Transductive Learning of Structural SVMs via Prior Knowledge Constraints

Chun-Nam Yu
Dept. of Computing Science & AICML
University of Alberta, AB, Canada
`chunnam@cs.ualberta.ca`

Abstract

Reducing the number of labeled examples required to learn accurate prediction models is an important problem in structured output prediction. In this paper we propose a new transductive structural SVM algorithm that learns by incorporating prior knowledge constraints on unlabeled data. Our formulation supports different types of prior knowledge constraints, and can be trained efficiently. Experiments on two citation and advertisement segmentation tasks show that our transductive structural SVM can learn effectively from unlabeled data, achieving similar prediction accuracies when compared against other state-of-art algorithms.

1 Introduction

Structural support vector machine (structural SVM) [18] is a widely used algorithm for learning structured output prediction models due to its simplicity and high prediction accuracy for many applications. However, thus far there is no published method to incorporate prior knowledge about the structured output learning problem into the training of structural SVMs. Prior knowledge is particularly useful when labeled data are scarce or expensive to collect, which is the case for structured output learning. Moreover, although the space of all possible outputs is huge (typically exponential in input length), the space of 'proper' outputs is usually much smaller because the output domain is in practice highly constrained. In sequence labeling problems including part-of-speech tagging, segmentation of citations and classified advertisements, many of the possible output sequences can be ruled out by simple domain knowledge. For example, most proper English sentences contain a noun and a verb, while segment boundaries in citations are usually marked by punctuation marks and the segments have to follow a certain order. Any output sequences that violate these rules are highly unlikely to be correct. These facts have been exploited by different researchers [3, 14, 1] to improve semi-supervised learning.

In this paper we propose a new method for incorporating these prior knowledge as (soft) constraints in the training of structural SVMs. The key idea of our approach is to decompose the penalty for violating these prior knowledge constriants as the sum of a quadratic penalty term and a simplified constraint violation penalty involving auxiliary variables. This results in a tractable optimization problem that we can solve efficiently with DC programming methods.

Our transductive structural SVM allows flexible types of prior knowledge constraints to be expressed. It can handle both constraints on individual examples (e.g., each sentence has to contain at least one verb) and sample-based expectation constraints (e.g., the average number of nouns over all sentences has to be greater than the average number of verbs). For most types of constriants, inference during training can be performed efficiently via dynamic programming or beam search. Interestingly, our method also suggests a new formulation for learning structural SVMs for supervised sequence/graph labeling, by encoding the labels as constraints on outputs.

Experiments on two citation and advertisement segmentation tasks show that our transductive structural SVM can learn effectively from enforcing constraints over unlabeled data, improving the prediction accuracies quite substantially when the number of labeled examples is small. It achieves similar prediction accuracies when compared against other state-of-art algorithms. Our experiments also show that, when the

Appearing in Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX. Copyright 2012 by the authors.

labels in a sequence labeling task are encoded as constraints, our transductive structural SVM is able to learn a model with prediction accuracies almost identical with a standard supervised structural SVM.

2 Related Works

Most of the related works on applying constraints in semi-supervised learning come from the natural language processing and information retrieval communities, where reducing the labeling effort required to train good prediction models is a top priority. Chang et al. [3] introduced *constraint-driven learning* (CODL), a self-training style algorithm for semi-supervised learning. They applied constraints to improve the labeling of a pool of unlabeled examples, before the unlabeled examples were added to re-train the model. CODL optimizes an approximate likelihood based on n-best list.

Bellare et al. [1] proposed an alternating projection algorithm for learning with expectation constraints over unlabeled data. The expectation constraints are maintained over an auxiliary distribution q instead of the predictive distribution p, and the KL-divergence between p and q are minimized as part of the objective. This arrangement provides improved computational efficiency over their original Generalized Expectation criterion [14]. Ganchev et al. [5] also proposed a method called *posterior regularization*, which utilizes a very similar training objective.

Liang et al. [12] proposed a Bayesian model that treats various information such as labeled data, labeled features, and constraints as *measurements* that can be applied in learning. Direct optimization of this Bayesian model is intractable and they proposed approximate inference methods for it.

Unlike CODL which focuses on constraints on individual examples, our transductive structural SVM can handle both example-based and expectation constraints under the same framework via the introduction of auxiliary variables. Our use of auxiliary variables is similar to the use of auxiliary distribution q in [1], but instead of minimizing the KL-divergence we minimize the quadratic distance between the auxiliary variables and the model predictions instead.

Previously Zien et al. [21] also considered the problem of transductive learning of structural SVMs. But their formulation was based on the large-margin assumption used in semi-supervised classification and cannot incorporate domain-dependent prior knowledge.

3 Structural SVM with Constraints

Suppose we are given a set of labeled training examples $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, structural SVM (margin-rescaling) [18] learns a weight vector $\vec{w}$ by solving the following convex optimization problem:

$$\min_{\vec{w}} \frac{1}{2}\|\vec{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \max_{\hat{y}\in\mathcal{Y}}[\Delta(y_i, \hat{y}) + \vec{w}\cdot\delta\Phi_i(\hat{y})], \quad (1)$$

where we use the shorthand $\delta\Phi_i(\hat{y})$ to denote $\Phi(x_i, \hat{y}) - \Phi(x_i, y_i)$. The joint feature map $\Phi(x, y)$ extracts relevant features measuring the compatibility of input x and output y, scored by the weight vector $\vec{w}$. The loss function $\Delta(y, \hat{y})$ measures the cost of predicting an alternative output $\hat{y}$ instead of the reference output y. Denoting the prediction output for a fixed weight vector $\vec{w}$ for a given input x as $y(x; \vec{w})$, where

$$y(x; \vec{w}) = \text{argmax}_{\hat{y}\in\mathcal{Y}}\, \vec{w}\cdot\Phi(x, \hat{y}),$$

it is well known that the prediction loss for training example x_i is upper-bounded by the second term in Eq (1) [18]:

$$\Delta(y_i, y(x_i; \vec{w})) \le \max_{\hat{y}\in\mathcal{Y}}[\Delta(y_i, \hat{y}) + \vec{w}\cdot\delta\Phi_i(\hat{y})].$$

Thus the structural SVM optimization problem minimizes a sum of convex upper bounds of the prediction loss $\Delta(y_i, y(x_i; \vec{w}))$ over the training set, subject to regularization over the norm of the weight vector $\vec{w}$. This algorithm gives state-of-art performance in many structured output prediction applications [18, 16].

