Support Vector Machine Learning for Interdependent and Structured Output Spaces

Ioannis Tsochantaridis IT@CS.BROWN.EDU
Thomas Hofmann TH@CS.BROWN.EDU
Department of Computer Science, Brown University, Providence, RI 02912

Thorsten Joachims TJ@CS.CORNELL.EDU
Department of Computer Science, Cornell University, Ithaca, NY 14853

Yasemin Altun ALTUN@CS.BROWN.EDU
Department of Computer Science, Brown University, Providence, RI 02912

Abstract

Learning general functional dependencies is one of the main goals in machine learning. Recent progress in kernel-based methods has focused on designing flexible and powerful input representations. This paper addresses the complementary issue of problems involving complex outputs such as multiple dependent output variables and structured output spaces. We propose to generalize multiclass Support Vector Machine learning in a formulation that involves features extracted jointly from inputs and outputs. The resulting optimization problem is solved efficiently by a cutting plane algorithm that exploits the sparseness and structural decomposition of the problem. We demonstrate the versatility and effectiveness of our method on problems ranging from supervised grammar learning and named-entity recognition, to taxonomic text classification and sequence alignment.

1. Introduction

This paper deals with the general problem of learning a mapping from inputs $\mathbf{x} \in \mathcal{X}$ to discrete outputs $\mathbf{y} \in \mathcal{Y}$ based on a training sample of input-output pairs $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n) \in \mathcal{X} \times \mathcal{Y}$ drawn from some fixed but unknown probability distribution. Unlike the case of multiclass classification where $\mathcal{Y} = \{1, ..., k\}$ with interchangeable, arbitrarily numbered labels, we consider structured output spaces $\mathcal{Y}$. Elements $\mathbf{y} \in \mathcal{Y}$ may be, for instance, sequences, strings, labeled trees, lattices, or graphs. Such problems arise in a variety of applications, ranging from multilabel classification and classification with class taxonomies, to label sequence learning, sequence alignment learning, and supervised grammar learning, to name just a few.

We approach these problems by generalizing large margin methods, more specifically multi-class Support Vector Machines (SVMs) (Weston & Watkins, 1998; Crammer & Singer, 2001), to the broader problem of learning structured responses. The naive approach of treating each structure as a separate class is often intractable, since it leads to a multiclass problem with a very large number of classes. We overcome this problem by specifying discriminant functions that exploit the structure and dependencies within $\mathcal{Y}$. In that respect, our approach follows the work of Collins (2002; 2004) on perceptron learning with a similar class of discriminant functions. However, the maximum margin algorithm we propose has advantages in terms of accuracy and tunability to specific loss functions. A similar philosophy of using kernel methods for learning general dependencies was pursued in Kernel Dependency Estimation (KDE) (Weston et al., 2003). Yet, the use of separate kernels for inputs and outputs and the use of kernel PCA with standard regression techniques significantly differs from our formulation, which is a more straightforward and natural generalization of multiclass SVMs.

2. Discriminants and Loss Functions

We are interested in the general problem of learning functions $f : \mathcal{X} \to \mathcal{Y}$ based on a training sample of input-output pairs. As an illustrating example, consider the case of natural language parsing, where the function f maps a given sentence $\mathbf{x}$ to a parse tree

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

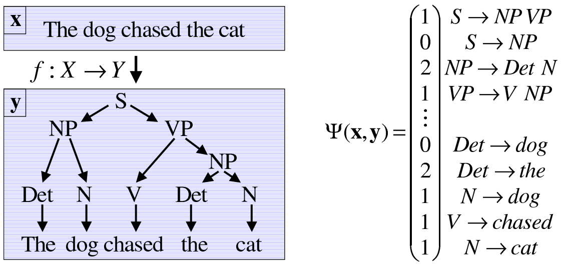

Figure 1. Illustration of natural language parsing model.

$\mathbf{y}$. This is depicted graphically in Figure 1. The approach we pursue is to learn a *discriminant function* $F : \mathcal{X} \times \mathcal{Y} \to \Re$ over input/output pairs from which we can derive a prediction by maximizing F over the response variable for a specific given input $\mathbf{x}$. Hence, the general form of our hypotheses f is

$$f(\mathbf{x}; \mathbf{w}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}) \,, \tag{1}$$

where $\mathbf{w}$ denotes a parameter vector. It might be useful to think of $-F$ as a $\mathbf{w}$-parameterized family of cost functions, which we try to design in such a way that the minimum of $F(\mathbf{x}, \cdot; \mathbf{w})$ is at the desired output $\mathbf{y}$ for inputs $\mathbf{x}$ of interest. Throughout this paper, we assume F to be linear in some *combined feature representation* of inputs and outputs $\Psi(\mathbf{x}, \mathbf{y})$,

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle \,. \tag{2}$$

The specific form of Ψ depends on the nature of the problem and special cases will be discussed subsequently.

Using again natural language parsing as an illustrative example, we can chose F such that we get a model that is isomorphic to a Probabilistic Context Free Grammar (PCFG). Each node in a parse tree $\mathbf{y}$ for a sentence $\mathbf{x}$ corresponds to grammar rule g_j, which in turn has a score w_j. All valid parse trees $\mathbf{y}$ (i.e. trees with a designated start symbol S as the root and the words in the sentence $\mathbf{x}$ as the leaves) for a sentence $\mathbf{x}$ are scored by the sum of the w_j of their nodes. This score can thus be written as $F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$, where $\Psi(\mathbf{x}, \mathbf{y})$ is a histogram vector counting how often each grammar rule g_j occurs in the tree $\mathbf{y}$. $f(\mathbf{x}; \mathbf{w})$ can be efficiently computed by finding the structure $\mathbf{y} \in Y$ that maximizes $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ via the CKY algorithm (see Manning and Schuetze (1999)).

