Structured Prediction with Relative Margin

Pannagadatta Shivaswamy and Tony Jebara
Department of Computer Science
Columbia University, New York, NY
pks2103,jebara@cs.columbia.edu

Abstract

In structured prediction problems, outputs are not confined to binary labels; they are often complex objects such as sequences, trees, or alignments. Support Vector Machine (SVM) methods have been successfully extended to such prediction problems. However, recent developments in large margin methods show that higher order information can be exploited for even better generalization. This article first points out a shortcoming of the SVM approach for the structured prediction; an efficient formulation is then presented to overcome the problem. The proposed algorithm exploits the fact that both the minimum and the maximum of quantities of interest are often efficiently computable even though quantities such as mean, median and variance may not be. The resulting formulation produces state-of-the-art performance on sequence learning problems. Dramatic improvements are also seen on multi-class problems.

1 Introduction

Traditional machine learning algorithms are designed to handle simple outputs such as a real valued target or a class label from k possible classes. With the application of machine learning tools in ambitious areas such as natural language processing, biology and computer vision, many techniques have been extended to provide complex objects as outputs. For example, in natural language processing, input examples are sentences while outputs are tags such as the role of the word or the part-of-speech the word represents in the sentence. It is difficult to apply traditional approaches to such problems, in particular, when the number of possible outputs can be exponentially large (as a function of the length of the output). For example, brute force enumeration of all the classification constraints implicated by a complex structured output space may be prohibitive in a traditional SVM setting. Recently, there has been extensive interest in the machine learning community for solving such complex-output problems [4].

One example structured prediction problem is learning from sequences. Traditionally, Hidden Markov Models (HMM) were used in learning problems where output predictions are sequences. Discriminative methods have recently been brought to bear on sequence problems. Conditional Random Fields (CRFs) [10] take a probabilistic approach to avoid constraint enumeration. Boosting and online frameworks [1] have also been proposed. Subsequently, Hidden Markov Support Vector Machines [2] and Maximum Margin Markov networks [14] produced improved accuracy by maximizing margin while mimicking HMM and Markov Network-style dependencies between inputs and outputs. Recently, a more general solution was proposed in the SVM framework for structured prediction problems [15, 16].

While the large margin methods (such as perceptrons and Support Vector Machines) have been very successful over the last decade, recently, there has been growing interest to improve generalization by exploiting second order information. For example, the Second Order Perceptron algorithm [5] was shown to admit better mistake bounds than the classic perceptron algorithm. Similarly, both a hyperplane and a covariance were learned (under Gaussian assumptions) in an online setting [9, 7] as well as in a batch setting via so-called Gaussian Margin Machines (GMM) [8]. Unfortunately Gaussian Margin Machines involve an expensive log-determinant optimization step. In addition, so-called Universum examples have been used for better generalization [17, 13]. This approach requires one to choose Universum examples, however, it is often hard to find appropriate Universum data suitable for a given problem domain. Using geometric intuitions and VC dimension arguments, Ellipsoidal Kernel Machines (EKM) were proposed by [11]. Since EKMs require an expensive log-determinant optimization, a quadratic optimization approach was subsequently proposed in the Relative Margin Machine (RMM)[12] framework which scaled to large problems. The motivation behind this line of research is the fact that large margin on its own is not a meaningful quantity; a better way to measure margin is in relation to the

spread of the data. While this motivation is articulated in different ways in the above methods, it is a recurring theme in most of them. This paper extends this idea to the structured prediction setting.

2 Structured Prediction with SVMs

In structured prediction problems, a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ from an input space to a discrete output space is estimated from training data. This mapping function is recovered by performing a maximization involving another augmented function $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ such that: $f(\mathbf{x}) = \arg\max_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ where $\mathbf{w}$ denotes the parameters of the function F. As in most kernel based methods, the functional form is assumed to be linear, i.e., $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \boldsymbol{\psi}(\mathbf{x}, \mathbf{y}), \mathbf{w} \rangle$, where $\boldsymbol{\psi}(\mathbf{x}, \mathbf{y})$ is a joint feature mapping for the pair $(\mathbf{x}, \mathbf{y})$. The specific form of this feature mapping depends on the application.

