sentence x, but not the linguistic structure h (e.g. parse trees, word alignments) that connects them. This missing information h, even if not observable, is crucial for expressing high-fidelity models for these tasks. It is important to include these information in the model as latent variables.

Latent variables have long been used to model observations in generative probabilistic models such as Hidden Markov Models. In discriminative models, however, the use of latent variables is much less explored. Recently, there has been some work on Conditional Random Fields (Wang et al., 2006) with latent variables. Even less explored is the use of latent variables in large-margin structured output learning such as Max-Margin Markov Networks or Structural SVMs (Taskar et al., 2003; Tsochantaridis et al., 2004). While these non-probabilistic models offer excellent performance on many structured prediction tasks in the fully observed case, they currently do not support the use of latent variables, which excludes many interesting applications.

In this paper, we propose an extension of the Structural SVM framework to include latent variables. We identify a particular, yet rather general, formulation for which there exists an efficient algorithm to find a local optimum using the Concave-Convex Procedure. The resulting algorithm is similarly modular as the Structural SVM algorithms for the fully observed case. To illustrate the generality of our Latent Structural SVM algorithm, we provide experimental results on three different applications in computational biology, natural language processing, and information retrieval.

1.1. Related Works

Many of the early works in introducing latent variables into discriminative models were motivated by computer vision applications, where it is natural to use latent variables to model human body parts or parts of objects in detection tasks. The work in (Wang et al., 2006) introduces Hidden Conditional Random Fields, a discriminative probabilistic latent variable model for structured prediction, with applications to two computer vision tasks. In natural language processing there is also work in applying discriminative probabilistic latent variable models, for example the training of PCFG with latent annotations in a discriminative manner (Petrov & Klein, 2007). The non-convex likelihood functions of these problems are usually optimized using gradient-based methods.

The Concave-Convex Procedure (Yuille & Rangarajan, 2003) employed in our work is a general framework for minimizing non-convex functions which falls into the class of DC (Difference of Convex) programming. In recent years there have been numerous appli-

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

cations of the algorithm in machine learning, including training non-convex SVMs and transductive SVMs (Collobert et al., 2006). The approach in (Smola et al., 2005) employs CCCP to handle missing data in SVMs and Gaussian Processes and is closely related to our work. However our approach is non-probabilistic and avoids the computation of partition functions, which is particularly attractive for structured prediction. Very recently the CCCP algorithm has also been applied to obtain tighter non-convex loss bounds on structured learning (Chapelle et al., 2008).

In the computer vision community there are recent works on training Hidden CRF using the max-margin criterion (Felzenszwalb et al., 2008; Wang & Mori, 2008). In these works they focus on classification problems only and their training problem formulations are a special case of our proposal below. Interestingly, the algorithm in (Felzenszwalb et al., 2008) coincides with our approach for binary classification but was derived in a different way.

2. Structural SVMs

Suppose we are given a training set of input-output structure pairs $\mathcal{S}=\{(x_1,y_1),\ldots,(x_n,y_n)\}\in(\mathcal{X}\times\mathcal{Y})^n$. We want to learn a linear prediction rule of the form

$$f_{\boldsymbol{w}}(x)=\operatorname{argmax}_{y\in\mathcal{Y}}\left[\boldsymbol{w}\cdot\Phi(x,y)\right], \tag{1}$$

where Φ is a joint feature vector that describes the relationship between input x and structured output y, with $\boldsymbol{w}$ being the parameter vector. The optimization problem of computing this argmax is typically referred to as the "inference" or "prediction" problem.

When training Structural SVMs, the parameter vector $\boldsymbol{w}$ is determined by minimizing the (regularized) risk on the training set $(x_1,y_1),...,(x_n,y_n)$. Risk is measured through a user-supplied loss function $\Delta(y,\hat{y})$ that quantifies how much the prediction $\hat{y}$ differs from the correct output y. Note that Δ is typically non-convex and discontinuous and there are usually exponentially many possible structures $\hat{y}$ in the output space $\mathcal{Y}$. The Structural SVM formulation (Tsochantaridis et al., 2004) overcomes these difficulties by replacing the loss function Δ with a piecewise linear convex upper bound (margin rescaling)

$$\Delta(y_i,\hat{y}_i(\boldsymbol{w}))\leq\max_{\hat{y}\in\mathcal{Y}}[\Delta(y_i,\hat{y})+\boldsymbol{w}\cdot\Phi(x_i,\hat{y})]-\boldsymbol{w}\cdot\Phi(x_i,y_i)$$

where $\hat{y}_i(\boldsymbol{w})=\operatorname{argmax}_{y\in\mathcal{Y}}\,\boldsymbol{w}\cdot\Phi(x_i,y)$.

