1 SVM Learning for Interdependent and Structured Output Spaces

Yasemin Altun
Toyota Technological Institute at Chicago, Chicago IL 60637 USA
altun@tti-c.org

Thomas Hofmann
Google, Zurich, Austria
thofmann@google.com

Ioannis Tsochantaridis
Google, Mountain View CA USA
ioannis@google.com

1.1 Introduction

Supervised learning, one of the most important areas of machine learning, is the general problem of learning a function that predicts the best value for a response variable y for an observation x by making use of a sample of input-output pairs. Traditionally, in classification, the values that y can take are *simple*, in the sense that they can be characterized by an arbitrary identifier. However, in many real-world applications the outputs are often complex, in that either there are dependencies between classes (eg. taxonomies used for example in document classification), or the classes are objects that have some internal structure such that they describe a configuration over inter-dependent components (eg. sequences, parse trees). For such problems, which are commonly called *structured output prediction* problems, standard multiclass approaches render ineffective, since the size of the output space is very large (eg. the set of label sequences scale exponentially with the length of the input sequence). More importantly, it is crucial to capture the common properties that are shared by the set of classes in order to generalize across classes as well as to generalize across input patterns.

In this paper, we approach the structured output prediction problems by generalizing a multiclass Support Vector Machine formulation by Crammer and Singer (2001) to the broad problem of learning for interdependent and structured outputs.

To that extent, we specify discriminant functions that exploit the dependencies and structure of outputs. This framework enables generalization across classes and prediction of classes that may not have been observed in the training set. We provide the details of this framework for three important special cases, namely hierarchical classification, label sequence learning and weighted context-free grammar learning.

The standard 0-1 cost function is not adequate to capture the differences between classes in interdependent and structured output spaces. More sophisticated cost functions such as Hamming loss and F_1 score are common in practice, for example for sequence and parse trees. We generalize the separation margin notion for structured outputs and device max-margin formulations that directly incorporate the cost functions that the classifier is evaluated on. These formulations result in a potentially prohibitive, more specifically exponential, number of constraints. However, exploiting the sparsity and the (de-)coupling of these constraints, we present a cutting-plane algorithm that is guaranteed to satisfy the exponential number of constraints upto an ϵ-precision without evaluating them explicitly.

We empirically evaluate our approach in document classification as an instance of hierarchical classification, named entity recognition as an instance of label-sequence learning and natural language parsing as an instance of learning weighted context-free grammars. Experimental results show that our framework is advantageous over the more standard approaches in terms of the cost function on which the classifiers are evaluated.

1.2 A Framework for Structured/Interdependent Output Learning

We are interested in the task of inferring a *complex* label $\mathbf{y} \in \mathcal{Y}$ for a (possibly structured) observation $\mathbf{x} \in \mathcal{X}$. Given a sample of input-output pairs $S = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)\}$ generated from an unknown distribution P, the goal is to learn a mapping $f : \mathcal{X} \to \mathcal{Y}$ between input spaces $\mathcal{X}$ and interdependent/structured output spaces $\mathcal{Y}$. Hence, it is a generalization of the supervised classification problem where values of the random variables are predicted not only with respect to observations but also with respect to the values of other related random variables.

The prediction function f is evaluated according to a cost function $\triangle : \mathcal{Y} \times \mathcal{Y} \to \Re$, which measures the similarity between two labels. We focus on cost functions where $\triangle(\mathbf{y}, \mathbf{y}) = 0$ and $\triangle(\mathbf{y}, \mathbf{y}') \geq 0, \mathbf{y} \neq \mathbf{y}'$. For example, 0-1 loss, Hamming loss and $1-F_1$ loss are the canonical cost functions of multiclass classification, label sequence learning and parsing respectively. The goal of learning a function that minimizes the cost over the unknown P is commonly approximated by learning a function that minimizes the empirical cost

$$\mathcal{R}_{\mathcal{S}}^{\triangle}(f) = \frac{1}{n} \sum_{i=1}^{n} \triangle(\mathbf{y}_i, f(\mathbf{x}_i)). \tag{1.1}$$

In general, minimizing this cost on the sample is NP complete. Following the usually practice in machine learning, we investigate optimizing surrogate functions of the empirical cost.