Let $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ be the training dataset. The structured prediction version of the SVM (referred to as StructSVM) is posed as following optimization problem :[1] [2]

$$\min_{\mathbf{w}, \xi} \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle + \frac{C}{n} \sum_{i=1}^{n} \xi_i \qquad (1)$$
$$\text{s.t } \langle \mathbf{w}, \boldsymbol{\delta\psi}_i(\mathbf{y}) \rangle \geq 1 - \xi_i,\ \xi_i \geq 0 \quad \forall i \in N, \mathbf{y} \neq \mathbf{y}_i$$

Here $\boldsymbol{\delta\psi}_i(\mathbf{y})$ is a shorthand notation for $\boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y}_i) - \boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y})$. The constraint in the above formulation ensures that the real-valued prediction F for $(\mathbf{x}_i, \mathbf{y}_i)$ is larger than the prediction for $(\mathbf{x}_i, \mathbf{y})$ with the slack accounting for any violations . Minimizing $\frac{1}{2}\mathbf{w}^\top \mathbf{w}$ leads to a large margin solution. The parameter $C > 0$ trades off between the margin and the slack variables. Recovering $\mathbf{w}$ with the above maximum-margin optimization problem and using $f(\mathbf{x})$ for prediction leads to state-of-the-art performance on many structured prediction benchmark tasks [16].

3 Motivation for the Relative Margin

In Section 2, StructSVM was shown to separate the joint feature map of $\mathbf{x}_i$ and its correct label $\mathbf{y}_i$ (i.e., $\boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y}_i)$) from the joint feature map of $\mathbf{x}_i$ with any other label $\mathbf{y} \in \mathcal{Y} \setminus \{\mathbf{y}_i\}$ (i.e., $\boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y})$) with a large margin. While this approach provides strong empirical performance, this section points out a shortcoming of maximum margin which may compromise accuracy in some settings.

Consider the simple two dimensional example in the top row of Figure 1; three different scaled versions of a dataset are shown. The triangle represents $\boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y}_i)$ and the squares represent $\boldsymbol{\psi}(\mathbf{x}_i, \mathbf{y})$ for $\mathbf{y} \in \mathcal{Y} \setminus \{\mathbf{y}_i\}$.

Consider the leftmost plot in the top row of Figure 1. One possible decision boundary separating the correct class map from the incorrect class maps is shown in green (or in light shade) which is the large relative margin estimate. The solution shown in red (or in dark shade) is the large absolute margin estimate; it achieves the largest margin possible while still separating the feature map of the example with the correct label from the example with the incorrect labels.

Next, consider the same set of points after scaling the data (along some direction) in the second and the third plots. With progressive scaling, the large margin solution increasingly deviates from the relative margin solution, showing a sensitivity to scaling transformations and, more generally, to affine transformations.

The bottom row of Figure 1 shows the projections of the data onto classification boundaries given by the relative margin and the absolute margin solutions. In the first solution, all square points are mapped to a narrow region while the points in the absolute margin solution are mapped to a wider region. With progressive scaling, the spread of the projection increases for the large margin solution.

While both solutions make zero errors on the training data, the goal of any learning algorithm is to generalize well to test data. Higher spread of the projection means a higher variance of the projected points on the real line. Thus, it is more likely that an error will be made from a future unseen test example since the high variance projections of the incorrect labels are more likely to result in a jump beyond the decision boundary (leading to a classification error). This motivation is similar in spirit to the motivation for Fisher Linear Discriminant Analysis. However, LDA merely uses second order statistics leading to inferior performance compared to standard SVMs. In this paper, relative margin is achieved by trading off the margin and the spread of the projections.

This example suggests that large margin is not sufficient by itself; it is meaningful only with respect to the corresponding spread of the data. A large margin seems of lesser significance if the data is also widely spread orthogonal to the decision plane. However, a margin that is not as large in the absolute sense can still be significant if the data is not highly spread orthogonally to the decision plane. Roughly, this is the idea behind most of the recent papers in the line of research [5, 9, 7, 8, 17, 13, 11, 12] mentioned in the introduction. This intuition agrees with the margin to radius ratio motivating SVMs, however, the radius is a generic way of measuring spread as it holds for any projection direction. The margin to spread ratio focuses along the particular projection found by the classifier and motivates the RMM approach in this article.

In the case of the SVM, [12] empirically showed that

[1] For brevity, the set $\{1, 2, \ldots, n\}$ is denoted by N.

[2] For brevity, the margin rescaling and slack rescaling extensions are omitted; they are straightforward extensions.

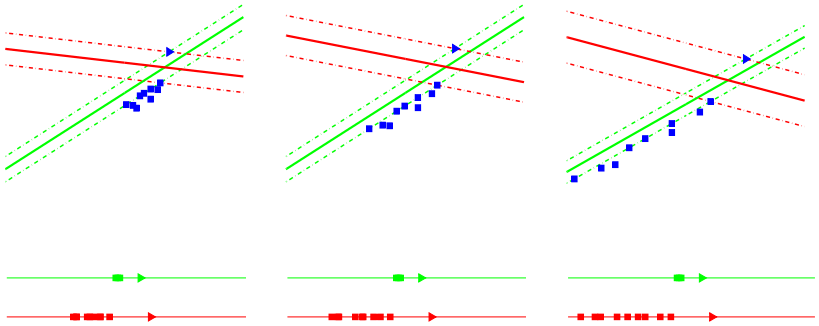

Figure 1. Top: Triangular points represent $\psi(\mathbf{x}_i, \mathbf{y}_i)$ and squares represent $\psi(\mathbf{x}_i, \mathbf{y})$ for $\mathbf{y} \in \mathcal{Y} \setminus \{\mathbf{y}_i\}$. As the data is scaled along the x-axis, the red (or dark shade) large absolute margin solution deviates from the green (or light shade) large relative margin solution. Bottom: The projections of the examples in the top row on the real line for the large margin and the relative margin solutions in each case. Projections have been drawn at different scales for clarity.