To train Structural SVMs we then solve the following convex optimization problem:

$$\min_{\boldsymbol{w}}\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{n}\left[\max_{\hat{y}\in\mathcal{Y}}[\Delta(y_i,\hat{y})+\boldsymbol{w}\cdot\Phi(x_i,\hat{y})]-\boldsymbol{w}\cdot\Phi(x_i,y_i)\right].$$

Despite the typically exponential size of $\mathcal{Y}$, this optimization problem can be solved efficiently using cutting-plane or stochastic gradient methods. Structural SVMs give excellent performance on many structured prediction tasks, especially when the model Φ is high-dimensional and it is necessary to optimize to non-standard loss functions Δ.

3. Structural SVM with Latent Variables

As argued in the introduction, however, in many applications the input-output relationship is not completely characterized by $(x,y)\in\mathcal{X}\times\mathcal{Y}$ pairs in the training set alone, but also depends on a set of unobserved latent variables $h\in\mathcal{H}$. To generalize the Structural SVM formulation, we extend our joint feature vector $\Phi(x,y)$ with an extra argument h to $\Phi(x,y,h)$ to describe the relation among input x, output y, and latent variable h. We want to learn a prediction rule of the form

$$f_{\boldsymbol{w}}(x)=\operatorname{argmax}_{(y,h)\in\mathcal{Y}\times\mathcal{H}}\left[\boldsymbol{w}\cdot\Phi(x,y,h)\right]. \tag{2}$$

At first glance, a natural way to extend the loss function Δ is to again include the latent variables $h\in\mathcal{H}$ similar to above, to give

$$\Delta((y_i,h_i^*(\boldsymbol{w})),(\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w}))),$$

where $h_i^*(\boldsymbol{w})=\operatorname{argmax}_{h\in\mathcal{H}}\,\boldsymbol{w}\cdot\Phi(x_i,y_i,h)$ and $(\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w}))=\operatorname{argmax}_{(y,h)\in\mathcal{Y}\times\mathcal{H}}\,\boldsymbol{w}\cdot\Phi(x_i,y,h)$.

Essentially, this extended loss measures the difference between the pair $(\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w}))$ given by the prediction rule and the best latent variable $h_i^*(\boldsymbol{w})$ that explains the input-output pair (x_i,y_i) in the training set. Like in the fully observed case, we can derive a hinge-loss style upper bound

$$\begin{aligned}
&\Delta((y_i,h_i^*(\boldsymbol{w})),(\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w})))\\
\leq\;&\Delta((y_i,h_i^*(\boldsymbol{w})),(\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w})))\\
&-[\boldsymbol{w}\cdot\Phi(x_i,y_i,h_i^*(\boldsymbol{w}))-\boldsymbol{w}\cdot\Phi(x_i,\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w}))]\\
=\;&\left(\max_{(\hat{y},\hat{h})\in\mathcal{Y}\times\mathcal{H}}\boldsymbol{w}\cdot\Phi(x_i,\hat{y},\hat{h})\right)+\Delta((y_i,h_i^*(\boldsymbol{w})),(\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w})))\\
&-\left(\max_{h\in\mathcal{H}}\boldsymbol{w}\cdot\Phi(x_i,y_i,h)\right).
\end{aligned} \tag{3}$$

In the case of Structural SVMs without latent variables, the complex dependence on $\boldsymbol{w}$ within the loss Δ can be removed using the following inequality, commonly referred to as "loss-augmented inference" in

Structural SVM training:

$$\begin{aligned}&\left(\max_{\hat{y}\in\mathcal{Y}} \boldsymbol{w}\cdot\Phi(x,\hat{y})\right)+\Delta(y_i,\hat{y}_i(\boldsymbol{w}))\\ \leq\ &\max_{\hat{y}\in\mathcal{Y}}[\boldsymbol{w}\cdot\Phi(x_i,\hat{y}_i)+\Delta(y_i,\hat{y})].\end{aligned}\tag{4}$$

When latent variables are included, however, the dependence of Δ on the latent variables $h_i^*(\boldsymbol{w})$ of the correct label y_i prevents us from applying this trick.

To circumvent this difficulty, let us rethink the definition of loss function from Equation (3). As we will see below, many real world applications do not require the loss functions to depend on the offending $h_i^*(\boldsymbol{w})$. In applications such as parsing and object recognition, the latent variables serve as indicator for mixture components or intermediate representations and are not part of the output. As a result, the natural loss functions that we are interested in for these tasks usually do not depend on the latent variables.

We therefore focus on the case where the loss function Δ does not depend on the latent variable $h_i^*(\boldsymbol{w})$:

$$\Delta((y_i,h_i^*(\boldsymbol{w})),(\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w})))=\Delta(y_i,\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w})).$$

Note that however the loss function may still depend on $\hat{h}_i(\boldsymbol{w})$. In the case where the latent variable $\hat{h}$ is a part of the prediction which we care about we can still define useful asymmetric loss functions $\Delta(y,\hat{y},\hat{h})$ for learning. The applications of noun phrase coreference resolution and optimizing for precision@k in document retrieval in the experiments section below are good examples of this.