In structured and interdependent output prediction, two factors make it essential to generalize across sets of labels as well as to generalize across input patterns. First, in most cases, the very large size of the label sets renders any learning that is independent over class labels intractable. More importantly, capturing common properties shared by sets of labels enable us to use data points across classes, and even generalize to class labels that are not observed in the sample but likely to occur as the label of a new observation. Therefore, the standard approach of multiclass classification, i.e. learning a function $F_y : \mathcal{X} \to \Re$ for each class independently and inferring the label of an observation by maximizing $F_y(\mathbf{x})$ over all labels, is not appropriate for this setting. We define a *discriminant function* $F : \mathcal{X} \times \mathcal{Y} \to \Re$ over the joint input-output space where $F(\mathbf{x}, \mathbf{y})$ can be interpreted as measuring the compatibility of $\mathbf{x}$ and $\mathbf{y}$. Each such function F induces a mapping f,

$$f(\mathbf{x}) = \operatorname*{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}), \tag{1.2}$$

where $\mathbf{w}$ denotes a parameter vector and ties are broken arbitrarily. We restrict the space of F to linear functions over some feature representation Ψ, which is defined on the joint input-output space

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

Ψ is chosen with respect to the dependency structure of $\mathbf{y}$ and $\mathbf{x}$ and commonalities within $\mathbf{y}$'s in order to enable generalization across labels. Before we present several interesting special cases, we need some definitions. We define the canonical (binary) representation of outputs $y \in \mathcal{Y} = \{1, \ldots, k\}$ by unit vectors

$$\Lambda^c(y) \equiv (\delta(y,1), \delta(y,2), \ldots, \delta(y,k))' \in \{0,1\}^k.$$

so that $\langle \Lambda^c(y), \Lambda^c(y') \rangle = \delta(y, y')$. Let the tensor product $\otimes$ and the concatenation $\odot$ be defined as

$$\begin{aligned} \otimes :& \Re^d \times \Re^k \to \Re^{dk}, \quad [\mathbf{a} \otimes \mathbf{b}]_{i+(j-1)d} \equiv [\mathbf{a}]_i [\mathbf{b}]_j, \\ \odot :& \Re^d \times \Re^k \to \Re^{d+k}, \quad \mathbf{a} \odot \mathbf{b} \equiv (\mathbf{a}', \mathbf{b}')'. \end{aligned}$$

The following proposition states that feature representations derived from $\odot$ and $\otimes$ operations, such as all the joint feature maps Ψ defined in this chapter, are induced by valid kernels.

Proposition 1 *Let Φ and $\bar{\Phi}$ be feature representations induced by kernels $k, \bar{k}$ over $X \times X, \bar{X} \times \bar{X}$ respectively (i. e. $k(a, \bar{a}) = \langle \Phi(a), \Phi(\bar{a}) \rangle$). Then, for any $a, \bar{a} \in X, b, \bar{b} \in \bar{X}$, $\Phi \otimes \bar{\Phi}$ and $\Phi \odot \bar{\Phi}$ are induced by kernels $k_\otimes, k_\odot$ where*

$$k_\otimes((a,b),(\bar{a},\bar{b})) = k(a,\bar{a})\bar{k}(b,\bar{b}), \tag{1.3}$$

$$k_\odot((a,b),(\bar{a},\bar{b})) = k(a,\bar{a}) + \bar{k}(b,\bar{b}). \tag{1.4}$$

Proof *The claims follow from the definitions of $\odot$ and $\otimes$ operations and from the fact that sums and pointwise products of two kernels are also kernels (Schölkopf and Smola, 2002).* ■

For the rest of this section, we assume the existence of an arbitrary feature representation of the inputs, $\Phi(\mathbf{x}) \in \Re^d$, and a kernel function k that induces Φ.

It is easy to see that multi-class classification is a special case of our framework where $\mathcal{Y} = \{1, \ldots, k\}$. Let the weight vector $\mathbf{w}$ be a concatenation of all $\mathbf{w}_r$ with $\mathbf{w}_r$ being a weight vector associated with the r-th class, $\mathbf{w} = \mathbf{w}_1 \odot \cdots \odot \mathbf{w}_k$. Defining the joint feature map is given by $\Psi(\mathbf{x}, y) \equiv \Phi(\mathbf{x}) \otimes \Lambda^c(y)$, results in the familiar multiclass discriminant function $F(\mathbf{x}, y; \mathbf{w}) = \langle \mathbf{w}_y, \Phi(\mathbf{x}) \rangle$.