3.1 Prior Knowledge as Constraints

Suppose in addition to the labeled training data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, we are given a set of unlabeled data of size m $\{x_{n+1}, \ldots, x_{n+m}\}$. In addition to minimizing an upper bound over the loss function Δ on the labeled data, we might also want to ensure that the predictions $y(x_j; \vec{w})$ over the unlabeled data are 'proper' and satisfy certain constraints according to our prior knowledge. For example, in part-of-speech tagging, we might want to ensure that each output $y(x_j; \vec{w})$ contains at least one verb. Let $\phi : \mathcal{X}\times\mathcal{Y}\to\mathbb{R}$ be a function that counts the number of verbs in an output sequence. This constraint can be expressed as:

$$\phi(x_j, y(x_j; \vec{w})) \ge 1 \qquad \text{for } n+1 \le j \le n+m, \quad (2)$$

and can be incorporated in the structural SVM optimization problem in Eq (1). As another example, in handwriting recognition, we might want to ensure the 3-gram letter frequencies of the outputs to match those from English documents from the same domain. Let $\phi : \mathcal{X}\times\mathcal{Y}\to\mathbb{R}^d$ be a function that counts the number of each type of 3-grams in the output (with $d = 26^3$), we can express the above constraint as:

$$\frac{1}{m}\sum_{j=n+1}^{n+m} \phi(x_j, y(x_j;\vec{w})) = \vec{a}, \tag{3}$$

where $\vec{a} \in \mathbb{R}^d$ is a 3-gram frequency vector estimated from a large corpus in the same domain.

Instead of the hard constraints in Eqs (2) and (3) above, in practice a more common approach is to penalize their corresponding soft version based on the hinge loss and L1-norm difference:

$$[1 - \phi(x_j, y(x_j;\vec{w}))]_+, \tag{4}$$

$$\left\|\frac{1}{m}\sum_{j=n+1}^{n+m} \phi(x_j, y(x_j;\vec{w})) - \vec{a}\right\|_1. \tag{5}$$

3.2 Penalizing Constraint Violations with Auxiliary Variables

The major difficulty in dealing with constraints in Eqs (4) and (5) comes from $\phi(x_j, y(x_j;\vec{w}))$, which is usually a non-convex and discontinuous function of $\vec{w}$. This makes it very hard to solve for $\vec{w}$ with these constraints incorporated. To make progress on this problem, let us assume we encode the constraints in Eqs (4) and (5) as a constraint penalty function $\Gamma : \mathbb{R}^{d\times m} \to \mathbb{R}$:

$$\Gamma(\phi_{n+1}(y(x_{n+1};\vec{w})), \ldots, \phi_{n+m}(y(x_{n+m};\vec{w}))),$$

where we introduce the shorthand $\phi_j(y)$ for $\phi(x_j, y)$.

We assume Γ is convex and Lipschitz-continuous with Lipschitz constant L. This holds for the type of (soft) linear equality and inequality constraints we have considered so far. We can then incorporate the constraint penalty function into Eq (1):

$$\min_{\vec{w}} \frac{1}{2}\|\vec{w}\|^2 + \frac{C_1}{n}\sum_{i=1}^{n} \max_{\hat{y}\in\mathcal{Y}}[\Delta(y_i, \hat{y}) + \vec{w}\cdot\delta\Phi_i(\hat{y})]$$
$$+C_2\Gamma(\phi_{n+1}(y(x_{n+1};\vec{w})), \ldots, \phi_{n+m}(y(x_{n+m};\vec{w}))), \tag{6}$$

where C_1, C_2 are regularization constants. The constraint penalty Γ acts like a (data-dependent) prior term over the weight vector $\vec{w}$.

The key idea to make the optimization problem tractable is to introduce auxiliary vectors $\vec{v}_j \in \mathbb{R}^d$ to replace $\phi_j(y(x_j;\vec{w}))$ in Γ, and enforce $\phi_j(y(x_j;\vec{w}))$ and $\vec{v}_j$ to be close to each other. Focusing on the term $\Gamma(\phi_{n+1}(y(x_{n+1};\vec{w})), \ldots, \phi_{n+m}(y(x_{n+m};\vec{w})))$ in Eq (6), we can replace it with the expression:

$$\frac{\mu}{2}\sum_{j=n+1}^{n+m} \|\phi_j(y(x_j;\vec{w})) - \vec{v}_j\|^2 + \Gamma(\vec{v}_{n+1}, \ldots, \vec{v}_{n+m}). \tag{7}$$

This idea is related to the use of quadratic penalty function for turning a constrained minimization problem into an unconstrained minimization problem [15]. The parameter μ controls the tradeoff between minimizing Γ and making sure that $\vec{v}_j$ and $\phi_j(y(x_j;\vec{w}))$ are close. As μ tends to infinity, $\vec{v}_j$ will get closer and closer to $\phi_j(y(x_j;\vec{w}))$, so that $\Gamma(\vec{v}_{n+1}, \ldots, \vec{v}_{n+m})$ will also get closer and closer to $\Gamma(\phi_{n+1}(y(x_{n+1};\vec{w})), \ldots, \phi_{n+m}(y(x_{n+m};\vec{w})))$ (by our Lipschitz continous assumption). This idea is also similar to the use of auxiliary distribution q in [1], but in our case quadratic distance is employed in place of KL-divergence.

The next step involves upper-bounding the quadratic term. Let us first consider the following upper bound, derived using similar techniques employed in previous works on large-margin structured prediction [4, 19]:

$$\begin{aligned}
&\frac{\mu}{2}\|\phi_j(y(x_j;\vec{w})) - \vec{v}_j\|^2 \\
&= \vec{w}\cdot\Phi(x_j, y(x_j;\vec{w})) + \frac{\mu}{2}\|\phi_j(y(x_j;\vec{w})) - \vec{v}_j\|^2 \\
&\quad - \vec{w}\cdot\Phi(x_j, y(x_j;\vec{w})) \\
&\leq \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j, \hat{y}) + \frac{\mu}{2}\|\phi_j(\hat{y}) - \vec{v}_j\|^2] - \vec{w}\cdot\Phi(x_j, y(x_j;\vec{w})) \\
&= \frac{\mu}{2}\|\vec{v}_j\|^2 + \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j, \hat{y}) - \mu\vec{v}_j\cdot\phi_j(\hat{y}) + \frac{\mu}{2}\|\phi_j(\hat{y})\|^2] \\
&\quad - \max_{\hat{y}\in\mathcal{Y}} \vec{w}\cdot\Phi(x_j, \hat{y})
\end{aligned} \tag{8}$$

This bound is the difference of two convex functions (maximum over a set of linear functions in $\vec{w}$ and $\vec{v}_j$), and can be minimized using difference of convex (DC) programming methods [8, 20].