"whitening" the data with the covariance matrix can overcome the problem associated with absolute margin to some extent. That paper also showed that whitening in a feature space is computationally prohibitive in the generic case since it involves a matrix inversion. In the case of structured prediction there is an additional problem. The mappings considered are the joint feature maps $\psi(\mathbf{x}_i, \mathbf{y})$ for all the different possible values of $\mathbf{y} \in \mathcal{Y}$. There could be exponentially many or even infinitely many such points for a given problem. Thus, clearly, any such simple preprocessing step considering all these joint feature mappings is expensive. Similarly, EKM [11] and GMM [8] are expensive due to logdet optimization. Thus, we consider extending the RMM [12], which is a quadratic optimization for the binary classification case, to structured prediction problems.

The key idea of RMM is to maximize the margin while bounding the spread of the projections. In the structured prediction case, the idea is to separate the "winner" $< \mathbf{w}, \psi(\mathbf{x}_i, \mathbf{y}_i) >$ from the "runner up" $\max_{\mathbf{y} \neq \mathbf{y}_i} < \mathbf{w}, \psi(\mathbf{x}_i, \mathbf{y}) >$ with a large margin. Thus, the corresponding goal of a relative margin implementation would be to separate the winner from the runner up with a large margin with respect to the separation between the winner and the "worst contender" $\min_{\mathbf{y} \neq \mathbf{y}_i} < \mathbf{w}, \psi(\mathbf{x}_i, \mathbf{y}) >$.

The cutting plane algorithm for StructSVM exploits the fact that the $\min < \mathbf{w}, \psi(\mathbf{x}_i, \mathbf{y}) >$ over $\mathbf{y}$ is efficiently computable even if the output space $\mathcal{Y}$ is exponentially large. Bounding the separation between the winner and the worst contender would require the max of $< \mathbf{w}, \psi(\mathbf{x}_i, \mathbf{y}) >$ over $\mathbf{y}$ as well. In fact, in most structured spaces, only the minimum and maximum over $\mathcal{Y}$ of the function $< \mathbf{w}, \psi(\mathbf{x}_i, \mathbf{y}) >$ are estimable efficiently while estimating the mean, median or variance of $< \mathbf{w}, \psi(\mathbf{x}_i, \mathbf{y}) >$ over $\mathcal{Y}$ might be NP-hard or computationally prohibitive. This particular property will be exploited to obtain an efficient algorithm for the structured prediction RMM.

4 Structured RMM

Section 3 indicates that finding a large relative margin solution requires a trade-off between large margin and small spread of the projections of the data. This intuition leads to a natural combination of RMM from [12] and the StructSVM (formulation (1)) approach. Furthermore, the resulting method can be solved by a cutting plane algorithm with nearly the same computational effort compared to StructSVM. Surprisingly, however, this relative margin variant of structured prediction also leads to dramatic improvements in classification accuracy.

The limitation of the spread in the RMM can naturally be combined in structured prediction settings by maintaining the constraints from the formulation (1) while also incorporating an upper bound B on the difference in projections:

$$\min_{\mathbf{w},\xi} \frac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle + \frac{C}{n} \sum_{i=1}^{n} \xi_i \tag{2}$$

$$\begin{aligned} \text{s.t } \langle \mathbf{w}, \boldsymbol{\delta\psi}_i(\mathbf{y}) \rangle \geq 1 - \xi_i,\ \xi_i \geq 0 \quad & \forall i \in N, \mathbf{y} \neq \mathbf{y}_i \\ -B \leq \langle \mathbf{w}, \boldsymbol{\delta\psi}_i(\mathbf{y}) \rangle \leq B \quad & \forall i \in N, \forall \mathbf{y} \in \mathcal{Y}. \end{aligned}$$

The original structured prediction constraints in Equation (2) are referred to as the classification constraints while the added constraints will be referred to as the bounding constraints. Here, $B > 1$ is an additional parameter in the new framework which trades off between margin maximization and spread minimization by bounding the projections. When $B = \infty$, the StructSVM method is exactly recovered. For other settings of B, the new formulation may produce solutions that improve generalization accuracy. Although the parameter B seems somewhat ad-hoc, experiments in Section 5 show that it can be set systematically as

with other parameters in machine learning algorithms (e.g. the C value in StructSVM) and is easily adjusted using validation.