With the redefined loss $\Delta(y_i,\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w}))$, the bound in Equation (3) becomes

$$\begin{aligned}&\Delta((y_i,h_i^*(\boldsymbol{w})),(\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w})))\\ \leq&\left(\max_{(\hat{y},\hat{h})\in\mathcal{Y}\times\mathcal{H}}[\boldsymbol{w}\cdot\Phi(x_i,\hat{y},\hat{h})+\Delta(y_i,\hat{y},\hat{h})]\right)-\left(\max_{h\in\mathcal{H}}\boldsymbol{w}\cdot\Phi(x_i,y_i,h)\right).\end{aligned}$$

Using the same reasoning as for fully observed Structural SVMs, this gives rise to the following optimization problem for Structural SVMs with latent variables:

$$\begin{aligned}\min_{\boldsymbol{w}}\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{n}&\left(\max_{(\hat{y},\hat{h})\in\mathcal{Y}\times\mathcal{H}}[\boldsymbol{w}\cdot\Phi(x_i,\hat{y},\hat{h})+\Delta(y_i,\hat{y},\hat{h})]\right)\\ &-C\sum_{i=1}^{n}\left(\max_{h\in\mathcal{H}}\boldsymbol{w}\cdot\Phi(x_i,y_i,h)\right).\end{aligned}\tag{5}$$

It is easy to observe that the above formulation reduces to the usual Structural SVM formulation in the absence of latent variables. The formulation can also be easily extended to include kernels, although the usual extra cost of computing inner products in nonlinear kernel feature space applies.

Finally, note that the redefined loss distinguishes our approach from transductive structured output learning (Zien et al., 2007). When the loss Δ depends only on the fully observed label y_i, it rules out the possibility of transductive learning, but the restriction also results in simpler optimization problems compared to the transductive cases (for example, the approach in (Zien et al., 2007) involves constraint removals to deal with dependence on $h_i^*(\boldsymbol{w})$ within the loss Δ).

4. Solving the Optimization Problem

A key property of Equation (5) that follows from the redefined loss $\Delta(y_i,\hat{y}_i(\boldsymbol{w}),\hat{h}_i(\boldsymbol{w}))$ is that it can be written as the difference of two convex functions:

$$\begin{aligned}\min_{\boldsymbol{w}}&\left[\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{n}\max_{(\hat{y},\hat{h})\in\mathcal{Y}\times\mathcal{H}}[\boldsymbol{w}\cdot\Phi(x_i,\hat{y},\hat{h})+\Delta(y_i,\hat{y},\hat{h})]\right]\\ &-\left[C\sum_{i=1}^{n}\max_{h\in\mathcal{H}}\boldsymbol{w}\cdot\Phi(x_i,y_i,h)\right].\end{aligned}$$

This allows us to solve the optimization problem using the Concave-Convex Procedure (CCCP) (Yuille & Rangarajan, 2003). The general template for a CCCP algorithm for minimizing a function $f(\boldsymbol{w})-g(\boldsymbol{w})$, where f and g are convex, works as follows:

Algorithm 1 Concave-Convex Procedure (CCCP)

1: Set $t=0$ and initialize $\boldsymbol{w}_0$
2: **repeat**
3: Find hyperplane $\boldsymbol{v}_t$ such that $-g(\boldsymbol{w})\leq -g(\boldsymbol{w}_t)+(\boldsymbol{w}-\boldsymbol{w}_t)\cdot\boldsymbol{v}_t$ for all $\boldsymbol{w}$
4: Solve $\boldsymbol{w}_{t+1}=\operatorname{argmin}_{\boldsymbol{w}} f(\boldsymbol{w})+\boldsymbol{w}\cdot\boldsymbol{v}_t$
5: Set $t=t+1$
6: **until** $[f(\boldsymbol{w}_t)-g(\boldsymbol{w}_t)]-[f(\boldsymbol{w}_{t-1})-g(\boldsymbol{w}_{t-1})]<\epsilon$

The CCCP algorithm is guaranteed to decrease the objective function at every iteration and to converge to a local minimum or saddle point (Yuille & Rangarajan, 2003). Line 3 constructs a hyperplane that upper bounds the concave part of the objective $-g$, so that the optimization problem solved at line 4 is convex.

In terms of the optimization problem for Latent Structural SVM, the step of computing the upper bound for the concave part in line 3 involves computing

$$h_i^*=\operatorname{argmax}_{h\in\mathcal{H}}\boldsymbol{w}_t\cdot\Phi(x_i,y_i,h)\tag{6}$$

for each i. We call this the "latent variable completion" problem. The hyperplane constructed is $\boldsymbol{v}_t=\sum_{i=1}^{n}\Phi(x_i,y_i,h_i^*)$.