Let us now examine more interesting special cases.

1.2.1 Hierarchical Classification

In many applications, such as document classification and word sense disambiguation, taxonomies and hierarchies are natural ways to organize classes of objects. These problems are instances of interdependent output spaces where the feature representation is defined as follows: Let a taxonomy be a set of elements $\mathcal{Z} \supseteq \mathcal{Y}$ equipped with a partial order $\prec$, which can be by a tree or a lattice, and let $\beta_{(y,z)} \in \Re$ be a measure of similarity with respect to the partial order $\prec$. We generalize the canonical representation of outputs to $\Lambda(y) \in \Re^p$, such that for all $z \in \mathcal{Z}$

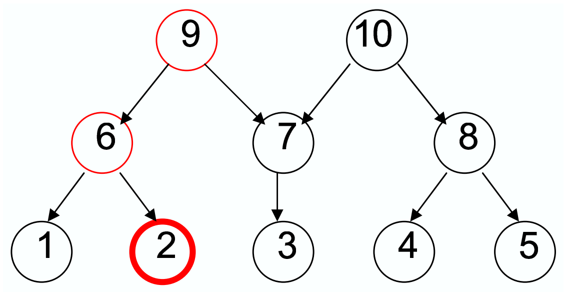
$$\lambda_z(y) = \begin{cases} \beta_{(y,z)} & \text{if } y \prec z \text{ or } y = z \\ 0 & \text{otherwise} \end{cases}$$

Then, defining the joint input-output feature map via the tensor product,

$$\Psi(\mathbf{x}, y) = \Phi(\mathbf{x}) \otimes \Lambda(y).$$

effectively introduces a weight vector $\mathbf{w}_z$ for all $z \in \mathcal{Z}$, i.e. for every node in the hierarchy. A simple derivation shows that the weight vector of a class is a linear combination of its processors' weights and the discriminant is given by

$$F(\mathbf{x}, y; \mathbf{w}) = \sum_{z: y \prec z \text{ or } z = y} \beta(y,z) \langle \mathbf{w}_z, \Phi(\mathbf{x}) \rangle .$$



$$\Lambda(2) = \begin{pmatrix} 0\\1\\0\\0\\0\\1\\0\\0\\1\\0 \end{pmatrix} \qquad \Psi(\mathbf{x}, 2) = \begin{pmatrix} 0\\\mathbf{x}\\0\\0\\0\\\mathbf{x}\\0\\0\\\mathbf{x}\\0 \end{pmatrix}$$

$$\langle \mathbf{w}, \Psi(\mathbf{x}, 2)\rangle = \langle w_2, \mathbf{x}\rangle + \langle w_6, \mathbf{x}\rangle + \langle w_9, \mathbf{x}\rangle$$

Figure 1.1 Classification with taxonomies.

Thus, the features λ_z are shared by all successor classes of z and the joint feature representation enables *generalization across classes*. Figure 1.1 shows an example of the joint feature map Ψ of the second class for a given hierarchy.

It follows immediately from (1.3) of Proposition 1 that the inner product of the joint feature map decomposes into kernels over input and output spaces

$$\langle \Psi(\mathbf{x}, y), \Psi(\mathbf{x}', y')\rangle = \langle \Lambda(y), \Lambda(y')\rangle\, k(\mathbf{x}, \mathbf{x}').$$

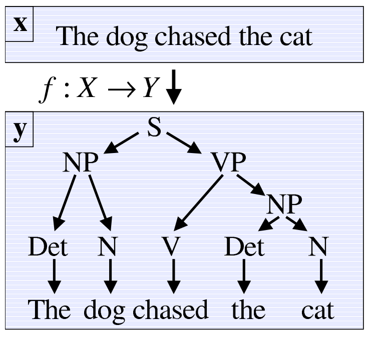
1.2.2 Label Sequence Learning

Label sequence learning is the task of predicting a sequence of labels $\mathbf{y} = (y^1, \ldots, y^l)$ for a given observation sequence $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^l)$. Applications of this problem are ubiquitous in many domains such as computational biology, information retrieval, natural language processing and speech recognition. We denote by $l_\mathbf{x}$ the length of an observation sequence, by Σ the set of possible labels for each individual variable y^t, and by $\mathcal{Y}(\mathbf{x})$ the set of label sequences for $\mathbf{x}$. Then, $\mathcal{Y}(\mathbf{x}) = \Sigma^{l_\mathbf{x}}$.