While the primal problem above is useful for deriving the algorithm, in practice, the dual of (2) is solved. The cutting plane algorithm is a standard tool for optimizing the dual and only kernel evaluations between the pairs $\boldsymbol{\delta\psi}_i(\mathbf{y})$ are required. The dual of (2) can be shown to be:

$$\max_{\alpha,\beta,\beta^*} \sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})} - B\sum_{i,\mathbf{y}} \beta_{(i\mathbf{y})} - B\sum_{i,\mathbf{y}} \beta^*_{(i\mathbf{y})} \quad (3)$$

$$-\frac{1}{2}\Bigg\langle \sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}) - \sum_{i,\mathbf{y}} \beta_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}) + \sum_{i,\mathbf{y}} \beta^*_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}), \sum_{j,\mathbf{y}\neq\mathbf{y}_j} \alpha_{(j\mathbf{y})}\boldsymbol{\delta\psi}_j(\mathbf{y}) - \sum_{j,\mathbf{y}} \beta_{(j\mathbf{y})}\boldsymbol{\delta\psi}_j(\mathbf{y}) + \sum_{j,\mathbf{y}} \beta^*_{(j\mathbf{y})}\boldsymbol{\delta\psi}_j(\mathbf{y})\Bigg\rangle$$

$$\text{s.t.} 0 \leq \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})} \leq \frac{C}{n} \quad \forall i \in N,$$

$$\alpha_{(i\mathbf{y})} \geq 0, \beta_{(i\mathbf{y})} \geq 0, \beta^*_{(i\mathbf{y})} \geq 0 \quad \forall i \in N,$$

where $\alpha_{(i\mathbf{y})}, \beta_{(i\mathbf{y})}$ and $\beta^*_{(i\mathbf{y})}$ are the Lagrange multipliers of the primal constraints. From the dual variables, $\mathbf{w}$ can be readily obtained as:

$$\sum_{i,\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}) - \sum_{i,\mathbf{y}} \beta_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}) + \sum_{i,\mathbf{y}} \beta^*_{(i\mathbf{y})}\boldsymbol{\delta\psi}_i(\mathbf{y}).$$

4.1 Cutting Plane Algorithm

Similar to the algorithm for StructSVM, the cutting plane algorithm for StructRMM starts with an empty working set of primal constraints. The constraints in the primal formulation correspond to the variables in the dual formulation. Hence, the algorithms can be equivalently seen as starting with an empty working set of dual variables. At each step, the most violated constraint for the i^{th} example is found. If there is any such violation, the corresponding dual variable is added to the working set. The algorithm then optimizes the dual objective over the variables in the working set. The steps involved in both the StructSVM and the StructRMM implementations are shown in Algorithm 1.

Unlike the cutting plane algorithm for StructSVM, the algorithm for StructRMM requires both the maximum and the minimum of $\langle \mathbf{w}, \boldsymbol{\delta\psi}_i(\mathbf{y})\rangle$ over $\mathbf{y} \neq \mathbf{y}_i$. It is possible to efficiently find the maximum and the minimum structured prediction in many problems. For example, in label sequence learning problems, finding the above maximum (minimum) corresponds to finding the longest (shortest) paths in a graph induced by the emission and the transition probabilities. If there is an algorithm to find the longest path, the shortest path can be found merely by negating all weights. This is true for a variety of algorithms used in structured prediction such as Viterbi decoding, max-product belief propagation, maximum weight spanning tree estimation, and maximum weight matching.

As with the cutting plane algorithm for StructSVM, it is necessary to find the second maximum or the minimum possible label sequence to exclude the possibility of adding label $\mathbf{y}_i$ for the i^{th} example. This can be achieved efficiently [6] as well.

In sequence problems, the most violated constraints are found using dynamic programming which corresponds to the Viterbi decoding algorithm. Therefore, the steps of Algorithm 1 that involve an arg max or an argmin over $\mathbf{y}$ (which resides in a large space $\mathcal{Y}$ of possible outputs) can be solved efficiently via fast decoding algorithms.

4.2 Runtime

Recall the following lemma and the corollary from [16]:

Lemma 4.1 *Let* $\mathbf{J}$ *be a symmetric, positive semi-definite matrix, and define a concave objective in* γ $\theta(\gamma) = -\frac{1}{2}\gamma^\top \mathbf{J}\gamma + \langle \mathbf{h}, \gamma\rangle$ *, which is assumed to be bounded above. Assume that a solution* γ^0 *and an optimization direction* $\boldsymbol{\eta}$ *are given such that* $\langle \theta(\gamma^0), \boldsymbol{\eta}\rangle > 0$*. Then optimizing* θ *starting from* γ^0 *along the chosen direction* $\boldsymbol{\eta}$ *will increase the objective by:* $\frac{1}{2}\frac{\langle \nabla\theta(\gamma^0), \boldsymbol{\eta}\rangle^2}{\eta^\top \mathbf{J}\boldsymbol{\eta}} > 0$.