Computing the new iterate $\boldsymbol{w}_{t+1}$ in line 1 involves solving the standard Structural SVM optimization problem by completing y_i with the latent variables h_i^* as if they were completely observed:

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n} \max_{(\hat{y},\hat{h})\in\mathcal{Y}\times\mathcal{H}} [\boldsymbol{w}\cdot\Phi(x_i,\hat{y},\hat{h}) + \Delta(y_i,\hat{y},\hat{h})] - C\sum_{i=1}^{n} \boldsymbol{w}\cdot\Phi(x_i,y_i,h_i^*). \quad (7)$$

Thus the CCCP algorithm applied to Structural SVM with latent variables gives rise to a very intuitive algorithm that alternates between imputing the latent variables h_i^* that best explain the training pair (x_i, y_i) and solving the Structural SVM optimization problem while treating the latent variables as completely observed. This is similar to the iterative process of Expectation Maximization (EM). But unlike EM which maximizes the expected log likelihood under the marginal distribution of the latent variables, we are minimizing the regularized loss against a single latent variable h_i^* that best explains (x_i, y_i).

In our implementation, we used an improved version of the cutting plane algorithm called the proximal bundle method (Kiwiel, 1990) to solve the standard Structural SVM problem in Equation (7). In our experience the proximal bundle method usually converges using fewer iterations than the cutting plane algorithm (Joachims et al., To appear) in the experiments below. The algorithm also fits very nicely into the CCCP algorithmic framework when it is employed to solve the standard Structural SVM optimization problem inside the CCCP loop. The solution $\boldsymbol{w}_{t-1}$ from the last iteration can be used as a starting point in a new CCCP iteration, without having to reconstruct all the cuts from scratch. We will provide some computational experience at the end of the experiments section.

5. Experiments

Below we demonstrate three applications of our Latent Structural SVM algorithm. Some of them have been discussed in the machine learning literature before, but we will show that our Latent Structural SVM framework provides new and straightforward solution approaches with good predictive performance. A software package implementing the Latent Structural SVM algorithm is available for download at `http://www.cs.cornell.edu/~cnyu/latentssvm/`.

5.1. Discriminative Motif Finding

Our development of the Latent Structural SVM was motivated by a motif finding problem in yeast DNA through collaboration with computational biologists. Motifs are repeated patterns in DNA sequences that are believed to have biological significance. Our dataset consists of ARSs (autonomously replicating sequences) screened in two yeast species S. kluyveri and S. cerevisiae. Our task is to predict whether a particular sequence is functional (i.e., whether they start the replication process) in S. cerevisiae and to find out the motif responsible. All the native ARSs in S. cerevisiae are labeled as positive, since by definition they are functional. The ones that showed ARS activity in S. kluyveri were then further tested to see whether they contain functional ARS in S. cerevisiae, since they might have lost their function due to sequence divergence of the two species during evolution. They are labeled as positive if functional and negative otherwise. In this problem the latent variable h is the position of the motif in the positive sequences, since current experimental procedures do not have enough resolution to pinpoint their locations. Altogether we have 124 positive examples and 75 negative examples. In addition we have 6460 sequences from the yeast intergenic regions for background model estimation.

Popular methods for motif finding includes methods based on EM (Bailey & Elkan, 1995) and Gibbs-sampling. For this particular yeast dataset we believe a discriminative approach, especially one incorporating large-margin separation, is beneficial because of the close relationship and DNA sequence similarity among the different yeast species in the dataset.

Let x_i denote the ith base (A, C, G, or T) in our input sequence x of length n. We use the common position-specific weight matrix plus background model approach in our definition of feature vector:

$$\Psi(x,y,h) = \begin{cases} \sum_{j=1}^{l} \phi_{PSM}^{(j)}(x_{h+j}) + \sum_{i=1}^{h} \phi_{BG}(x_i) + \sum_{i=h+l+1}^{n} \phi_{BG}(x_i) & [\text{if } y=+1] \\ \sum_{i=1}^{n} \phi_{BG}(x_i) & [\text{if } y=-1], \end{cases}$$

where $\phi_{PSM}^{(j)}$ is the feature count for the jth position of the motif in the position-specific weight matrix, and ϕ_{BG} is the feature count for the background model (we use a Markov background model of order 3).