In order to encode the dependencies of the observation-label sequences which are commonly realized as a Markov chain, we define Ψ to include interactions between input features and labels ($\Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t)$), as well as interactions between neighboring label variables ($\Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})$) for every position t. Then, using the stationary property, our joint feature map is a sum over all positions

$$\Psi(\mathbf{x}, \mathbf{y}) = \left[\sum_{t=1}^{l_\mathbf{x}} \Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t)\right] \odot \left[\eta \sum_{t=1}^{l_\mathbf{x}-1} \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})\right], \qquad (1.5)$$

where $\eta \geq 0$ is a scalar balancing the two types of contributions. Clearly, this representation can be generalized by including higher order inter-dependencies of labels (e. g. $\Lambda^c(y^t) \otimes \Lambda^c(y^{t+1}) \otimes \Lambda^c(y^{t+2})$), by including input features from a window centered at the current position (e. g. replacing $\Phi(\mathbf{x}^t)$ with $\Phi(\mathbf{x}^{t-r}, \ldots, \mathbf{x}^t, \ldots, \mathbf{x}^{t+r})$) or by combining higher order output features with input features (e. g. $\sum_t \Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})$). The important constraint on designing the feature map is the



$$\Psi(\mathbf{x},\mathbf{y}) = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \\ \vdots \\ 0 \\ 2 \\ 1 \\ 1 \\ 1 \end{pmatrix} \begin{array}{c} S \to NP\,VP \\ S \to NP \\ NP \to Det\ N \\ VP \to V\ NP \\ \\ Det \to dog \\ Det \to the \\ N \to dog \\ V \to chased \\ N \to cat \end{array}$$

Figure 1.2 Natural language parsing.

efficient computation of the discriminant function, which in the case of (1.5) is given by

$$F(\mathbf{x},\mathbf{y};\mathbf{w}) = \left\langle \mathbf{w}_{ol}, \sum_{t=1}^{l_{\mathbf{x}}} \Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t) \right\rangle + \eta \left\langle \mathbf{w}_{ll}, \sum_{t=1}^{l_{\mathbf{x}}-1} \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1}) \right\rangle, \tag{1.6}$$

where $\mathbf{w} = \mathbf{w}_{ol} \odot \mathbf{w}_{ll}$ is the concatenation of weights of the two dependency types.

As indicated in Proposition 1, the inner product of the joint feature map decomposes into kernels over input and output spaces

$$\langle \Psi(\mathbf{x},\mathbf{y}), \Psi(\mathbf{x}',\mathbf{y}') \rangle = \sum_{t=1}^{l_{\mathbf{x}}} \sum_{s=1}^{l_{\mathbf{x}'}} \delta(y^t, \bar{y}^s) k(\mathbf{x}^t, \bar{\mathbf{x}}^s) + \eta^2 \sum_{t=1}^{l_{\mathbf{x}}-1} \sum_{s=1}^{l_{\mathbf{x}'}-1} \delta(y^t, \bar{y}^s) \delta(y^{t+1}, y^{s+1}). \tag{1.7}$$

where we used the equality $\langle \Lambda^c(\sigma), \Lambda^c(\bar{\sigma}) \rangle = \delta(\sigma, \bar{\sigma})$.

1.2.3 Weighted Context-Free Grammars

Parsing is the task of predicting a labeled tree $\mathbf{y}$ that is a particular configuration of grammar rules generating a given sequence $\mathbf{x} = (x^1, ..., x^l)$. Let us consider a context-free grammar in Chomsky Normal Form. The rules of this grammar are of the form $\sigma \to \sigma'\sigma''$, or $\sigma \to x$, where $\sigma, \sigma', \sigma'' \in \Sigma$ are non-terminals, and $x \in T$ are terminals. Similar to the sequence case, we define the joint feature map $\Psi(x, y)$ to contain features representing inter-dependencies between labels of the nodes of the tree (e.g. $\psi_{\sigma \to \sigma'\sigma''}$ via $\Lambda^c(y^{rs}) \otimes \Lambda^c(y^{rt}) \otimes \Lambda^c(y^{(t+1)s})$) and features representing the dependence of labels to observations (e.g. $\psi_{\sigma \to \tau}$ via $\Phi^c(x^t) \otimes \Lambda^c(y^t)$). Here y^{rs} denotes the label of the root of a subtree spanning from x^r to x^s. This definition leads to equations similar to (1.5), (1.6) and (1.7). Extensions to this representation is possible, for example by defining higher order features that can be induced using kernel functions over sub-trees (Collins and Duffy, 2002).