However, as we are going to use the Convex-Concave Procedure (CCCP) [20] to solve the DC programs below, this particular bound in Eq (8) suffers from a serious drawback. In the upper bounding step of the CCCP algorithm we replace the concave term with the linearization $\vec{w}\cdot\Phi(x_j, y^*)$, where $y^* = y(x_j;\vec{w}^{(t)})$ is the prediction from the current solution $\vec{w}^{(t)}$. As y^* is estimated using our current solution $\vec{w}^{(t)}$ without any information from the constraints, the CCCP algorithm tends to get trapped in local minima that are close to the initial solution.

Consider instead the following bound that has stronger symmetry in its convex and concave part:

$$\max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j, \hat{y}) + \frac{\mu}{4}\|\phi_j(\hat{y}) - \vec{v}_j\|^2]$$
$$- \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j, \hat{y}) - \frac{\mu}{4}\|\phi_j(\hat{y}) - \vec{v}_j\|^2]. \tag{9}$$

It is fairly straightforward to show that the expression is an upper bound on the quadratic distance, in the following lemma:

Lemma 1.

$$\frac{\mu}{4}\|\phi_j(y(x_j;\vec{w}))-\vec{v}_j\|^2 \le \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y})+\frac{\mu}{4}\|\phi_j(\hat{y})-\vec{v}_j\|^2] - \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y})-\frac{\mu}{4}\|\phi_j(\hat{y})-\vec{v}_j\|^2].$$

Proof. Notice by definition of max,

$$\vec{w}\cdot\Phi(x_j,y(x_j;\vec{w}))+\frac{\mu}{4}\|\phi_j(y(x_j;\vec{w}))\|^2 \le \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y})+\frac{\mu}{4}\|\phi_j(\hat{y})-\vec{v}_j\|^2] \quad (10)$$

Also, as μ is positive and the quadratic distance is non-negative,

$$\vec{w}\cdot\Phi(x_j,y(x_j;\vec{w})) = \max_{\hat{y}\in\mathcal{Y}}\vec{w}\cdot\Phi(x_j,\hat{y}) \ge \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y})-\frac{\mu}{4}\|\phi_j(\hat{y})-\vec{v}_j\|^2] \quad (11)$$

Subtracting inequality (11) from inequality (10) gives the result. □

Another way to motivate the bound in Eq (9) is to consider its upper-bounding step in CCCP, which involves the following inference problem:

$$\bar{y}(x_j;\vec{w},\vec{v}_j)=\operatorname*{argmax}_{\hat{y}\in\mathcal{Y}}[\vec{w}\Phi(x_j,\hat{y})-\frac{\mu}{4}\|\vec{v}_j-\phi_j(\hat{y})\|^2]. \quad (12)$$

The output $\bar{y}(x_j;\vec{w},\vec{v}_j)$ tries to score high with $\vec{w}$, while maintaining a small quadratic distance to our current $\vec{v}_j$. As $\vec{v}_j$ is optimized to minimize the constraint violation penalty Γ, a small quadratic distance to $\vec{v}_j$ implies a small constraint violation penalty. Thus $\bar{y}(x_j;\vec{w},\vec{v}_j)$ balances between achieving a high output score and maintaining the constraints, with tradeoff tuned by μ. We call this *constraint-guided inference.* This ensures that information from the constraints is used in the upper-bounding step of the CCCP algorithm to avoid bad local minima.

Following exactly the same steps as in Eq (8), we can also show that

$$\frac{\mu}{2}\|\phi_j(\bar{y}(x_j;\vec{w},\vec{v}_j))-\vec{v}_j\|^2 \le \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y})+\frac{\mu}{4}\|\phi_j(\hat{y})-\vec{v}_j\|^2] - \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y})-\frac{\mu}{4}\|\phi_j(\hat{y})-\vec{v}_j\|^2].$$

Compared to Lemma 1, the upper bound on the output of constraint-guided inference $\bar{y}(x_j;\vec{w},\vec{v}_j)$ is tighter than that on the prediction $y(x_j;\vec{w})$, by a factor of 2.

Finally, plugging the bound in Eq (9) back into Eq (6), we can solve the following optimization problem for transduction:

$$\min_{\vec{w},\vec{v}_{n+1},\ldots,\vec{v}_{n+m}} \frac{1}{2}\|\vec{w}\|^2+\frac{C_1}{n}\sum_{i=1}^{n}\max_{\hat{y}\in\mathcal{Y}}[\Delta(y_i,\hat{y})+\vec{w}\cdot\delta\Phi_i(\hat{y})] + C_2\Gamma(\vec{v}_{n+1},\ldots,\vec{v}_{n+m}) + \frac{C_2\mu}{2m}\sum_{j=n+1}^{n+m}\|\vec{v}_j\|^2 + \frac{C_2}{m}\sum_{j=n+1}^{n+m}\Big\{\max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y})-\frac{\mu}{2}\vec{v}_j\cdot\phi_j(\hat{y})+\frac{\mu}{4}\|\phi_j(\hat{y})\|^2] - \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y})+\frac{\mu}{2}\vec{v}_j\cdot\phi_j(\hat{y})-\frac{\mu}{4}\|\phi_j(\hat{y})\|^2]\Big\}. \quad (13)$$

We have now obtained a computationally tractable formulation of transductive structural SVM with side constraints.

3.3 Comparison with Related Methods

Previous work on transductive structural SVMs [21] assumes the correct outputs can be separated from the incorrect outputs with large margin specified by the loss Δ, generalizing the large-margin assumption in binary classification. In our transductive structural SVM formulation in Eq (13) above, we assume instead prior knowledge constraints are satisfied by the unlabeled data, so that good model parameters $\vec{w}$ should have low constraint violation penalty value Γ. The underlying transduction principles are quite different for these two different approaches.

Our problem formulation, and the optimization procedure using CCCP below, are also closely related to latent structural SVM [19]. Latent structural SVM could be applied to enforce some of the constraints in the experiments in Section 5. For example, in the citations segmentation task, there are constraints stating that the first word in each citation belongs to the field *title*, and the word 'proceedings' belongs to the field *journal.* We can regard these constraints as observed information y for an unlabeled example, and treat the rest of the sequence as latent variable h. This allows us to use the latent structural SVM algorithm for learning, by iteratively imputing the latent variables h and re-optimizing the weight vector. However, this essentially treats all the prior knowledge as *hard constraints*, and could be problematic when the constraints can be violated or when multiple constraints conflict with each other. For example, the constraint stating that segment boundary has to occur at punctuation marks is violated quite often for the citations data, and accuracy will degrade if we enforce it as a hard constraint. Our proposed transductive structural SVM is instead based on *soft constraints* and allows the constraints to be violated in the imputation (constraint-guided inference Eq (12)), with the auxiliary variables $\vec{v}_j$ controlling the costs of violation. This gives a more robust approach of applying prior knowledge constraints.