For the positive sequences, we randomly initialized the motif position h uniformly over the whole length of the sequence. We optimized over the zero-one loss Δ for classification and performed a 10-fold cross validation. We make use of the set of 6460 intergenic sequences in training by treating them as negative examples (but they are excluded in the test sets). Instead of penalizing their slack variables by C in the objective we only penalize these examples by $C/50$ to avoid

Table 1. Classification Error on Yeast DNA (10-fold CV)

	Error rate
Gibbs sampler ($l = 11$)	32.49%
Gibbs sampler ($l = 17$)	31.47%
Latent Structural SVM ($l = 11$)	11.09%
Latent Structural SVM ($l = 17$)	12.00%

overwhelming the training set with negative examples (with the factor 1/50 picked by cross-validation). We trained models using regularization constant C from $\{0.1, 1, 10, 100, 1000\}$ times the size of the training set (5992 for each fold), and each model is re-trained 10 times using 10 different random seeds.

As control we ran a Gibbs sampler (Ng & Keich, 2008) on the same dataset, with the same set of intergenic sequences for background model estimation. It reports good signals on motif lengths $l = 11$ and $l = 17$, which we compare our algorithm against. To provide a stronger baseline we optimize the classification threshold of the Gibbs sampler on the test set and report the best accuracy over all possible thresholds. Table 1 compares the accuracies of the Gibbs sampler and our method averaged across 10 folds. Our algorithm shows a significant improvement over the Gibbs sampler (with p-value $< 10^{-4}$ in a paired t-test). As for the issue of local minima, the standard deviations on the classification error over the 10 random seeds, averaged over 10 folds, are 0.0648 for $l = 11$ and 0.0546 for $l = 17$. There are variations in solution quality due to local minima in the objective, but they are relatively mild in this task and can be overcome with a few random restarts.

In this application the Latent Structural SVM allows us to exploit discriminative information to better detect motif signals compared to traditional unsupervised probabilistic model for motif finding. Currently we are working with our collaborators on ways to interpret the position-specific weight matrix encoded in the weight vector trained by the Latent Structural SVM.

5.2. Noun Phrase Coreference via Clustering

In noun phrase coreference resolution we would like to determine which noun phrases in a text refer to the same real-world entity. In (Finley & Joachims, 2005) the task is formulated as a correlation clustering problem trained with Structural SVMs. In correlation clustering the objective function maximizes the sum of pairwise similarities. However this might not be the most appropriate objective, because in a cluster of coreferent noun phrases of size k, many of the $O(k^2)$ links contain only very weak signals. For example, it is difficult to determine whether a mention of the name 'Tom' at the beginning of a text and a pronoun 'he' at the end of the text are coreferent directly without scanning through the whole text.

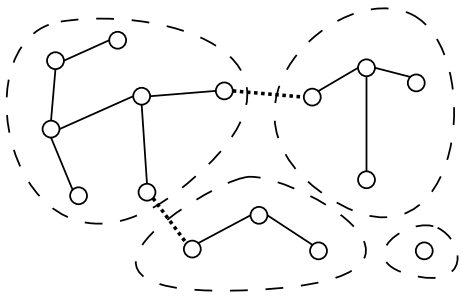

Figure 1. The circles are the clusters defined by the label y. The set of solid edges is one spanning forest h that is consistent with y. The dotted edges are examples of incorrect links that will be penalized by the loss function.

Following the intuition that humans might determine if two noun phrases are coreferent by reasoning transitively over strong coreference links (Ng & Cardie, 2002), we model the problem of noun phrase coreference as a single-link agglomerative clustering problem. Each input x contains all n noun phrases in a document, and all the pairwise features $\boldsymbol{x}_{ij}$ between the ith and jth noun phrases. The label y is a partition of the n noun phrases into coreferent clusters. The latent variable h is a spanning forest of 'strong' coreference links that is consistent with the clustering y. A spanning forest h is consistent with a clustering y if every cluster in y is a connected component in h (i.e., a tree), and there are no edges in h that connects two distinct clusters in y (Figure 1).

To score a clustering y with a latent spanning forest h, we use a linear scoring model that adds up all the edge scores for edges in h, parameterized by $\boldsymbol{w}$:

$$\boldsymbol{w} \cdot \Phi(x, y, h) = \sum_{(i,j) \in h} \boldsymbol{w} \cdot \boldsymbol{x}_{ij}.$$

To predict a clustering y from an input x (argmax in Equation (2)), we can run any Maximum Spanning Tree algorithm such as Kruskal's algorithm on the complete graph of n noun phrases in x, with edge weights defined by $\boldsymbol{w} \cdot \boldsymbol{x}_{ij}$. The output h is a spanning forest instead of a spanning tree because two trees will remain disconnected if all edges connecting the two trees have negative weights. We then output the clustering defined by the forest h as our prediction y.