Corollary 4.2 *Under the same assumption as in Lemma 4.1 and for the special case of an optimization direction* $\boldsymbol{\eta} = e_r$*, the objective improves by* $\frac{1}{2\mathbf{J}_{rr}}\left(\frac{\partial\theta}{\partial\gamma_r}\right)^2 > 0$.

First, note that, if in the cutting plane algorithms a classification constraint is the worst violator, the expression for improvements from [16] still hold in each step. The improvement is given by,

$$\min\left\{\frac{C\epsilon}{2n}, \frac{\epsilon^2}{8R_i^2}\right\} \quad (4)$$

where $R_i = \max_{\mathbf{y}}\{\|\boldsymbol{\delta\psi}_i(\mathbf{y})\|\}$.

Note that the Lagrange multipliers corresponding to the bounding constraints in the primal StructRMM formulations can only be positive; there are no additional constraints on them. Thus, when the worst violations in the cutting plane algorithm are bounding constraints, we can simply apply Corollary 4.2. Thus, an expression can be derived for the improvement obtained by a step of the cutting plane method in Algorithm 1.

First, γ is identified with the variables α, β and β^* in (3). However, note that only a subset of all these variables is added when the dual is solved using the cutting plane algorithm. $\mathbf{J}$ is identified with the positive semi-definite kernel

Algorithm 1 Cutting plane algorithm for StructRMM.

Require: $(\mathbf{x}_i, y_i)_{i=1}^n$, Parameters: $C, B, \epsilon, \epsilon_B$.

1: $S_i^1 \leftarrow \emptyset$, $S_i^2 \leftarrow \emptyset$, and $S_i^3 \leftarrow \emptyset$ forall $1 \leq i \leq n$
2: **repeat**
3: **for** $i \leftarrow 1$ to n **do**
4: $flag \leftarrow 0$
5: $\tilde{\mathbf{y}}_1 \leftarrow \arg\max_{\mathbf{y}\in\mathcal{Y}} H(\mathbf{y}) := 1 - \langle \boldsymbol{\delta\psi}_i(\mathbf{y}), \mathbf{w}\rangle$
6: $\tilde{\mathbf{y}}_2 \leftarrow \arg\max_{\mathbf{y}\in\mathcal{Y}} G(\mathbf{y}) := \langle \boldsymbol{\delta\psi}_i(\mathbf{y}), \mathbf{w}\rangle$
7: $\tilde{\mathbf{y}}_3 \leftarrow \arg\min_{\mathbf{y}\in\mathcal{Y}} -G(\mathbf{y})$
8: $\{$ where $\mathbf{w} = \sum_j \sum_{\mathbf{y}'\in S_j} \alpha_{(j\mathbf{y}')} \boldsymbol{\delta\psi}_j(\mathbf{y}') - \sum_j \sum_{\mathbf{y}'\in U_j} \beta_{(j\mathbf{y}')} \boldsymbol{\delta\psi}_j(\mathbf{y}') + \sum_j \sum_{\mathbf{y}'\in U_j} \beta^*_{(j\mathbf{y}')} \boldsymbol{\delta\psi}_j(\mathbf{y}')$ $\}$
9: $\xi_i \leftarrow \max(0, \max_{\mathbf{y}\in S_i} H(\mathbf{y}))$
10: $V_1 \leftarrow H(\tilde{\mathbf{y}}_1) - \xi_i - \epsilon$, $V_2 \leftarrow G(\tilde{\mathbf{y}}_2) - B - \epsilon_B$, $V_3 \leftarrow -G(\tilde{\mathbf{y}}_3) - B - \epsilon_B$
11: $j \leftarrow \arg\max_{i\in\{1,2,3\}} V_i$
12: **if** $V_j > 0$ **then** $S_i^j \leftarrow S_i^j \cup \{\tilde{\mathbf{y}}_j\}$, $flag \leftarrow 1$ **end if**
13: **if** $flag$ **equal to** 1 **then** Optimize the dual over S^1, S^2 and S^3, update α, β, β^* **end if**
14: **end for**
15: **until** no variables are added to S_i^1, S_i^2 or S_i^3 for any i

matrix whose ij^{th} term is formed from the kernel between $\boldsymbol{\delta}\psi_i(\mathbf{y})$ and $\boldsymbol{\delta}\psi_j(\mathbf{y})$ for two of the added constraints in the cutting plane algorithm.