3.4 Solving the Optimization Problem

As in the training of structural SVMs , we can solve Eq (13) with the cutting plane algorithm [11] in conjunction with CCCP. Since the overall algorithm is very similar to previous approaches in [19], the details are relegated to the Appendix. Here we just want to briefly mention the inferences required for computing the cutting planes for the terms in Eq (13). First we need the usual loss-augmented inference for the loss over labeled data:

$$\hat{y}_i = \text{argmax}_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_i,\hat{y}) + \Delta(y_i,\hat{y})].$$

Second we need to compute cutting planes over the constraint penalty Γ for each $\vec{v}_j$,

$$\frac{\partial}{\partial \vec{v}_j}\Gamma(\vec{v}_{n+1},\ldots,\vec{v}_{n+m}).$$

The difficulty of this computation depends on the structure of Γ, but is usually quite simple for linear equality or inequality constraints (Eqs (4) and (5)). Finally, we need to compute cutting planes for the convex and concave part of Eq (9):

$$\hat{y}_j = \underset{\hat{y}\in\mathcal{Y}}{\text{argmax}}[\vec{w}\cdot\Phi(x_j,\hat{y}) - \frac{\mu}{2}\vec{v}_j\cdot\phi_j(\hat{y}) + \frac{\mu}{4}\|\phi_j(\hat{y})\|^2] \quad (14)$$

$$\bar{y}_j = \underset{\bar{y}\in\mathcal{Y}}{\text{argmax}}[\vec{w}\cdot\Phi(x_j,\bar{y}) + \frac{\mu}{2}\vec{v}_j\cdot\phi_j(\bar{y}) - \frac{\mu}{4}\|\phi_j(\bar{y})\|^2]. \quad (15)$$

The computation of these cuts for the quadratic distance term requires a more detailed discussion, which we defer to Section 3.5.

To conclude this section, we can prove a lemma describing the quality of the solution returned by the cutting plane algorithm:

Lemma 2. *Suppose at the termination of the cutting plane algorithm the solution* $\vec{w}^*$, $\vec{v}^*_{n+1},\ldots,\vec{v}^*_{n+m}$ *is returned. Let*

$$A = \frac{1}{m}\sum_{j=n+1}^{n+m}\Big\{\max_{\hat{y}\in\mathcal{Y}}[\vec{w}^*\cdot\Phi(x_j,\hat{y}) + \frac{\mu}{4}\|\phi_j(\hat{y}) - \vec{v}^*_j\|^2] - \max_{\hat{y}\in\mathcal{Y}}[\vec{w}^*\cdot\Phi(x_j,\hat{y}) - \frac{\mu}{4}\|\phi_j(\hat{y}) - \vec{v}^*_j\|^2]\Big\}$$

$$B = \Gamma(\vec{v}^*_{n+1},\ldots,\vec{v}^*_{n+m}).$$

Then the constraint violation penalty Γ *on the unlabeled data is bounded by:*

$$\Gamma(y_{n+1},\ldots,y_{n+m}) \le 2L\sqrt{\frac{mA}{\mu}} + B,$$

where $y_j = y(x_j;\vec{w}^*)$ *are the argmax predictions.*

Proof. Notice by Lemma 1, A is an upper bound of

$$\frac{1}{m}\sum_{j=n+1}^{n+m}\frac{\mu}{4}\|\phi(y(x_j;\vec{w}^*)) - \vec{v}^*_j\|^2.$$

Hence the squared 2-norm difference between all $\vec{v}^*_j$ and $\phi(y(x_j;\vec{w}^*))$ (when they are stacked as a single vector) is bounded by $4mA/\mu$. For a convex function f with Lipschitz constant L, we have $f(x) \le f(y) + L\|y-x\|_2$. The result follows from this inequality and our assumption that Γ is convex and L-Lipschitz. $\square$

The above lemma gives a bound on the quality of solution (based on constraint violation Γ) on the unlabeled data using the quantities A and B. The quantities A and B are directly minimized by our algorithm. In general when μ increases, the quadratic distance between $\vec{v}_j$ and $\phi_j(y(x_j;\vec{w}))$ (represented by A) decreases. But at the same time the loss Γ over $\vec{v}_{n+1},\ldots,\vec{v}_{n+m}$ (represented by B) would usually increase. Thus to obtain a good bound we need to obtain a careful trade-off between these two terms through the choice of μ.

3.5 Inference for the Quadratic Penalty Term

Consider Eq (14), where we need to compute

$$\hat{y} = \text{argmax}_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y}) - \frac{\mu}{2}\vec{v}_j\cdot\phi_j(\hat{y}) + \frac{\mu}{4}\|\phi_j(\hat{y})\|^2].$$

Assuming $\phi_j(\hat{y})$ decomposes into the same clique potentials as $\Phi(x_j,\hat{y})$, the major remaining difficulty in the above computation comes from the quadratic term $\mu/4\|\phi_j(\hat{y})\|^2$. For some problems $\|\phi_j(\hat{y})\|^2$ is constant for all output labels $\hat{y}\in\mathcal{Y}$, so that this quadratic term can be ignored. For example, this is true for the class-balance constraints in the training of transductive SVM [10], and also true for the alternative formulation of structural SVM for sequence/graph labeling in Section 4. Another important case where inference is simple is when the entries of $\phi_j(\hat{y})$ are binary-valued (0 or 1), e.g., the entries indicate whether the output $\hat{y}$ violates particular constraints or not. Denote the kth dimension of $\phi_j(\hat{y})$ as $(\phi_j(\hat{y}))_k$, we have

$$\|\phi_j(\hat{y})\|^2 = \sum_{k=1}^{d}(\phi_j(\hat{y}))_k^2 = \sum_{k=1}^{d}(\phi_j(\hat{y}))_k = \vec{1}\cdot\phi_j(\hat{y}).$$

Thus the squared norm $\|\phi_j(\hat{y})\|^2$ is effectively linear under the binary-value assumption. These two special cases already cover a lot of interesting applications where prior knowledge constraints need to be incorporated. For the computation of the linear upper bound for the concave term in Eq (15), the same technique applies by symmetry.

For other problems we can make use of double dynamic programming to compute the argmax exactly, similar to the computation of feature covariance in entropy regularization [9]. Other possibilities include linearizing the quadratic term or performing local search to find an approximate solution. However we will not pursue these options in the current paper.