Learning over structured output spaces $\mathcal{Y}$ inevitably involves loss functions other than the standard zero-one classification loss (cf. Weston et al. (2003)). For example, in natural language parsing, a parse tree that differs from the correct parse in a few nodes only should be treated differently from a parse tree that is radically different. Typically, the correctness of a predicted parse tree is measured by its F_1 score (see e.g. Johnson (1999)), the harmonic mean of precision of recall as calculated based on the overlap of nodes between the trees. We thus assume the availability of a bounded loss function $\triangle : \mathcal{Y} \times \mathcal{Y} \to \Re$ where $\triangle(\mathbf{y}, \hat{\mathbf{y}})$ quantifies the loss associated with a prediction $\hat{\mathbf{y}}$, if the true output value is $\mathbf{y}$. If $P(\mathbf{x}, \mathbf{y})$ denotes the data generating distribution, then the goal is to find a function f within a given hypothesis class such that the risk

$$\mathcal{R}_P^{\triangle}(f) = \int_{\mathcal{X} \times \mathcal{Y}} \triangle(\mathbf{y}, f(\mathbf{x}))\, dP(\mathbf{x}, \mathbf{y}) \,. \tag{3}$$

is minimized. We assume that P is unknown, but that a finite training set of pairs $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y} : i = 1, \ldots, n\}$ generated i.i.d. according to P is given. The performance of a function f on the training sample $\mathcal{S}$ is described by the empirical risk $\mathcal{R}_{\mathcal{S}}^{\triangle}(f)$. For $\mathbf{w}$-parameterized hypothesis classes, we will also write $\mathcal{R}_P^{\triangle}(\mathbf{w}) \equiv \mathcal{R}_P^{\triangle}(f(\cdot; \mathbf{w}))$ and similarly for the empirical risk.

3. Margins and Margin Maximization

First, we consider the separable case in which there exists a function f parameterized by $\mathbf{w}$ such that the empirical risk is zero. If we assume that $\triangle(\mathbf{y}, \mathbf{y}') > 0$ for $\mathbf{y} \neq \mathbf{y}'$ and $\triangle(\mathbf{y}, \mathbf{y}) = 0$, then the condition of zero training error can then be compactly written as a set of non-linear constraints

$$\forall i : \quad \max_{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i} \{\langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle\} < \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}_i) \rangle \,. \tag{4}$$

Each nonlinear inequalities in (4) can be equivalently replaced by $|\mathcal{Y}| - 1$ linear inequalities, resulting in a total of $n|\mathcal{Y}| - n$ linear constraints,

$$\forall i, \; \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle > 0 \,, \tag{5}$$

where we have defined the shorthand $\delta\Psi_i(\mathbf{y}) \equiv \Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y})$.

If the set of inequalities in (5) is feasible, there will typically be more than one solution $\mathbf{w}^*$. To specify a unique solution, we propose to select the $\mathbf{w}$ with $\|\mathbf{w}\| \leq 1$ for which the score of the correct label $\mathbf{y}_i$ is uniformly most different from the closest runner-up $\hat{\mathbf{y}}_i(\mathbf{w}) = \operatorname{argmax}_{\mathbf{y} \neq \mathbf{y}_i} \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle$. This generalizes the maximum-margin principle employed in SVMs (Vapnik, 1998) to the more general case considered in this paper. The resulting hard-margin optimization problem is

$$\mathrm{SVM}_0: \min_{\mathbf{w}} \quad \frac{1}{2}\|\mathbf{w}\|^2 \tag{6a}$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq 1\,. \tag{6b}$$

To allow errors in the training set, we introduce slack variables and propose to optimize a soft-margin criterion. While there are several ways of doing this, we follow Crammer and Singer (2001) and introduce one slack variable for every non-linear constraint (4), which will result in an upper bound on the empirical risk and offers some additional algorithmic advantages. Adding a penalty term that is linear in the slack variables to the objective results in the quadratic program

$$\mathrm{SVM}_1: \min_{\mathbf{w}, \boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i, \text{ s.t. } \forall i, \xi_i \geq 0 \tag{7a}$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq 1 - \xi_i\,. \tag{7b}$$

Alternatively, we can also penalize margin violations by a quadratic term $\frac{C}{2n}\sum_i \xi_i^2$ leading to an analogue optimization problem which we refer to as SVM_2. In both cases, $C > 0$ is a constant that controls the trade-off between training error minimization and margin maximization.

SVM_1 implicitly considers the zero-one classification loss. As argued above, this is inappropriate for problems like natural language parsing, where $|\mathcal{Y}|$ is large. We now propose two approaches that generalize the above formulations to the case of arbitrary loss functions $\triangle$. Our first approach is to *re-scale the slack variables* according to the loss incurred in each of the linear constraints. Intuitively, violating a margin constraint involving a $\mathbf{y} \neq \mathbf{y}_i$ with high loss $\triangle(\mathbf{y}_i, \mathbf{y})$ should be penalized more severely than a violation involving an output value with smaller loss. This can be accomplished by multiplying the violation by the loss, or equivalently, by scaling slack variables with the inverse loss, which yields the problem

$$\mathrm{SVM}_1^{\triangle s}: \min_{\mathbf{w}, \boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i, \text{ s.t. } \forall i, \xi_i \geq 0 \tag{8}$$

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq 1 - \frac{\xi_i}{\triangle(\mathbf{y}_i, \mathbf{y})}. \tag{9}$$

A justification for this formulation is given by the subsequent proposition (proof omitted).