1.3 A Maximum Margin Formulation

We propose a maximum margin approach for the problem of learning in structured and interdependent output spaces. We first generalize the multiclass separation margin of (Crammer and Singer, 2001), where the margin of an instance $(\mathbf{x}, \mathbf{y})$ with respect to $\mathbf{w}$ is given by

$$\gamma(\mathbf{x}, \mathbf{y}; \mathbf{w}) = F(\mathbf{x}, \mathbf{y}; \mathbf{w}) - \max_{\mathbf{y}' \in \mathcal{Y} \setminus \mathbf{y}} F(\mathbf{x}, \mathbf{y}'; \mathbf{w}).$$

Then, the maximum margin problem can be defined as finding the weight vector $\mathbf{w}$ that maximizes the minimum margin of the sample, $\min_i \gamma(\mathbf{x}^i, \mathbf{y}^i)$. If the data is separable ($\gamma > 0$), there exists multiple solutions to this problem, since the margin can be made arbitrarily large by scaling $\mathbf{w}$. This can be resolved by fixing either the norm of $\mathbf{w}$ (eg.$\|\mathbf{w}\| = 1$) or the margin (eg. $\min_i \gamma(\mathbf{x}^i, \mathbf{y}^i) \geq 1$). Following the latter, we have a constraint optimization problem

$$\begin{aligned} \text{SVM}_0: \quad & \min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 \\ & \text{s.t. } F(\mathbf{x}_i, \mathbf{y}_i; \mathbf{w}) - \max_{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i} F(\mathbf{x}_i, \mathbf{y}; \mathbf{w}) \geq 1, \quad \forall i. \end{aligned}$$

In order to accommodate for margin violations, we generalize this formulation by introducing linear penalties that are scaled according to the cost incurred by misclassification. 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 margin violation, given by $(1 - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle)$ where $\delta\Psi_i(\mathbf{y}) = \Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y}))$, by the cost

$$\text{SVM}_1^{\triangle s}: \quad \min_{\mathbf{w}, \boldsymbol{\xi}} \ \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + \frac{C}{n}\sum_{i=1}^{n} \xi_i \tag{1.9a}$$

$$\text{s.t. } \max_{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i} [\triangle(\mathbf{y}_i, \mathbf{y})(1 - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle)] \geq 1 - \xi_i \quad \forall i \tag{1.9b}$$

Here $C > 0$ is a constant controlling the trade of between the loss and the regularizer. Note that $\text{SVM}_1^{\triangle s}$ can also be stated without constraints via a hinge loss function

$$\text{SVM}_1^{\triangle s}: \quad \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \max_{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i} [\triangle(\mathbf{y}_i, \mathbf{y})(1 - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle)_+], \tag{1.10}$$

where $(a)_+ = \max(0, a)$ denotes the hinge loss. It is easy to show that $\text{SVM}_1^{\triangle s}$ is a surrogate of the empirical cost (1.1).

Proposition 2 *Let $\mathbf{w}^*$ be the optimal solution to $SVM_1^{\triangle s}$. Then, the empirical risk $\mathcal{R}_{\mathcal{S}}^{\triangle}(\mathbf{w}^*)$ is upper bounded by $\frac{1}{n}\sum_{i=1}^{n} \max_{\mathbf{y} \neq \mathbf{y}^i} [\triangle(\mathbf{y}_i, \mathbf{y})(1 - \langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y}) \rangle)_+]$.*