4 Alternative Formulation of Structural SVM for Labeling

Our new method for incorporating soft constraints on the outputs provides an interesting alternative for training structural SVMs with labeled data. Consider the problem of graph labeling with the loss function Δ being the Hamming loss. Let x be an input graph, x^i be the ith node in x, and $l(x)$ be the size of the graph. Let y be its corresponding labeling, and y^i the label at the ith node. Let c be the size of the output state space for each node. We can define the constraint function $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^{l(x)\times c}$ as follows:

$$\phi(x,y)_k = \begin{cases} 1 & \text{if } k = ic + j,\ y^i = j \\ 0 & \text{if } k = ic + j,\ y^i \neq j \end{cases},$$

with $0 \le i < l(x)$, $0 \le j < c$. Note that for this particular ϕ the output dimension is dependent on the size of the input graph $l(x)$, and can be different for different examples. In the case of sequence labeling we can think of $\phi(x,y)$ as indicator function over the trellis graph. Define the constraint function Γ as follows:

$$\Gamma(\vec{v}_1,\ldots,\vec{v}_n) = \frac{1}{2}\sum_{i=1}^{n} \|\vec{v}_i - \phi(x_i,y_i)\|_1.$$

Note that this definition of Γ coincides with the Hamming loss when $\vec{v}_i$ is a valid output encoding of some $\hat{y}$, i.e., $\vec{v}_i = \phi(x_i,\hat{y})$.

With the definition of ϕ and Γ as above, we can adopt Eq (13) but drop the standard loss term for labeled data completely to obtain the following:

$$\begin{aligned} &\min_{\vec{w},\vec{v}_1,\ldots,\vec{v}_n} \frac{1}{2}\|\vec{w}\|^2 + \frac{C_2\mu}{2n}\sum_{i=1}^{n}\|\vec{v}_i\|^2 + C_2\Gamma(\vec{v}_1,\ldots,\vec{v}_n) \\ &+ \frac{C_2}{n}\sum_{i=1}^{n}\Big\{\max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_i,\hat{y}) - \frac{\mu}{2}\vec{v}_i\cdot\phi_i(\hat{y}) + \frac{\mu}{4}\|\phi_i(\hat{y})\|^2] \\ &- \max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_i,\hat{y}) + \frac{\mu}{2}\vec{v}_i\cdot\phi_i(\hat{y}) - \frac{\mu}{4}\|\phi_i(\hat{y})\|^2]\Big\}. \quad (16) \end{aligned}$$

The information from labels y_i is now completely incorporated via Γ. This corresponds to an extreme case where the side constraints are so strong that they completely determine the labels over the unlabeled data.

One particular consequence of the definition of $\phi(x,y)$ is that given a fixed input x, the squared norm $\|\phi(x,\hat{y})\|^2$ is constant for all $\hat{y} \in \mathcal{Y}$. Therefore the inference for the quadratic penalty term in Eq (14) can be computed *exactly*. Indeed the inference for this particular definition of ϕ is almost the same as the loss-augmented inference in the training of standard structural SVM, with $\Delta(y_i,\hat{y})$ replaced by $\vec{v}_i \cdot \phi_i(\hat{y})$.

5 Experiments

In the following we evaluate our new transductive structural SVM algorithm on two different tasks. In the first evaluation we apply the algorithm to the problem of citation and advertisement segmentation to see whether it can learn effectively from prior knowledge constraints, and also compare its performance against other state-of-art algorithms. In the second evaluation we consider the task of handwriting recognition to compare the performance of standard structural SVM against our version outlined in Section 4.

5.1 Citation and Advertisement Segmentation

We obtain the citations and Craigslist aparments advertisement datasets from the authors of [3] and followed their tokenization procedure. The citations dataset consists of 500 labeled references of computer science research papers, and we split them into 300/100/100 for training/development/test sets as in [3]. The task is to segment the references into fields such as *author*, *editor*, *title*, and *journal*. The advertisement dataset consists of 300 manually labeled Craigslist apartment listings, and we split them into 100/100/100 for training/development/testing. The task is to segment the advertisement into fields such as *rent*, *size*, *restrictions*, and *features*. In addition to the labeled data, each of these two datasets are supplemented by 1000 unlabeled examples collected in [6].

The prior knowledge constraints for these two datasets are taken from Table 1 of [3] (the table is not reproduced here due to space limitation). There are three main types of constraints. The first type is transition constraints, which restricts segment boundaries to occur only at punctuation marks or newline. The second type is word constraints, which states that certain words can only belong to certain fields. For example, the words ‘kitchen’ and ‘parking’ have to belong to the field *features* in the advertisement dataset. The third type is non-local constraint which requires beam search for inference. For example, in the citations data each field is required to be a consecutive list of words, and cannot occur more than once in a citation.

For an input sequence of length L, we can encode these constraints into a vector $\phi(x,\hat{y})$ of dimension $2L$.

There are $L-1$ dimensions for $L-1$ transitions, and we set the ith dimension to 1 (and 0 otherwise) if there is no transition of state or the ith token is a punctuation. Similarly we have L dimensions for the word rules, and we set the ith dimension to 1 if the ith label belongs to the set of allowable labels of the ith word, or the ith word is not a restricted word. Finally we have one extra binary feature for the global constraint. In this encoding $\phi(x,\hat{y})$ is binary and inference for the quadratic penalty term can be performed efficiently. Here we pick the constrained penalty Γ to be:

$$\Gamma(\vec{v}_{n+1},\ldots,\vec{v}_{n+m}) = \sum_{j=n+1}^{n+m} \|\vec{v}_j - \vec{1}\|_1.$$

We use the vector $\vec{1}$ as the regression target as the vector of all '1's means all constraints are satisfied.

In the experiments below we report per-token accuracy of our transductive structural SVM (which we denote as Γ-SSVM) with different labeled training set sizes. We report the average accuracy over 5 bootstrap samples of labeled data, using the same subsets as in [3]. We aggregate the test set, development set, and the 1000 unlabeled examples as our unlabeled data $\{x_{n+1},\ldots,x_{n+m}\}$ in our transductive structural SVM formulation. The parameters C_1, C_2, μ are picked with the developement set. In the transductive experiments the weight vector $\vec{w}$ is initialized with the supervised solution on the labeled data, while the auxiliary vectors $\vec{v}_j$ are initialized at $\vec{1}$ (the regression target).