Proposition 1. *Denote by $(\mathbf{w}^*, \boldsymbol{\xi}^*)$ the optimal solution to $SVM_1^{\triangle s}$. Then $\frac{1}{n}\sum_{i=1}^{n} \xi_i^*$ is an upper bound on the empirical risk $\mathcal{R}_{\mathcal{S}}^{\triangle}(\mathbf{w}^*)$.*

The optimization problem $\mathrm{SVM}_2^{\triangle s}$ can be derived analogously, where $\triangle(\mathbf{y}_i, \mathbf{y})$ is replaced by $\sqrt{\triangle(\mathbf{y}_i, \mathbf{y})}$ in order to obtain an upper bound on the empirical risk.

A second way to include loss functions is to *re-scale the margin* as proposed by Taskar et al. (2004) for the special case of the Hamming loss. The margin constraints in this setting take the following form:

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq \triangle(\mathbf{y}_i, \mathbf{y}) - \xi_i \tag{10}$$

This set of constraints yield an optimization problem $\mathrm{SVM}_1^{\triangle m}$ which also results in an upper bound on $\mathcal{R}_{\mathcal{S}}^{\triangle}(\mathbf{w}^*)$. In our opinion, a potential disadvantage of the margin scaling approach is that it may give significant weight to output values $\mathbf{y} \in \mathcal{Y}$ that are not even close to being confusable with the target values $\mathbf{y}_i$, because every increase in the loss increases the required margin.

4. Support Vector Machine Learning

The key challenge in solving the QPs for the generalized SVM learning is the large number of margin constraints; more specifically the total number of constraints is $n|\mathcal{Y}|$. In many cases, $|\mathcal{Y}|$ may be extremely large, in particular, if $\mathcal{Y}$ is a product space of some sort (e.g. in grammar learning, label sequence learning, etc.). This makes standard quadratic programming solvers unsuitable for this type of problem.

In the following, we propose an algorithm that exploits the special structure of the maximum-margin problem, so that only a much smaller subset of constraints needs to be explicitly examined. The algorithm is a generalization of the SVM algorithm for label sequence learning (Hofmann et al., 2002; Altun et al., 2003) and the algorithm for inverse sequence alignment (Joachims, 2003). We will show how to compute arbitrarily close approximations to all of the above SVM optimization problems in polynomial time for a large range of structures and loss functions. Since the algorithm operates on the dual program, we will first derive the Wolfe dual for the various soft margin formulations.

4.1. Dual Programs

We will denote by $\alpha_{i\mathbf{y}}$ the Lagrange multiplier enforcing the margin constraint for label $\mathbf{y} \neq \mathbf{y}_i$ and example $(\mathbf{x}_i, \mathbf{y}_i)$. Using standard Lagrangian duality techniques, one arrives at the following dual QP for the hard margin case SVM_0

$$\max_{\alpha} \sum_{i, \mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} - \frac{1}{2} \sum_{\substack{i, \mathbf{y} \neq \mathbf{y}_i \\ j, \bar{\mathbf{y}} \neq \mathbf{y}_j}} \alpha_{i\mathbf{y}} \alpha_{j\bar{\mathbf{y}}} \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \rangle \tag{11a}$$

$$\text{s.t. } \forall i, \forall \mathbf{y} \neq \mathcal{Y} \setminus \mathbf{y}_i : \quad \alpha_{i\mathbf{y}} \geq 0\,. \tag{11b}$$

A kernel $K((\mathbf{x},\mathbf{y}),(\mathbf{x}',\mathbf{y}'))$ can be used to replace the inner products, since inner products in $\delta\Psi$ can be easily expressed as inner products of the original Ψ-vectors.

For soft-margin optimization with slack re-scaling and linear penalties ($\mathrm{SVM}_1^{\triangle s}$), additional box constraints

$$n \sum_{\mathbf{y}\neq\mathbf{y}_i} \frac{\alpha_{i\mathbf{y}}}{\triangle(\mathbf{y}_i,\mathbf{y})} \leq C, \; \forall i \tag{12}$$

are added to the dual. Quadratic slack penalties (SVM_2) lead to the same dual as SVM_0 after altering the inner product to $\langle\delta\Psi_i(\mathbf{y}),\delta\Psi_j(\bar{\mathbf{y}})\rangle + \delta ij \frac{n}{C\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}\sqrt{\triangle(\mathbf{y}_j,\bar{\mathbf{y}})}}$. $\delta ij = 1$, if $i=j$, else 0.

Finally, in the case of margin re-scaling, the loss function affects the linear part of the objective function $\max_\alpha \sum_{i,\mathbf{y}} \alpha_{i\mathbf{y}}\triangle(\mathbf{y}_i,\mathbf{y}) - Q(\alpha)$ (where the quadratic part Q is unchanged from (11a)) and introduces standard box constraints $n\sum_{\mathbf{y}\neq\mathbf{y}_i}\alpha_{i\mathbf{y}} \leq C$.