Adding a dual variable in the cutting plane algorithm simply corresponds to optimizing over an axis parallel direction e_r. As stated before, it is only necessary to consider adding a dual variable β or β^* in (3) since the improvement in the objective is already known when a classification constraint is added.

Suppose a dual variable $\beta_{(i\mathbf{y})}$ is added in the cutting plane algorithm to the set S^2, then: $\frac{\partial\theta}{\partial\beta_{(i\mathbf{y})}}(\gamma^0) = -B + \langle \mathbf{w}^*, \boldsymbol{\delta\psi}_i(\mathbf{y})\rangle$, where $\mathbf{w}^*$ is the current solution of the cutting plane algorithm.

The fact that this dual variable was added to the working set in Step 15 to S^2 implies that $\langle \mathbf{w}^*, \boldsymbol{\delta\psi}_i(\mathbf{y})\rangle > B + \epsilon_B$, since only violating constraints result in such additions to the working sets. As a consequence, $\frac{\partial\theta}{\partial\beta_{(i\mathbf{y})}}(\gamma^0) > \epsilon_B$. Finally, $\mathbf{J}_{(i\mathbf{y})(i\mathbf{y})} = \langle \boldsymbol{\delta\psi}_i(\mathbf{y}), \boldsymbol{\delta\psi}_i(\mathbf{y})\rangle \leq R_i^2$.

It is now possible to insert both terms in Corollary 4.2 to get an expression for the minimum improvement achieved when an element is added to S^2. Thus: $\boldsymbol{\delta}\theta(\gamma^0) \geq \frac{1}{2}\frac{\epsilon_B^2}{R_i^2}$. Exactly the same improvement is achieved when an element is added to S^3 in a similar way.

Once the improvement in the objective is known for both types of constraints, the overall guaranteed improvement is the minimum of the two possible improvements in each step. Thus, considering (4), the improvement in each step is: $\min\left\{\frac{1}{2}\frac{\epsilon_B^2}{R_i^2}, \frac{C\epsilon}{2n}, \frac{\epsilon^2}{8R_i^2}\right\}$.

Finally, observe that the initial value of the dual objective is 0 when no variables are in the working sets. Since the dual objective is upper bounded by the primal objective, C is an upper bound. Therefore, it is possible to calculate the number of steps required for Algorithm 1 to terminate. The above arguments are summarized in the following theorem.

Theorem 4.3 *With $\bar{R} = \max_i R_i$, for any $\epsilon > 0$, $\epsilon_B > 0$, Algorithm 1 terminates after adding $\max\left\{\frac{2\bar{R}^2 C}{\epsilon_B^2}, \frac{2n}{\epsilon}, \frac{8C\bar{R}^2}{\epsilon^2}\right\}$ constraints to the working sets S^1, S^2 or S^3.*

In practice, the implementation of Algorithm 1 can actually add all the top violating constraints of each type for each datum i; this may further speedup the algorithm since jointly optimizing over several variables can give bigger improvements than optimizing over a single variable. Furthermore, the upper bound on the number of steps of the cutting plane method is very conservative and does not show any direct dependence of B. However, in practice, smaller B values require more time to converge. Nevertheless, while the theoretical bounds on convergence are helpful, practical experiments show that the new method, just like the StructSVM, remains efficient and, surprisingly, brings improvements in accuracy as well.

5 Experiments

This section provides empirical justification for the proposed algorithm on label sequence learning and multi-class problems. [2] showed that discriminative methods typically performed better than generative approaches on the label sequence learning problems. From the results from [2], it is clear that StructSVM improved over a regularized version of the CRF. Thus, the following experiments investigate StructSVM, StructRMM and a regularized CRF in label sequence learning problems. Comparisons are directly

	NER	POS
CRF	5.13 ± 0.28	11.34 ± 0.64
StructSVM	5.09 ± 0.32	11.14 ± 0.60
StructRMM	$\mathbf{5.05 \pm 0.28}$	$\mathbf{10.42 \pm 0.47}$
p-value	0.07	0.00

Table 1. Error rates (mean $\pm$ standard deviation) and p-values on the label sequence learning tasks. The improvements of StructRMM over StructSVM are of the same (or higher) order as the improvements of StructSVM over CRFs.

between StructSVM and StructRMM in the case of multi-class classification problems. In all the experiments, features are normalized to a *zero-one box*.