Table 1 shows the results for our transductive structural SVM against constraint-driven learning (CODL) [3] and alternating projection (AP) [1] on labeled set sizes of 5, 20 and 300. Our basic feature map $\Phi(x,y)$ consists of a first-order token based HMM with emission and transition features. In Chang et al. [3] the authors distinguished between two cases for CODL, depending on whether the constraints are used in prediction (I) or not (no I). In our transductive structural SVM we also distinguish between two cases: when only the basic HMM features are used in $\Phi(x,y)$ (no I), and when extra features from the constraint feature map $\phi(x,y)$ are added to $\Phi(x,y)$ (I). These extra features include special transition features that recognize punctuations, and a feature to score whether the global constraint is satisfied.

From Table 1, we can first see that incorporating constraints over unlabeled data using Γ-SSVM improves test set accuracy over the supervised baseline of standard structural SVM (sup), except for the full training set. This is true for the basic feature map $\Phi(x,y)$ (no I) and also true when extra features are added (I). Not surprisingly using extra features has better accuracies than just using basic features, but it is also interesting to note that the benefits from applying constraints over unlabeled data is also larger with extra features (from 2-3% for no I to about 4% for I). With 300 labeled examples enforcing constraints over the unlabeled data does not improve accuracies, as most of the constraints are already satisfied.

Table 1: Results (per-token accuracy) on the *Citations* dataset for different labeled set size N. Bold denotes the best performance in each category.

method \ N	5	20	300
CODL (I)	**77.8**	**86.1**	93.5
AP	75.6	85.4	94.8
Γ-SSVM (I)	76.8	**86.0**	**95.1**
sup (I)	72.6	82.3	**95.1**
CODL (no I)	71.0	79.4	88.2
Γ-SSVM (no I)	**72.8**	**81.4**	**92.8**
sup (no I)	69.4	79.3	**92.7**

Compared with the other methods, Γ-SSVM performs uniformly better than CODL when only basic features (no I) are used. When extra constraint features are employed Γ-SSVM performs on par or better than CODL except for sample size 5. Γ-SSVM also has (slightly) better accuracies on all three labeled set sizes than AP when extra features are used.

We next consider the advertisement segmentation task. Here we consider labeled data sizes of 10, 25, and 100 to match those used in previous published works. In addition to CODL we compare against conditional random fields with generalized expectation (CRF+GE) [13] and the Bayesian measurement framework (BM) [12]. Direct comparisons on this dataset is more difficult as these works all used slightly different tokenization and feature sets. For exmaple, CRF+GE used 'prototype' features based on SVD [7] while BM employed word clusterings features [2]. To make the results more comparable we added the set of 33 extra labeled features (word rules) in Table 1 of [13] (used by both CRF+GE and BM).

Table 2 shows the per-token accuracy of different algorithms on this task. Again we can see marked improvement of Γ-SSVM from the supervised baseline except the full labeled data set size (100). The improvement is particularly large for the small sample size of 10. When only basic features are used (no I), Γ-SSVM has very similar accuracy when compared to CODL.

When extra features are used (I), the probabilistic approaches (CRF+GE and BM) seem to be performing slightly better than Γ-SSVM. This could be due to the SVD or word clustering features used by these models, or to the better ability to maintain uncertainty over violated constraints through conditional distribu-

Table 2: Results (per-token accuracy) on the *Advertisement* dataset for different labeled set size N. Bold denotes the best performance in each category.

method \ N	10	25	100
CODL (I)	**74.7**	**78.5**	81.7
CRF + GE	72.6	76.7	80.1
BM	71.4	76.5	**82.5**
Γ-SSVM	72.0	75.9	80.5
sup (I)	65.2	74.1	80.6
CODL (no I)	70.9	74.8	**78.6**
Γ-SSVM (no I)	**71.2**	**75.3**	**78.6**
sup (no I)	63.1	72.3	78.1

tions. Indeed compared with the citations dataset, the constraints used in the advertisement data are more frequently violated by the true labels. Nevertheless, our Γ-SSVM can still learn from constraints over unlabeled data and achieve comparable accuracies. On the other hand, CODL with constraints outperforms all the other models on small sample sizes. This could be due to the application of CODL's 'soft' constraints, which considers the minimum number of swapping moves (hamming distance) required to satisfy a constraint.

As a summary, Γ-SSVM is an effective method for transductive learning with unlabeled data, and it achieves prediction accuracies on par with other state-of-art algorithms on these two segmentation tasks. As with other large-margin learning methods, one major strength of Γ-SSVM is that during learning only point estimate inference is required (loss-augmented, constraint-guided, etc). This avoids the use of n-best list approximation or MCMC required for the probabilistic models (e.g. Gibbs sampling in [1] for non-local constraints), and is especially valuable when the corresponding inference over constraints is difficult.

5.2 Handwriting Recognition

We also perform experiments on the alternative formulation of structural SVM for sequence labeling outlined in Section 4. We use the OCR data from [17] and use a first-order token-based HMM as features. From Table 3 we can see that Γ-SSVM has almost identical performance when compared against standard structural SVM (Δ-SSVM) over 10-fold cross validation. This is despite the fact we initialize the weights with $\vec{w}^{(0)} = \vec{0}$ and the auxiliary vectors $\vec{v}_j = \phi_j(y(x_j; \vec{w}^{(0)}))$. This is interesting because there are parameters C_2 and μ ($C_2 = 6260$ and $\mu = 1$ in our case) that can get close to the supervised solution, even when the transductive training objective is non-convex. Our transductive structural SVM algorithm is capable of learning a model comparable to a supervised model when the

Table 3: Results on OCR dataset. Γ-SSVM has similar accuracy compared to standard structural SVM (Δ-SSVM). The number in bracket is the standard error of 10CV.

	Per-token Accuracy
Δ-SSVM	86.2 (0.8)
Γ-SSVM	85.5 (0.6)

constraints are 'very strong' (completely define the labels in this case).

6 Conclusions and Future Work

As future directions we are interested in further investigating the different types of DC bounds over the quadratic distance. The tightness of these bounds have a direct impact on how well we are able to minimize the constraint violation penalty Γ. We are also interested in investigating the use of annealing for the parameter μ, to see if we could obtain local minima with better solution objectives with such a strategy. On the other hand, our experiments on the OCR task suggests the roles of the supervised loss Δ on labeled data and the constraint penalty Γ are interchangeable, and there are potential new ways to integrate their roles in learning.

In conclusion, we have proposed a new formulation of transductive structural SVM that learns by enforcing prior knowledge constraints over unlabeled data. We decomposed the constraint violation penalty into the sum of a quadratic distance term and a simplified penalty term involving auxiliary variables, and described an efficient algorithm for solving the associated optimization problem. It achieves similar prediction accuracies on two segmentation tasks when compared against other state-of-art semi-supervised algorithms that utilizes constraints.

Acknowledgments

I would like to thank Ming-Wei Chang for providing the data used in the experiments, and Thorsten Joachims and Xinhua Zhang for discussions on ideas leading to this work.