4.2. Algorithm

The algorithm we propose aims at finding a small set of active constraints that ensures a sufficiently accurate solution. More precisely, it creates a nested sequence of successively tighter relaxations of the original problem using a cutting plane method. The latter is implemented as a variable selection approach in the dual formulation. We will show that this is a valid strategy, since there always exists a polynomially-sized subset of constraints so that the corresponding solution fulfills all constraints with a precision of at least ϵ. This means, the remaining – potentially exponentially many – constraints are guaranteed to be violated by no more than ϵ, without the need for explicitly adding them to the optimization problem.

We will base the optimization on the dual program formulation which has two important advantages over the primal QP. First, it only depends on inner products in the joint feature space defined by Ψ, hence allowing the use of kernel functions. Second, the constraint matrix of the dual program (for the L_1-SVMs) supports a natural problem decomposition, since it is block diagonal, where each block corresponds to a specific training instance.

Pseudocode of the algorithm is depicted in Algorithm 1. The algorithm applies to all SVM formulations discussed above. The only difference is in the way the cost function gets set up in step 5. The algorithm maintains a working set S_i for each training example $(\mathbf{x}_i,\mathbf{y}_i)$ to keep track of the selected constraints which define the current relaxation. Iterating through the training examples $(\mathbf{x}_i,\mathbf{y}_i)$, the algorithm proceeds by finding the (potentially) "most violated" constraint, involving some output value $\hat{\mathbf{y}}$ (line 6). If the (appropriately scaled) margin violation of this constraint exceeds the current value of ξ_i by more than ϵ (line 8), the dual variable corresponding to $\hat{\mathbf{y}}$ is added to the working set (line 9). This variable selection process in the dual program corresponds to a successive strengthening of the primal problem by a cutting plane that cuts off the current primal solution from the feasible set. The chosen cutting plane corresponds to the constraint that determines the lowest feasible value for ξ_i. Once a constraint has been added, the solution is re-computed wrt. S (line 10). Alternatively, we have also devised a scheme where the optimization is restricted to S_i only, and where optimization over the full S is performed much less frequently. This can be beneficial due to the block diagonal structure of the optimization problems, which implies that variables $\alpha_{j\mathbf{y}}$ with $j \neq i$, $\mathbf{y} \in S_j$ can simply be "frozen" at their current values. Notice that all variables not included in their respective working set are implicitly treated as 0. The algorithm stops, if no constraint is violated by more than ϵ. The presented algorithm is implemented and available[1] as part of *SVM*^{light}. Note that the SVM optimization problems from iteration to iteration differ only by a single constraint. We therefore restart the SVM optimizer from the current solution, which greatly reduces the runtime. A convenient property of both algorithms is that they have a very general and well-defined interface independent of the choice of Ψ

Algorithm 1 Algorithm for solving SVM_0 and the loss re-scaling formulations $\mathrm{SVM}_1^{\triangle s}$ and $\mathrm{SVM}_2^{\triangle s}$

1: Input: $(\mathbf{x}_1,\mathbf{y}_1),\ldots,(\mathbf{x}_n,\mathbf{y}_n)$, C, ϵ
2: $S_i \leftarrow \emptyset$ for all $i=1,\ldots,n$
3: **repeat**
4:   **for** $i=1,\ldots,n$ **do**
5:     set up cost function
      $\mathrm{SVM}_1^{\triangle s}$: $H(\mathbf{y}) \equiv (1-\langle\delta\Psi_i(\mathbf{y}),\mathbf{w}\rangle)\,\triangle(\mathbf{y}_i,\mathbf{y})$
      $\mathrm{SVM}_2^{\triangle s}$: $H(\mathbf{y}) \equiv (1-\langle\delta\Psi_i(\mathbf{y}),\mathbf{w}\rangle)\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}$
      $\mathrm{SVM}_1^{\triangle m}$: $H(\mathbf{y}) \equiv \triangle(\mathbf{y}_i,\mathbf{y}) - \langle\delta\Psi_i(\mathbf{y}),\mathbf{w}\rangle$
      $\mathrm{SVM}_2^{\triangle m}$: $H(\mathbf{y}) \equiv \sqrt{\triangle(\mathbf{y}_i,\mathbf{y})} - \langle\delta\Psi_i(\mathbf{y}),\mathbf{w}\rangle$
      where $\mathbf{w} \equiv \sum_j \sum_{\mathbf{y}'\in S_j} \alpha_{j\mathbf{y}'}\delta\Psi_j(\mathbf{y}')$.
6:     compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}\in Y} H(\mathbf{y})$
7:     compute $\xi_i = \max\{0, \max_{\mathbf{y}\in S_i} H(\mathbf{y})\}$
8:     **if** $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$ **then**
9:       $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
10:       $\alpha_S \leftarrow$ optimize dual over S, $S = \cup_i S_i$.
11:     **end if**
12:   **end for**
13: **until** no S_i has changed during iteration

[1] http://svmlight.joachims.org/

and $\triangle$. To apply the algorithm, it is sufficient to implement the feature mapping $\Psi(\mathbf{x}, \mathbf{y})$ (either explicit or via a joint kernel function), the loss function $\triangle(\mathbf{y}_i, \mathbf{y})$, as well as the maximization in step 6. All of those, in particular the constraint/cut selection method, are treated as black boxes. While the modeling of $\Psi(\mathbf{x}, \mathbf{y})$ and $\triangle(\mathbf{y}_i, \mathbf{y})$ is more or less straightforward, solving the maximization problem for constraint selection typically requires exploiting the structure of Ψ.