5.1 Label Sequence Learning

In label sequence learning problems, the label set $\mathcal{Y}$ is a fixed alphabet containing the labels $l_1, l_2, \cdots, l_k$. The training set consists of $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ where $\mathbf{y}_i$ is a sequence from the alphabet $\mathcal{Y}$. That is, $\mathbf{y}_i = (y_i^1, y_i^2, \cdots, y_i^{s_i})$, where, the length s_i of sequence i need not be constant across all examples. Considering the length of $\mathbf{y}$ to be T, the joint feature map for this problem is

$$\psi(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} \sum_{t=1}^{T} \phi(\mathbf{x}^t) \otimes \lambda(y^t) \\ \sum_{t=1}^{T-1} \lambda(y^t) \otimes \lambda(y^{t+1}) \end{pmatrix},$$

where $\lambda(y^t) = [\delta(l_1, y^t), \delta(l_2, y^t) \cdots \delta(l_k, y^t)]^\top$. This feature map mimics the hidden Markov model (HMM) by considering the adjacent labels. These labels are equivalent to the hidden states in the HMM and the features are equivalent to HMM outputs. The kernel between two joint feature maps thus becomes:

$$\langle \psi(\mathbf{x}, \mathbf{y}), \psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \rangle = \sum_{s=1}^{S} \sum_{t=1}^{T} \delta(y^t, \bar{y}^s) K(\mathbf{x}^t, \bar{\mathbf{x}}^s)$$
$$+ \sum_{s=1}^{S-1} \sum_{t=1}^{T-1} \delta(y^t, \bar{y}^s) \delta(y^{t+1}, \bar{y}^{s+1}),$$

where S is the length of the sequence $\bar{\mathbf{y}}$.

To evaluate the sequence learning method, the first experiment involves the Named Entity Recognition (NER) problem. A popular dataset for this task is from the Spanish news wire articles provided to the special NER session of the CoNLL 2002 conference. The words in these sentences are tagged with one of nine possible labels and the task is to predict the labels for each test sentence. In a second experiment, the Parts-of-Speech tagging task was considered using the sentences from the PennTree bank dataset. In this case, each word is labeled with one of forty five possible parts-of-speech and the task is to accurately predict these tags for unseen test examples. In both these experiments, the features used and the experimental setup are similar. The features describing words are all binary and follow the same protocols as in previous work [2].

A random collection of 240 examples served as the training set and two random subsets of size 1000 served as the validation and the test sets. As in previous work, the polynomial kernel of degree two was used. The StructSVM was first trained using the cutting plane algorithm. After training, the maximum value of $| < \mathbf{w}, \boldsymbol{\delta\psi}_i(\mathbf{y}) > |$ over all $\mathbf{y} \in \mathcal{Y}$, for any i, was noted. Subsequently, to evaluate the StructRMM, the B values were changed so that the maximum (minimum) allowed value of $< \mathbf{w}, \boldsymbol{\delta\psi}_i(\mathbf{y}) >$ ($- < \mathbf{w}, \boldsymbol{\delta\psi}_i(\mathbf{y}) >$) was within a fraction of the maximum found by the StructSVM. After training, the error rates were obtained on both validation and test sets for each setting of the parameters B and C. The setting which gave the lowest error on the validation set was used to determine the error rate on the test set (and vice versa). The experiment was repeated ten times over random draws of various sets and the results were averaged.

Table 1 shows the results on both the tasks for CRF, StructSVM and StructRMM. The improvement of StructRMM over StructSVM on the Named Entity Recognition task is nearly the same as that of StructSVM over CRF. However, in the case of the parts-of-speech tagging experiment, the improvement of StructRMM over StructSVM is much higher than that of StructSVM over CRF. Statistical significance tests were conducted with the results from different folds for StructSVM and StructRMM. Small p-values obtained from a paired t-test indicate that these improvements are statistically significant.

5.2 Multi-class classification

For the multiclass-classification, the domain $\mathcal{Y}$ is a set with k possible values, for example $\{1, 2, 3, \cdots k\}$. Given $(\mathbf{x}, \mathbf{y})$ where $\mathbf{y}_i$ is one of the k possible classes, the joint feature map for this problem is defined as the following: $\psi(\mathbf{x}, \mathbf{y}) := [\delta(1, \mathbf{y})\mathbf{x}^\top \delta(2, \mathbf{y})\mathbf{x}^\top \cdots \delta(k, \mathbf{y})\mathbf{x}^\top]^\top$. The joint feature map merely makes k "copies" of the features $\mathbf{x}$ but retains only one which is at the position of the position of the label $\mathbf{y}$. With this definition of the joint feature-map, the kernel between $(\mathbf{x}, \mathbf{y})$ and $(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ can be easily calculated: $\langle \psi(\mathbf{x}, \mathbf{y}), \psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \rangle = \sum_{i=1}^{k} \delta(i, \mathbf{y}) \delta(i, \bar{\mathbf{y}}) K(\mathbf{x}, \bar{\mathbf{x}})$. In the multi-class problems, the arg max (arg min) in the cutting plane algorithm 1 can be found by simple enumeration due to the small size of the output space.