References

[1] K. Bellare, G. Druck, and A. McCallum. Alternating projections for learning with expectation constraints. In *UAI*, 2009.

[2] P.F. Brown, P.V. Desouza, R.L. Mercer, V.J.D. Pietra, and J.C. Lai. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479, 1992.

[3] M.W. Chang, L. Ratinov, and D. Roth. Guiding semi-supervision with constraint-driven learning. In *ACL*, 2007.

[4] C.B. Do, Q. Le, C.H. Teo, O. Chapelle, and A. Smola. Tighter bounds for structured estimation. In *NIPS*, 2008.

[5] K. Ganchev, J. Graça, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *JMLR*, 11:2001–2049, 2010.

[6] T. Grenager, D. Klein, and C.D. Manning. Unsupervised learning of field segmentation models for information extraction. In *ACL*, 2005.

[7] A. Haghighi and D. Klein. Prototype-driven learning for sequence models. In *HLT-NAACL*, 2006.

[8] R. Horst and N.V. Thoai. DC programming: overview. *Journal of Optimization Theory and Applications*, 103(1):1–43, 1999.

[9] F. Jiao, S. Wang, C.H. Lee, R. Greiner, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *ACL*, 2006.

[10] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.

[11] T. Joachims, T. Finley, and C.N.J. Yu. Cutting-plane training of structural SVMs. *Machine Learning*, 77(1):27–59, 2009.

[12] P. Liang, M.I. Jordan, and D. Klein. Learning from measurements in exponential families. In *ICML*, 2009.

[13] G. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning of conditional random fields. In *ACL*, 2008.

[14] G.S. Mann and A. McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *JMLR*, 11:955–984, 2010.

[15] J. Nocedal and S.J. Wright. *Numerical optimization.* Springer verlag, 1999.

[16] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *ICML*, 2005.

[17] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2003.

[18] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *JMLR*, 6(2):1453–1484, 2006.

[19] C.-N. Yu and T. Joachims. Learning structural SVMs with latent variables. In *ICML*, 2009.

[20] A.L. Yuille and A. Rangarajan. The concave-convex procedure. *Neural Computation*, 15(4):915–936, 2003.

[21] A. Zien, U. Brefeld, and T. Scheffer. Transductive support vector machines for structured variables. In *ICML*, 2007.

APPENDIX - SUPPLEMENTARY MATERIALS

Cutting Plane Algorithm for Solving Eq (13)

By introducing one slack variable for each loss term, we can rewrite Eq (13) as:

$$\min_{\vec{w},\vec{v}_{n+1},\ldots,\vec{v}_{n+m}} \frac{1}{2}\|\vec{w}\|^2 + \frac{C_2\mu}{2m}\sum_{j=n+1}^{n+m}\|\vec{v}_j\|^2 + C_1\xi_1 + C_2\xi_2 + C_2\xi_3$$
$$-\frac{C_2}{m}\sum_{j=n+1}^{n+m}\max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y}) + \frac{\mu}{2}\vec{v}_j\cdot\phi_j(\hat{y}) - \frac{\mu}{4}\|\phi_j(\hat{y})\|^2] \tag{17}$$

$$\xi_1 \geq \frac{1}{n}\sum_{i=1}^{n}\max_{\hat{y}\in\mathcal{Y}}[\Delta(y_i,\hat{y}) + \vec{w}\cdot\delta\Phi_i(\hat{y})] \tag{18}$$

$$\xi_2 \geq \frac{1}{m}\sum_{j=n+1}^{n+m}\max_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y}) - \frac{\mu}{2}\vec{v}_j\cdot\phi_j(\hat{y}) + \frac{\mu}{4}\|\phi_j(\hat{y})\|^2] \tag{19}$$

$$\xi_3 \geq \Gamma(\vec{v}_{n+1},\ldots,\vec{v}_{n+m}) \tag{20}$$

Here we can apply the CCCP algorithm [20] to tackle the concave term, by constructing a linear upper bound for it. Let $\vec{w}$ be the current solution (a fixed weight vector). Define

$$\vec{q} = \frac{1}{m}\sum_{j=n+1}^{n+m}\Phi(x_j,\bar{y}_j)), \qquad \vec{p}_j = \phi_j(\bar{y}_j), \tag{21}$$

with $\bar{y}_j = \operatorname{argmax}_{\hat{y}\in\mathcal{Y}}[\vec{w}\cdot\Phi(x_j,\hat{y}) + \frac{\mu}{2}\vec{v}_j\cdot\phi_j(\hat{y}) - \frac{\mu}{4}\|\phi_j(\hat{y})\|^2]$.

The upper bounds $\vec{q}$, $\vec{p}_j$ are obtained through constraint-guided inference. Replacing the concave term with these linear subgradients, the program:

$$\min_{\vec{w},\vec{v}_{n+1},\ldots,\vec{v}_{n+m}} \frac{1}{2}\|\vec{w}\|^2 + \frac{C_2\mu}{2m}\sum_{j=n+1}^{n+m}\|\vec{v}_j\|^2 + C_1\xi_1 + C_2\xi_2$$
$$+ C_2\xi_3 - C_2\vec{w}\cdot\vec{q} - \frac{C_2\mu}{2m}\sum_{j=n+1}^{n+m}\vec{v}_i\cdot\vec{p}_j + \frac{C_2\mu}{4m}\sum_{j=n+1}^{n+m}\|\vec{p}_j\|^2 \tag{22}$$

with the same linear constraints over ξ_1, ξ_2, ξ_3 as in Eq (17) is a global convex upper bound of Eq (17). Solving this convex program for a new $\vec{w}'$ (and corresponding $\vec{v}'_j$) give a lower objective than the current solution $\vec{w}$ in Eq (17). Alternating between upper bound construction with $\vec{q}, \vec{p}_j$ and solving for $\vec{w}$ with this CCCP algorithm will lead to convergence to a local minima or saddle point [20].