4.3. Analysis

It is straightforward to show that the algorithm finds a solution that is close to optimal (e.g. for the $\mathrm{SVM}_1^{\triangle s}$, adding ϵ to each ξ_i is a feasible point of the primal at most ϵC from the maximum). However, it is not immediately obvious how fast the algorithm converges. We will show in the following that the algorithm converges in polynomial time for a large class of problems, despite a possibly exponential or infinite $|\mathcal{Y}|$.

Let us begin with an elementary Lemma that will be helpful for proving subsequent results. It quantifies how the dual objective changes, if one optimizes over a single variable.

Lemma 1. *Let $\boldsymbol{J}$ be a positive definite matrix and let us define a concave quadratic program*

$$W(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{J}\boldsymbol{\alpha} + \langle \mathbf{h}, \boldsymbol{\alpha}\rangle \quad s.t.\ \boldsymbol{\alpha} \geq 0$$

and assume $\boldsymbol{\alpha} \geq 0$ is given with $\alpha_r = 0$. Then maximizing W with respect to α_r while keeping all other components fixed will increase the objective by

$$\frac{\left(h_r - \sum_s \alpha_s J_{rs}\right)^2}{2J_{rr}}$$

provided that $h_r \geq \sum_s \alpha_s J_{rs}$.

Proof. Denote by $\boldsymbol{\alpha}[\alpha_r \leftarrow \beta]$ the solution $\boldsymbol{\alpha}$ with the r-th coefficient changed to β, then

$$W(\boldsymbol{\alpha}[\alpha_r \leftarrow \beta]) - W(\boldsymbol{\alpha}) = \beta\left(h_r - \sum_s \alpha_s J_{rs}\right) - \frac{\beta^2}{2}J_{rr}$$

The difference is maximized for

$$\beta^* = \frac{h_r - \sum_s \alpha_s J_{rs}}{J_{rr}}$$

Notice that $\beta^* \geq 0$, since $h_r \geq \sum_s \alpha_s J_{rs}$ and $J_{rr} > 0$. $\square$

Using this Lemma, we can lower bound the improvement of the dual objective in step 10 of Algorithm 1. For brevity, let us focus on the case of $\mathrm{SVM}_2^{\triangle s}$. Similar results can be derived also for the other variants.

Proposition 2. *Define $\triangle_i = \max_{\mathbf{y}} \triangle(\mathbf{y}_i, \mathbf{y})$ and $R_i = \max_{\mathbf{y}} \|\delta\Psi_i(\mathbf{y})\|$. Then step 10 in Algorithm 1, improves the dual objective for $SVM_2^{\triangle s}$ at least by $\frac{1}{2}\epsilon^2(\triangle_i R_i^2 + n/C)^{-1}$.*

Proof. Using the notation in Algorithm 1 one can apply Lemma 1 with $r = (i, \hat{\mathbf{y}})$ denoting the newly added constraint, $h_r = 1$, $J_{rr} = \|\delta\Psi_i(\hat{\mathbf{y}})\|^2 + \frac{n}{C\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}$ and $\sum_s \alpha_s J_{rs} = \langle w, \delta\Psi_i(\hat{\mathbf{y}})\rangle + \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \frac{n}{C\sqrt{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}}$. Note that $\alpha_r = 0$. Using the fact that $\sum_{\mathbf{y}\neq\mathbf{y}_i} \frac{n\alpha_{i\mathbf{y}}}{C\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}} = \xi_i$, Lemma 1 shows the following increase of the objective function when optimizing over α_r alone:

$$\frac{\left(1 - \langle \mathbf{w}, \delta\Psi_i(\hat{\mathbf{y}})\rangle - \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \frac{n}{C\sqrt{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}}\right)^2}{2\left(\|\delta\Psi_i(\hat{\mathbf{y}})\|^2 + \frac{n}{C\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}\right)}$$
$$\geq \frac{\epsilon^2}{2\left(\|\delta\Psi_i(\hat{\mathbf{y}})\|^2\triangle(\mathbf{y}_i,\hat{\mathbf{y}}) + \frac{n}{C}\right)}$$

The step follows from the fact that $\xi_i \geq 0$ and $\sqrt{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}(1 - \langle \mathbf{w}, \delta\Psi_i(\hat{\mathbf{y}})\rangle) > \xi_i + \epsilon$, which is the condition of step 8. Replacing the quantities in the denominator by their upper limit proves the claim, since jointly optimizing over more variables than just α_r can only further increase the dual objective. $\square$

This leads to the following polynomial bound on the maximum size of S.