For the loss function Δ, we would like to pick one that supports efficient computation in the loss-augmented inference, while at the same time penalizing incorrect spanning trees appropriately for our application. We

Table 2. Clustering Accuracy on MUC6 Data

	MITRE Loss	Pair Loss
SVM-cluster	41.3	2.89
Latent Structural SVM	44.1	2.66
Latent Structural SVM (modified loss, $r = 0.01$)	**35.6**	4.11

propose the loss function

$$\Delta(y, \hat{y}, \hat{h}) = n(y) - k(y) - \sum_{(i,j)\in\hat{h}} l(y, (i, j)), \qquad (8)$$

where $n(y)$ and $k(y)$ are the number of vertices and the number of clusters in the correct clustering y. The function $l(y, (i, j))$ returns 1 if i and j are within the same cluster in y, and -1 otherwise. It is easy to see that this loss function is non-negative and zero if and only if the spanning forest $\hat{h}$ defines the same clustering as y. Since this loss function is linearly decomposable into the edges in $\hat{h}$, the loss-augmented inference can also be computed efficiently using Kruskal's algorithm. Similarly the step of completing the latent variable h given a clustering y, which involves computing a highest scoring spanning forest that is consistent with y, can also be done with the same algorithm.

To evaluate our algorithm, we performed experiments on the MUC6 noun phrase coreference dataset. There are 60 documents in the dataset and we use the first 30 for training and the remaining 30 for testing. The pairwise features $\boldsymbol{x}_{ij}$ are the same as those in (Ng & Cardie, 2002). The regularization parameter C is picked from 10^{-2} to 10^{6} using a 10-fold cross validation procedure. The spanning forest h for each correct clustering y is initialized by connecting all coreferent noun phrases in chronological order (the order in which they appear in the document), so that initially each tree in the spanning forest is a linear chain.

Table 2 shows the result of our algorithm compared to the SVM correlation clustering approach in (Finley & Joachims, 2005). We present the results using the same loss functions as in (Finley & Joachims, 2005). Pair loss is the proportion of all $O(n^2)$ edges incorrectly classified. MITRE loss is a loss proposed for evaluating noun phrase coreference that is related to the F_1-score (Vilain et al., 1995).

We can see from the first two lines in the table that our method performs well on the Pair loss but worse on the MITRE loss when compared with the SVM correlation clustering approach. Error analysis reveals that our method trained with the loss defined by Equation (8) is very conservative when predicting links between noun phrases, having high precision but rather low recall. Therefore we adapt our loss function to make it more suitable for minimizing the MITRE loss. We modified the loss function in Equation (8) to penalize less for adding edges that incorrectly link two distinct clusters, using a penalty $r < 1$ instead of 1 for each incorrect edge added. With the modified loss (with $r = 0.01$ picked via cross-validation) our method performs much better than the SVM correlation clustering approach on the MITRE loss (p-value < 0.03 in a Z-test).

Unlike the SVM correlation clustering approach, where approximate inference is required, our inference procedure involves only simple and efficient maximum spanning tree calculations. For this noun phrase coreference task, the new formulation with Latent Structural SVM improves both the prediction performance and training efficiency over conventional Structural SVMs.

5.3. Optimizing for Precision@k in Ranking

Our last example application is related to optimizing for precision@k in document retrieval. Precision@k is defined to be the number of relevant documents in the top k positions given by a ranking, divided by k. For each example in the training set, the pattern x is a collection of n documents $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ associated with a query q, and the label $y \in \{-1, 1\}^n$ classifies whether each document in the collection is relevant to the query or not. However for the purpose of evaluating and optimizing for information retrieval performance measures such as precision@k and NDCG@k, the partial order of the documents given by the label y is insufficient. The label y does not tell us which the top k documents are. To deal with this problem, we can postulate the existence of a latent total order h on all documents related to the query, with h consistent with the partial order given by label y. To be precise, let h_j be the index of the jth most relevant document, such that $\boldsymbol{x}_{h_j} \geq_{tot} \boldsymbol{x}_{h_{j+1}}$ for j from 1 to $n-1$, where $\geq_{tot}$ is a total order of relevance on the documents $\boldsymbol{x}_i$, and let $>_{tot}$ be its strict version. The label y is consistent with the latent variable h if $y_i > y_j$ implies $\boldsymbol{x}_i >_{tot} \boldsymbol{x}_j$, so that all relevant documents in y comes before the non-relevant documents in the total order h. For optimizing for precision@k in this section, we can restrict h to be first k documents $h_1, \ldots, h_k$.

We use the following construction for the feature vector (in a linear feature space):

$$\Phi(x, y, h) = \frac{1}{k} \sum_{j=1}^{k} \boldsymbol{x}_{h_j}.$$

The feature vector only consists of contributions from the top k documents selected by h, when all other documents in the label y are ignored (with the restriction that h has to be consistent with y).