We are going to solve the convex program of Eq (22) in the dual. The dual of Eq (22) can be written as:

$$\begin{aligned}\max_{\vec{\alpha}\geq 0,\vec{\beta}\geq 0,\vec{\gamma}\geq 0} & \sum_{t=1}^{T}\alpha_t c^{(t)}+\sum_{t=1}^{T}\beta_t d^{(t)}+\sum_{t=1}^{T}\gamma_t e^{(t)} \\ & -\frac{1}{2}\|\vec{w}(\vec{\alpha},\vec{\beta})\|^2-\frac{C_2\mu}{2m}\|\vec{v}_j(\vec{\beta},\vec{\gamma})\|^2 \qquad (23)\\ s.t. \ & \sum_{t=1}^{T}\alpha_t=C_1, \quad \sum_{t=1}^{T}\beta_t=C_2, \quad \sum_{t=1}^{T}\gamma_t=C_2,\end{aligned}$$

where

$$\vec{w}(\vec{\alpha},\vec{\beta})=\sum_{t=1}^{T}\alpha_t\vec{g}^{(t)}+\sum_{t=1}^{T}\beta_t\vec{h}^{(t)}+C_2\vec{q} \qquad (24)$$

$$\vec{v}_j(\vec{\beta},\vec{\gamma})=\frac{m}{C_2\mu}\left(\frac{1}{m}\sum_{t=1}^{T}\beta_t\vec{h}_j^{(t)}+\sum_{t=1}^{T}\gamma_t\vec{f}_j^{(t)}+\frac{C_2\mu}{2m}\vec{p}_j\right). \qquad (25)$$

Here we use t to index the cuts added during iteration t of the cutting plane algorithm, up to the Tth iteration.

To complete the description of the algorithm, the cutting planes can be computed as follows. For ξ_1 we have

$$c^{(t)}=\frac{1}{n}\sum_{i=1}^{n}\Delta(y_i,\hat{y}_i), \quad \vec{g}^{(t)}=-\frac{1}{n}\sum_{i=1}^{n}\delta\Phi_i(\hat{y}_i), \qquad (26)$$

where $\hat{y}_i=\operatorname{argmax}_{\hat{y}\in\mathcal{Y}}[\vec{w}^{(t)}\cdot\Phi(x_i,\hat{y})+\Delta(y_i,\hat{y})]$.

For ξ_2 we have

$$\begin{aligned} d^{(t)}&=\frac{\mu}{4}\sum_{j=n+1}^{n+m}\|\phi_j(\hat{y}_j)\|^2, \\ \vec{h}^{(t)}&=\frac{-1}{m}\sum_{j=n+1}^{n+m}\Phi(x_j,\hat{y}_j), \quad \vec{h}_j^{(t)}=\frac{\mu}{2}\phi_j(\hat{y}_j),\end{aligned} \qquad (27)$$

where $\hat{y}_j=\underset{\hat{y}\in\mathcal{Y}}{\operatorname{argmax}}[\vec{w}^{(t)}\Phi(x_j,\hat{y})-\frac{\mu}{2}\vec{v}_j^{(t)}\phi_j(\hat{y})+\frac{\mu}{4}\|\phi_j(\hat{y})\|^2]$.

For ξ_3 we have

$$\begin{aligned} \vec{f}_j^{(t)}&\in\frac{\partial}{\partial\vec{v}_j}\Gamma(\vec{v}_{n+1}^{(t)},\ldots,\vec{v}_{n+m}^{(t)}), \\ e^{(t)}&=\Gamma(\vec{v}_{n+1}^{(t)},\ldots,\vec{v}_{n+m}^{(t)})-\sum_{j=n+1}^{n+m}\vec{v}_i^{(t)}\cdot\vec{f}_j^{(t)}.\end{aligned} \qquad (28)$$

Algorithm 1 Cutting Plane Procedure for Transductive Structural SVM with Constraints

```
1: Input: labeled data $\{(x_1,y_1),\ldots,(x_n,y_n)\}$, unlabeled data $\{x_{n+1},\ldots,x_{n+m}\}$, $C_1, C_2, \mu, \epsilon_1, \epsilon_2, \epsilon_3$
2: Initialize $\vec{w}^{(0)}$, $\vec{v}_j^{(0)}$. Set $t \leftarrow 0$, $D \leftarrow \infty$
3: **repeat**
4:    $D_{last} \leftarrow D$
5:    Compute $\vec{q}$, $\vec{p}_j$ using Eq (21)
6:    **while** $\xi_1-\epsilon_1 \geq RHS(18)$ or $\xi_2-\epsilon_2 \geq RHS(19)$ or $\xi_3-\epsilon_3 \geq RHS(20)$ **do**
7:       Compute cutting plane information $c^{(t)}$, $d^{(t)}$, $e^{(t)}$, $\vec{g}^{(t)}$, $\vec{h}^{(t)}$, $\vec{h}_j^{(t)}$, $\vec{f}_j^{(t)}$ using Eqs (26) to (28).
8:       Solve QP in Eq (23). Set $D$ to dual objective.
9:       $t \leftarrow t+1$
10:      Update $\vec{w}^{(t)}$, $\vec{v}_j^{(t)}$ by Eqs (24) and (25)
11:   **end while**
12: **until** $D_{last}-D<\epsilon$
13: Return $\vec{w}^{(t)}$, $\vec{v}_{n+1}^{(t)},\ldots,\vec{v}_{n+m}^{(t)}$
```

The whole algorithm is summarized in Algorithm 1. The terms $RHS(18)$, $RHS(19)$, $RHS(20)$ in the guard of the inner while loop refer to the right hand sides of Eqs (18) to (20). The outer loop stops when the decrease in objective between successive iteration is lower than some threshold ϵ. Here we set ϵ to be a function of the other parameters, with $\epsilon=C_1\epsilon_1+C_2\mu\epsilon_2+C_2\epsilon_3$. The weight vector $\vec{w}$ and auxiliary vectors $\vec{v}_j$ can be initialized in an application dependent manner.

One final comment with the above algorithm is on the storage requirement for the auxiliary vectors $\vec{v}_j$, especially when ϕ is high-dimensional (e.g., prior knowledge constraints over a n-gram frequencies). We do not need to store all $\vec{v}_j$ explicitly as we are solving the program in the dual. The $\vec{v}_j$'s can be computed on the fly whenever we need to generate a new cutting plane. From Eq (25) we can see $\vec{v}_j$ is a linear combination of $\vec{h}_j^{(t)}$, $\vec{f}_j^{(t)}$ and $\vec{p}_j$. While $\vec{h}_j^{(t)}=\mu/2\phi_j(\hat{y})$ and $\vec{p}_j=\phi_j(\bar{y}_j)$ are sparse (proportion to length/size of x_j), $\vec{f}_j^{(t)}$ could be dense, especially for constraints involving sample averages (e.g., Eq (5)). However, for constraints involving sample averages like $1/m\sum_{j=n+1}^{n+m}\vec{v}_j$, the subgradients $\vec{f}_j^{(t)}$ computed in Eq (28) is actually the same for all examples x_j due to symmetry. Hence the cutting plane generated can be shared for all examples x_j for this type of sample-based constraints. With the cleaning up of inactive cuts, the additional storage requirement is modest when compared to the cutting plane algorithm for standard structural SVMs.