Theorem 1. *With $\bar{R} = \max_i R_i$, $\bar{\triangle} = \max_i \triangle_i$ and for a given $\epsilon > 0$, Algorithm 1 for the $SVM_2^{\triangle s}$ terminates after incrementally adding at most $\epsilon^{-2}(C\bar{\triangle}^2\bar{R}^2 + n\bar{\triangle})$ constraints to the working set S.*

Proof. With $S = \emptyset$ the optimal value of the dual is 0. In each iteration a constraint $(i, \mathbf{y})$ is added that is violated by at least ϵ, provided such a constraint exists. After solving the S-relaxed QP in step 10, the objective will increase by at least $\frac{1}{2}\epsilon^2(\bar{\triangle}\bar{R}^2 + n/C)^{-1}$ according to Proposition 2. Hence after t constraints, the dual objective will be at least t times this amount. The result follows from the fact that the dual objective is upper bounded by the minimum of the primal, which in turn can be bounded by $\frac{1}{2}C\bar{\triangle}$. $\square$

Note that the number of constraints in S does not depend on $|\mathcal{Y}|$. This is crucial, since $|\mathcal{Y}|$ is exponential or infinite for many interesting problems. For problems where step 6 can be computed in polynomial time, the overall algorithm has a runtime polynomial in $n, \bar{R}, \bar{\triangle}$, $1/\epsilon$, since at least one constraint will be added while cycling through all n instances and since step 10 is polynomial.

5. Applications and Experiments

To demonstrate the effectiveness and versatility of our approach, we report results on a number of different tasks To adapt the algorithm to a new problem, it is sufficient to implement the feature mapping $\Psi(\mathbf{x}, \mathbf{y})$, the loss function $\triangle(\mathbf{y}_i, \mathbf{y})$, as well as the maximization in step 6.

5.1. Multiclass Classification

Our algorithm can implement the conventional winner-takes-all (WTA) multiclass classification (Crammer & Singer, 2001) as follows. Let $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_K\}$ and $\mathbf{w} = (\mathbf{v}'_1, \ldots, \mathbf{v}'_K)'$ is a stack of vectors, $\mathbf{v}_k$ being a weight vector associated with the k-th class $\mathbf{y}_k$. Following Crammer and Singer (2001) one can then define $F(\mathbf{x}, \mathbf{y}_k; \mathbf{w}) = \langle \mathbf{v}_k, \Phi(\mathbf{x}) \rangle$, where $\Phi(\mathbf{x}) \in \Re^D$ denotes an arbitrary input representation. These discriminant functions can be equivalently represented in the proposed framework by defining a joint feature map as follows $\Psi(\mathbf{x}, \mathbf{y}) \equiv \Phi(\mathbf{x}) \otimes \Lambda^c(\mathbf{y})$. Here Λ^c refers to the orthogonal (binary) encoding of the label $\mathbf{y}$ and $\otimes$ is the tensor product which forms all products between coefficients of the two argument vectors.

5.2. Classification with Taxonomies

The first generalization we propose is to make use of more interesting output features Λ than the orthogonal representation Λ^c. As an exemplary application of this kind, we show how to take advantage of known class taxonomies. Here a taxonomy is treated as a lattice in which the classes $\mathbf{y} \in \mathcal{Y}$ are the minimal elements. For every node z in the lattice (corresponding to a super-class or class) we introduce a binary attribute $\lambda_z(\mathbf{y})$ indicating whether or not z is a predecessor of $\mathbf{y}$. Notice that $\langle \Lambda(\mathbf{y}), \Lambda(\mathbf{y}') \rangle$ will count the number of common predecessors.

We have performed experiments using a document collection released by the World Intellectual Property Organization (WIPO), which uses the International Patent Classification (IPC) scheme. We have restricted ourselves to one of the 8 sections, namely section D, consisting of 1,710 documents in the WIPO-alpha collection. For our experiments, we have indexed the title and claim tags. We have furthermore subsampled the training data to investigate the effect of the training set size. Document parsing, tokenization and term normalization have been performed with the MindServer retrieval engine.[2] As a suitable loss function $\triangle$, we have used a tree loss function which defines the loss between two classes $\mathbf{y}$ and $\mathbf{y}'$ as the height of the first common ancestor of $\mathbf{y}$ and $\mathbf{y}'$ in the taxonomy. The results are summarized in Table 1 and show that the proposed hierarchical SVM learning architecture improves performance over the standard multiclass SVM in terms of classification accuracy as well as in terms of the tree loss.

Table 1. Results on the WIPO-alpha corpus, section D with 160 groups using 3-fold and 5-fold cross validation, respectively. 'flt' is a standard (flat) SVM multiclass model, 'tax' the hierarchical architecture. '0/1' denotes training based on the classification loss, '$\triangle$' refers to training based on the tree loss.

	flt 0/1	tax 0/1	flt $\triangle$	tax $\triangle$	
4 training instances per class					
acc	28.32	28.32	27.47	29.74	+5.01 %
$\triangle$-loss	1.36	1.32	1.30	1.21	+12.40 %
2 training instances per class					
acc	20.20	20.46	20.20	21.73	+7.57 %
$\triangle$-loss	1.54	1.51	1.39	1.33	+13.67 %

5.3. Label Sequence Learning

Label sequence learning deals with the problem of predicting a sequence of labels $\mathbf{y} = (y^1, \ldots, y^m)$, $y^k \in \Sigma$, from a given sequence of inputs $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^m)$. It subsumes problems like segmenting or annotating observation sequences and has widespread applications in optical character recognition, natural language processing, information extraction, and computational biology. In the following, we study our algorithm on a named entity recognition (NER) problem. More specifically, we consider a sub-corpus consisting of 300 sentences from the Spanish news wire article corpus which was provided for the special session of CoNLL2002 devoted to NER. The label set in this corpus consist of non-name and the beginning and continuation of person names, organizations, locations and miscellaneous names, resulting in a total of $|\Sigma| = 9$ different labels. In the setup followed in Altun et al. (2003), the joint feature map $\Psi(\mathbf{x}, \mathbf{y})$ is the histogram of state transition plus a set of features describing the emissions. An adapted version of the Viterbi algorithm is used to solve the *argmax* in line 6. For both perceptron and SVM a second degree polynomial kernel was used.