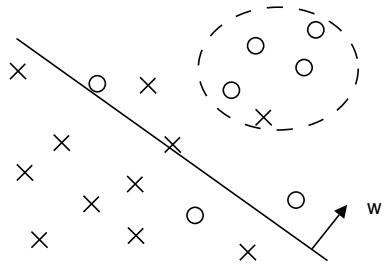

Figure 2. Latent Structural SVM tries to optimize for accuracy near the region for the top k documents (circled), when a good general ranking direction $\boldsymbol{w}$ is given

For the loss we use the following precision@k loss:

$$\Delta(y, \hat{y}, \hat{h}) = \min\{1, \frac{n(y)}{k}\} - \frac{1}{k}\sum_{j=1}^{k}[y_{h_j} == 1].$$

This loss function is essentially one minus precision@k, with slight modifications when there are less than k relevant documents in a collection. We replace 1 by $n(y)/k$ so that the loss can be minimized to zero, where $n(y)$ is the total number of relevant documents in y.

Intuitively, with this particular design of the feature vector and the loss function, the algorithm is trying to optimize for the classification accuracy in the region near the top k documents, while ignoring most of the documents in the rest of the feature space (Figure 2).

All the inference problems required for this application are efficient to solve. Prediction requires sorting based on the score $\boldsymbol{w} \cdot \boldsymbol{x}_j$ in decreasing order and picking the top k. The loss-augmented inference requires sorting based on the score $\boldsymbol{w} \cdot \boldsymbol{x}_j - [y_j == 1]$ and picking the top k for $\hat{h}$. Latent variable completion for y requires a similar sorting procedure on $\boldsymbol{w} \cdot \boldsymbol{x}_j$ and picking the top k, but during sorting the partial order given by the label y has to be respected (so that $\boldsymbol{x}_i$ comes before $\boldsymbol{x}_j$ when either $y_i > y_j$, or $y_i == y_j$ and $\boldsymbol{w} \cdot \boldsymbol{x}_i > \boldsymbol{w} \cdot \boldsymbol{x}_j$).

To evaluate our algorithm, we ran experiments on the OHSUMED tasks of the LETOR 3.0 dataset (Liu et al., 2007). We use the per-query-normalized version of the features in all our training and testing below, and employ exactly the same training, test, and validation sets split as given.

For this application it is vital to have a good initialization of the latent varibles h. Simple initialization strategies such as randomly picking k relevant documents indicated by the label y does not work for these datasets with noisy relevance judgements, which usually give the trivial zero vector as solution. Instead we adopt the following initialization strategy. Using the same training and validation sets in each fold, we trained a model optimizing for weighted average classification accuracy (weighted by the reciprocal of the number of documents associated by each query). Then for each fold the trained model is used as the initial weight vector to optimize for precision@k.

Table 3. Precision@k on OHSUMED dataset (5-fold CV)

OHSUMED	P@1	P@3	P@5	P@10
Ranking SVM	0.597	0.543	0.532	0.486
ListNet	0.652	0.602	0.550	0.498
Latent Structural SVM	0.680	0.573	0.567	0.494
Initial Weight Vector	0.626	0.557	0.524	0.464

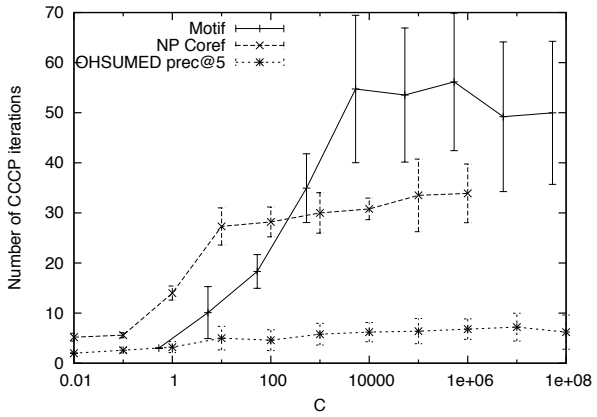

Figure 3. Number of CCCP iterations against C

We can see from Table 3 that our Latent Structural SVM approach performs better than the Ranking SVM (Herbrich et al., 2000; Joachims, 2002) on precision@1,3,5,10, one of the stronger baselines in the LETOR 3.0 benchmark. We also essentially tie with ListNet (Cao et al., 2007), one of the best overall ranking method in the LETOR 3.0 benchmark. As a sanity check, we also report the performance of the initial weight vectors used for initializing the CCCP. The Latent Structural SVM consistently improves upon these, showing that the good performance is not simply a result of good initialization.

5.4. Efficiency of the Optimization Algorithm

Figure 3 shows the number of iterations required for convergence for the three tasks for different values of the parameter C, averaged across all folds in their respective cross validation procedures. We fix the precision ϵ at 0.001 for the motif finding and optimizing for precision@k tasks, and use $\epsilon = 0.05$ for the noun phrase coreference task due to a different scaling of the loss function. We can see that in general the number of CCCP iterations required only grows very mildly with C, and most runs finish within 50 iterations. As the cost of each CCCP iteration is no more than solving a standard Structural SVM optimization problem (with

the completion of latent variables), the total number of CCCP iterations gives us a rough estimate of the cost of training Latent Structural SVMs, which is not particularly expensive. In practice the cost is even lower because we do not need to solve the optimization problem to high precision in the early iterations, and we can also reuse solution from the previous iteration for warm start in a new CCCP iteration.