For this experiment, the Optical Digits dataset [3] which contains 3823 digits represented via 64 scalar features that

Kernel	StructSVM	StructRMM	p-value
Poly 1	3.78 ± 0.54	3.85 ± 0.62	0.55
Poly 2	2.11 ± 0.43	$\mathbf{1.46 \pm 0.34}$	0.00
Ploy 3	1.73 ± 0.37	$\mathbf{1.24 \pm 0.43}$	0.00
Poly 4	1.55 ± 0.45	$\mathbf{1.18 \pm 0.43}$	0.00

Table 2. Error rates (mean $\pm$ standard deviation) and p-values for the multi-class problem. The average improvement of StructRMM over StructSVM is approximately 28% when using nonlinear kernels.

measure pixel intensities was considered. The dataset was first normalized so that each feature was in the range $[0, 1]$. Half of the examples in the dataset were randomly drawn to form a training set and the remaining examples were divided, randomly, into two sets of equal size to form validation and test sets. The StructSVM (StructRMM) was trained by exploring various C (C, B) values. The setting of the parameters B and C which achieved minimum error on the validation set was used to pick an error rate for StructSVM and StructRMM from the test set (and vice versa). The entire experiment was repeated for ten different draws of the training, test and validation sets. Furthermore, the experiments were performed for polynomial kernels with degrees one, two, three and four.

The results are presented in Table 2. With the linear kernel, StructRMM performed at the same level as StructSVM. But, with the polynomial kernel degrees two, three and four, the performance of StructRMM was significantly better than that of StructSVM. As before, statistical significance tests were conducted using the paired t-test. The tests suggest that there is no significant difference between StructSVM and StructRMM with linear kernel. However, for higher degree kernels, small p-values indicate that the improvement achieved by the StructRMM is statistically significant. In fact, the average improvement of StructRMM over StructSVM with these three kernels is about 28%.

6 Conclusions

A shortcoming of the large absolute margin method and its sensitivity to scaling was pointed out for structured prediction. Subsequently, a formulation was proposed to maximize the margin relative to the spread of the data. An efficient cutting plane algorithm, which can be solved in nearly same time as the StructSVM, was then proposed. Experiments on label sequence learning problems show that the improvement of StructRMM over StructSVM is of the same order as that of StructSVM over CRF. Dramatic improvements were obtained on multi-class classification problems as well. Additionally, the improvement in accuracy brought forth by the relative margin method does not carry significant increases in computational complexity.

[2]Work supported in part by National Science Foundation Grant IIS-0347499.

References

[1] Y. Altun, T. Hofmann, and M. Johnson. Discriminative learning for label sequence via boosting. In *In Advances in Neural Information Processing Systems 15*, 2003.

[2] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *In Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[3] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[4] G. Bakir, T. Hofmann, B. Schölkopf, A. J. Smola, B. Taskar, and S.V.N. Vishwanathan, editors. *Predicting Structured Data*. MIT Press, Cambridge, MA, 2006.

[5] N. Cesa-Bianchi, A. Conconi, and C. Gentile. A second-order perceptron algorithm. In *in Proc. 5th Annu. Conf. Computational Learning Theory (Lecture Notes in Artificial Intelligence). Berlin, Germany*, volume 2375, pages 121–137, 2002.

[6] Y-L Chow and R Schwartz. The n-best algorithm: an efficient procedure for finding top n sentence hypotheses. In *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal processing*, pages 81–84, 1991.

[7] K. Crammer, M. Dredze, and F. Pereira. Exact convex confidence-weighted learning. In *Advances in Neural Information Processing Systems 21*, Cambridge, MA, 2009. MIT Press.

[8] K. Crammer, M. Mohri, and F. Pereira. Gaussian margin machines. In *Proceedings of the Artificial Intelligence and Statistics*, 2009.

[9] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Internationcal Conference on Machine Leaarning*, 2008.

[10] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *In Proceedings of the Eighteenth International Conference on Machine Learning*, pages 282–289, 2001.

[11] P. K. Shivaswamy and T. Jebara. Ellipsoidal kernel machines. In *Proceedings of the Artificial Intelligence and Statistics*, 2007.

[12] P. K. Shivaswamy and T. Jebara. Relative margin machines. In *Advances in Neural Information Processing Systems 21*, Cambridge, MA, 2009. MIT Press.

[13] F. Sinz, O. Chapelle, A. Agarwal, and B. Schölkopf. An analysis of inference with the universum. In *Advances in Neural Information Processing Systems 20*, pages 1369–1376. MIT Press, Cambridge, MA, 2008.

[14] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. *In Advances in Neural Information Processing Systems 16*, 2004.

[15] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning (ICML)*, 2004.

[16] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6:1453–1484, September 2005.

[17] J. Weston, R. Collobert, F. H. Sinz, L. Bottou, and V. Vapnik. Inference with the universum. In *Proceedings of the International Conference on Machine Learning*, pages 1009–1016, 2006.