The results given in Table 2 for the zero-one loss, compare the generative HMM with Conditional Random Fields (CRF) (Lafferty et al., 2001), Collins' per-

[2] http://www.recommind.com

Table 2. Results of various algorithms on the Named Entity Recognition task (Altun et al., 2003).

Method	HMM	CRF	Perceptron	SVM
Error	9.36	5.17	5.94	5.08

Table 3. Results for various SVM formulations on the Named Entity Recognition task ($\epsilon = 0.01$, $C = 1$).

Method	Train Err	Test Err	Const	Avg Loss
SVM_2	0.2±0.1	5.1±0.6	2824±106	1.02±0.01
$SVM_2^{\triangle s}$	0.4±0.4	5.1±0.8	2626±225	1.10±0.08
$SVM_2^{\triangle m}$	0.3±0.2	5.1±0.7	2628±119	1.17±0.12

ceptron and the SVM algorithm. All discriminative learning methods substantially outperform the standard HMM. In addition, the SVM performs slightly better than the perceptron and CRFs, demonstrating the benefit of a large-margin approach. Table 3 shows that all SVM formulations perform comparably, probably due to the fact the vast majority of the support label sequences end up having Hamming distance 1 to the correct label sequence (notice that for loss equal to 1 all SVM formulations are equivalent).

5.4. Sequence Alignment

Next we show how to apply the proposed algorithm to the problem of learning how to align sequences $\mathbf{x} \in \mathcal{X} = \Sigma^*$. For a given pair of sequences $\mathbf{x}$ and $\mathbf{z}$, alignment methods like the Smith-Waterman algorithm select the sequence of operations (e.g. insertion, substitution) $\hat{\mathbf{a}}(\mathbf{x},\mathbf{z}) = \operatorname{argmax}_{\mathbf{a}\in\mathcal{A}} \langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{z},\mathbf{a})\rangle$ that transforms $\mathbf{x}$ into $\mathbf{y}$ and that maximizes a linear objective function derived from the (negative) operation costs $\mathbf{w}$. $\Psi(\mathbf{x},\mathbf{z},\mathbf{a})$ is the histogram of alignment operations. We use the value of $\langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{z},\hat{\mathbf{a}}(\mathbf{x},\mathbf{z}))\rangle$ as a measure of similarity.

In order to learn the cost vector $\mathbf{w}$ we use training data of the following type. For each native sequence $\mathbf{x}_i$ there is a most similar homologue sequence $\mathbf{z}_i$ along with what is believed to be the (close to) optimal alignment $\mathbf{a}_i$. In addition we are given a set of decoy sequences $\mathbf{z}_i^t$, $t = 1, \ldots, k$ with unknown alignments. The goal is to find a cost vector $\mathbf{w}$ so that homologue sequences are close to the native sequence, and so that decoy sequences are further away. With $\mathcal{Y}_i = \{z_i, z_i^1, ..., z_i^k\}$ as the output space for the i-th example, we seek a $\mathbf{w}$ so that $\langle \mathbf{w}, \Psi(\mathbf{x}_i,\mathbf{z}_i,\mathbf{a}_i)\rangle$ exceeds $\langle \mathbf{w}, \Psi(\mathbf{x}_i,\mathbf{z}_i^t,\mathbf{a})\rangle$ for all t and $\mathbf{a}$. This implies a zero-one loss and hypotheses of the form $f(\mathbf{x}_i,\mathbf{w}) = \operatorname{argmax}_{\mathbf{y}\in\mathcal{Y}_i} \max_a \langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{z},\mathbf{a})\rangle$. We use the Smith-Waterman algorithm to implement the $\max_a$.

Table 4 shows the test error rates (i.e. fraction of times the homolog is not selected) on the synthetic dataset described in Joachims (2003). The results are averaged over 10 train/test samples. The model contains 400 parameters in the substitution matrix Π and a cost δ for "insert/delete". We train this model using the SVM_2 and compare against a generative sequence alignment model, where the substitution matrix is computed as $\Pi_{ij} = \log\left(\frac{P(x_i,z_j)}{P(x_i)P(z_j)}\right)$ using Laplace estimates. For the generative model, we report the results for $\delta = -0.2$, which performs best on the test set. Despite this unfair advantage, the SVM performs better for low training set sizes. For larger training sets, both methods perform similarly, with a small preference for the generative model. However, an advantage of the SVM model is that it is straightforward to train gap penalties. As predicted by Theorem 1, the number of constraints $|S|$ is low. It appears to grows sub-linearly with the number of examples.

Table 4. Error rates and number of constraints $|S|$ depending on the number of training examples ($\epsilon = 0.1$, $C = 0.01$).