6. Conclusions

We have presented a framework and formulation for learning Structural SVMs with latent variables. We identify a particular case that covers a wide range of application problems, yet affords an efficient training algorithms using Convex-Concave Programming. The algorithm is modular and easily adapted to new applications. We demonstrated the generality of the Latent Structural SVM with three applications, and a future research direction will be to explore further applications of this algorithm in different domains.

Acknowledgments

This work is supported by NSF Award IIS-0713483. We would like to thank Tom Finley and Professor Uri Keich for the datasets, and the anonymous reviewers for their helpful suggestions to improve this paper.

References

Bailey, T., & Elkan, C. (1995). Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization. *Machine Learning*, *21*, 51–80.

Cao, Z., Qin, T., Liu, T., Tsai, M., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. *Proc. of the Int. Conf. on Mach. Learn.* (pp. 129–136).

Chapelle, O., Do, C., Le, Q., Smola, A., & Teo, C. (2008). Tighter bounds for structured estimation. *Adv. in Neural Inf. Process. Syst.* (pp. 281–288).

Collobert, R., Sinz, F., Weston, J., & Bottou, L. (2006). Trading convexity for scalability. *Proc. of the Int. Conf. on Mach. Learn.* (pp. 201–208).

Felzenszwalb, P., McAllester, D., & Ramanan, D. (2008). A Discriminatively Trained, Multiscale, Deformable Part Model. *Proc. Computer Vision and Pattern Recognition Conf.* (pp. 1–8).

Finley, T., & Joachims, T. (2005). Supervised clustering with support vector machines. *Proc. of the Int. Conf. on Mach. Learn.* (p. 217).

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. In *Advances in large margin classifiers*, chapter 7, 115–132. MIT Press.

Joachims, T. (2002). Optimizing search engines using clickthrough data. *ACM SIGKDD Conf. on Knowledge Discovery and Data Mining* (pp. 133–142).

Joachims, T., Finley, T., & Yu, C. (To appear). Cutting-plane training of structural SVMs. *Machine Learning*.

Kiwiel, K. (1990). Proximity control in bundle methods for convex nondifferentiable minimization. *Mathematical Programming*, *46*, 105–122.

Liu, T., Xu, J., Qin, T., Xiong, W., & Li, H. (2007). LETOR: Benchmark dataset for research on learning to rank for information retrieval. *SIGIR Workshop on Learning to Rank for Information Retrieval*.

Ng, P., & Keich, U. (2008). GIMSAN: a Gibbs motif finder with significance analysis. *Bioinformatics*, *24*, 2256.

Ng, V., & Cardie, C. (2002). Improving machine learning approaches to coreference resolution. *Proc. of Assoc. for Computational Linguistics* (p. 104).

Petrov, S., & Klein, D. (2007). Discriminative Log-Linear Grammars with Latent Variables. *Adv. in Neural Inf. Process. Syst.* (p. 1153).

Smola, A., Vishwanathan, S., & Hofmann, T. (2005). Kernel methods for missing variables. *Proc. of the Int. Conf. on Artif. Intell. and Stat.* (p. 325).

Taskar, B., Guestrin, C., & Koller, D. (2003). Max-margin Markov networks. *Adv. in Neural Inf. Process. Syst.* (p. 51).

Tsochantaridis, I., Hofmann, T., Joachims, T., & Altun, Y. (2004). Support vector machine learning for interdependent and structured output spaces. *Proc. of the Int. Conf. on Mach. Learn.* (p. 104).

Vilain, M., Burger, J., Aberdeen, J., Connolly, D., & Hirschman, L. (1995). A model-theoretic coreference scoring scheme. *Proceedings of the 6th conference on Message understanding* (pp. 45–52).

Wang, S., Quattoni, A., Morency, L., Demirdjian, D., & Darrell, T. (2006). Hidden Conditional Random Fields for Gesture Recognition. *Proc. Computer Vision and Pattern Recognition Conf.* (p. 1521).

Wang, Y., & Mori, G. (2008). *Max-margin hidden conditional random fields for human action recognition* (Technical Report TR 2008-21). School of Computing Science, Simon Fraser University.

Yuille, A., & Rangarajan, A. (2003). The Concave-Convex Procedure. *Neural Computation*, *15*, 915.

Zien, A., Brefeld, U., & Scheffer, T. (2007). Transductive support vector machines for structured variables. *Proc. of the Int. Conf. on Mach. Learn.* (pp. 1183–1190).