	Train Error		Test Error		
n	GenMod	SVM_2	GenMod	SVM_2	Const
1	20.0±13.3	0.0±0.0	74.3±2.7	47.0±4.6	7.8±0.3
2	20.0±8.2	0.0±0.0	54.5±3.3	34.3±4.3	13.9±0.8
4	10.0±5.5	2.0±2.0	28.0±2.3	14.4±1.4	31.9±0.9
10	2.0±1.3	0.0±0.0	10.2±0.7	7.1±1.6	58.9±1.2
20	2.5±0.8	1.0±0.7	3.4±0.7	5.2±0.5	95.2±2.3
40	2.0±1.0	1.0±0.4	2.3±0.5	3.0±0.3	157.2±2.4
80	2.8±0.5	2.0±0.5	1.9±0.4	2.8±0.6	252.7±2.1

5.5. Natural Language Parsing

We test the feasibility of our approach for learning a weighted context-free grammar (see Figure 1) on a subset of the Penn Treebank Wall Street Journal corpus. We consider the 4098 sentences of length at most 10 from sections F2-21 as the training set, and the 163 sentences of length at most 10 from F22 as the test set. Following the setup in Johnson (1999), we start based on the part-of-speech tags and learn a weighted grammar consisting of all rules that occur in the training data. To solve the *argmax* in line 6 of the algorithm, we use a modified version of the CKY parser of Mark Johnson[3] and incorporated it into SVM^{light}.

The results are given in Table 5. They show accuracy and micro-averaged F_1 for the training and the test set. The first line shows the performance for generative PCFG model using the maximum likelihood estimate (MLE) as computed by Johnson's implementation. The second line show the SVM_2 with zero-one loss, while the following lines give the results for the F_1-loss $\triangle(\mathbf{y}_i,\mathbf{y}) = (1 - F_1(\mathbf{y}_i,\mathbf{y}))$ using $SVM_2^{\triangle s}$ and

[3]At `http://www.cog.brown.edu/~mj/Software.htm`

Table 5. Results for learning a weighted context-free grammar on the Penn Treebank. CPU time measured in hours.

	Train		Test		Training Efficiency	
Method	Acc	F_1	Acc	F_1	Const	CPU(%QP)
PCFG	61.4	90.4	55.2	86.0	N/A	0
SVM_2	66.3	92.0	58.9	86.2	7494	1.2 (81.6%)
$\text{SVM}_2^{\triangle s}$	62.2	92.1	58.9	88.5	8043	3.4 (10.5%)
$\text{SVM}_2^{\triangle m}$	63.5	92.3	58.3	88.4	7117	3.5 (18.0%)

$\text{SVM}_2^{\triangle m}$. All results are for $C = 1$ and $\epsilon = 0.01$. All values of C between 10^{-1} to 10^2 gave comparable results. While the zero-one loss achieves better accuracy (i.e. predicting the complete tree correctly), the F_1-score is only marginally better. Using the F_1-loss gives substantially better F_1-scores, outperforming the MLE substantially. The difference is significant according to a McNemar test on the F_1-scores. We conjecture that we can achieve further gains by incorporating more complex features into the grammar, which would be impossible or at best awkward to use in a generative PCFG model. Note that our approach can handle arbitrary models (e.g. with kernels and overlapping features) for which the *argmax* in line 6 can be computed.

In terms of training time, Table 5 shows that the total number of constraints added to the working set is small. It is roughly twice the number of training examples in all cases. While the training is faster for the zero-one loss, the time for solving the QPs remains roughly comparable. The re-scaling formulations lose time mostly on the *argmax* in line 6. This might be sped up, since we were using a rather naive algorithm in the experiments.

6. Conclusions

We formulated a Support Vector Method for supervised learning with structured and interdependent outputs. It is based on a joint feature map over input/output pairs, which covers a large class of interesting models including weighted context-free grammars, hidden Markov models, and sequence alignment. Furthermore, the approach is very flexible in its ability to handle application specific loss functions. To solve the resulting optimization problems, we proposed a simple and general algorithm for which we prove convergence bounds. Our empirical results verify that the algorithm is indeed tractable. Furthermore, we show that the generalization accuracy of our method is at least comparable or often exceeds conventional approaches for a wide range of problems. A promising property of our method is that it can be used to train complex models, which would be difficult to handle in a generative setting.

Acknowledgments

The authors would like to thank Lijuan Cai for conducting the experiments on classification with taxonomies. This work was supported by the Paris Kanellakis Dissertation Fellowship, NSF-ITR Grant IIS-0312401, and NSF CAREER Award 0237381.

References

Altun, Y., Tsochantaridis, I., & Hofmann, T. (2003). Hidden markov support vector machines. *ICML*.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *EMNLP*.

Collins, M. (2004). *Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods*.

Crammer, K., & Singer, Y. (2001). On the algorithmic implementation of multi-class kernel-based vector machines. *Machine Learning Research*, *2*, 265–292.

Hofmann, T., Tsochantaridis, I., & Altun, Y. (2002). Learning over structured output spaces via joint kernel functions. *Sixth Kernel Workshop*.

Joachims, T. (2003). *Learning to align sequences: A maximum-margin approach* (Technical Report). Cornell University.

Johnson, M. (1999). PCFG models of linguistic tree representations. *Computational Linguistics*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*.

Manning, C. D., & Schuetze, H. (1999). *Foundations of statistical natural language processing*. MIT Press.

Taskar, B., Guestrin, C., & Koller, D. (2004). Max-margin markov networks. *NIPS 16*.

Vapnik, V. (1998). *Statistical learning theory*. Wiley and Sons Inc.

Weston, J., Chapelle, O., Elisseeff, A., Schölkopf, B., & Vapnik, V. (2003). Kernel dependency estimation. *NIPS 15*.

Weston, J., & Watkins, C. (1998). *Multi-class support vector machines* (Technical Report CSD-TR-98-04). Department of Computer Science, Royal Holloway, University of London.