Proof *If* $f(\mathbf{x}_i;\mathbf{w}^*) = \mathbf{y}_i$ *then* $\triangle(\mathbf{y}_i, f(\mathbf{x}_i;\mathbf{w})) = 0$ *and the bound holds trivially. If* $\hat{\mathbf{y}} \equiv f(\mathbf{x}_i;\mathbf{w}^*) \neq \mathbf{y}_i$*, then* $\max_{\mathbf{y}\neq\mathbf{y}^i} [\triangle(\mathbf{y}_i,\mathbf{y})(1 - \langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y})\rangle)_+] \geq \triangle(\mathbf{y}_i,\mathbf{y})$*. because* $\langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y})\rangle < 0$*. Since the bound holds for every training instance, it also holds for the average.* ■

A similar optimization problem can be given by the squared hinge loss, or equivalently by the squared penalties on the slack variable This loss, which we denote by $\text{SVM}_2^{\triangle s}$, is also an upper bound on the empirical cost $\mathcal{R}_{\mathcal{S}}^{\triangle}(\mathbf{w})$.

1.3.1 Optimizing the primal problem

By specifying the class of discriminant functions F to be the Reproducing Kernel Hilbert Space (RKHS) associated with the kernel k, where $k((\mathbf{x},\mathbf{y}),(\bar{\mathbf{x}},\bar{\mathbf{y}})) = \langle \Psi(\mathbf{x},\mathbf{y}), \Psi(\bar{\mathbf{x}},\bar{\mathbf{y}})\rangle$, it is easy to show that $\text{SVM}_1^{\triangle s}$ (1.10) is an instance of a more general optimization problem given by

$$F^* = \operatorname*{argmin}_{F\in\mathcal{H}} \sum_{i=1}^{n} \mathcal{L}(x^i, y^i, F) + \lambda \|F\|_{\mathcal{H}}^2,$$

where $\mathcal{L}$ is a convex loss function. The well-known Representer Theorem(Kimeldorf and Wahba, 1971) states that the solution F^* lies in the span of data points, in this case $(\mathbf{x}_i, \mathbf{y})$ for all $\mathbf{x}_i \in S$ and $\mathbf{y} \in \mathcal{Y}(\mathbf{x})$. Under fairly mild conditions, one can show that this space can be further reduced to the span of the substructures of the data points, via a straightforward variant of the Representer Theorem which was also presented in (Lafferty et al., 2004; Altun et al., 2005). Let $\mathcal{C}(\mathbf{x},\mathbf{y})$ be the set of assignments of all subcomponents of $(\mathbf{x},\mathbf{y})$. For example, if $(\mathbf{x},\mathbf{y})$ corresponds to a Markov Random Field, $\mathcal{C}(\mathbf{x},\mathbf{y})$ is given by the clique assignments. Furthermore, let $\mathcal{C}(S) = \cup_{\mathbf{x}\in S, \mathbf{y}\in\mathcal{Y}(\mathbf{x})} \mathcal{C}(\mathbf{x},\mathbf{y})$. If the discriminant $F(\mathbf{x},\mathbf{y})$ decomposes into a function h defined over the subcomponents of $(\mathbf{x},\mathbf{y})$ and the loss $\mathcal{L}$ is local in the sense that $\mathcal{L}(\mathbf{x},\mathbf{y},F)$ is determined by the values of h over the set $\mathcal{C}(\mathbf{x})$, F^* can be represented over $\mathcal{C}(S)$.

Theorem 3 *For any local loss function* $\mathcal{L}$ *and any sample* S*, there exist some weights* $\alpha_c, \forall c \in \mathcal{C}(S)$ *such that* F^* *admits a representation of the form*

$$F^*(\mathbf{x},\mathbf{y};\alpha) = \sum_{c\in\mathcal{C}(\mathbf{x},\mathbf{y})} h^*(c;\alpha),$$
$$h^*(c;\alpha) = \sum_{c'\in\mathcal{C}(S)} \alpha_c \tilde{k}(c,c).$$

where the kernel function $\tilde{k}$ *is defined so that* $h(c) = \left\langle h, \tilde{k}(c,.)\right\rangle$*.*

Note that since all joint feature maps $\Psi(\mathbf{x},\mathbf{y})$ considered in Section 1.2 decompose into subcomponents of $(\mathbf{x},\mathbf{y})$, the corresponding F and the margin losses,

$\mathrm{SVM}_1^{\triangle s}$ and $\mathrm{SVM}_2^{\triangle s}$, satisfy the conditions in Theorem 3. Then, we can reformalize $\mathrm{SVM}_1^{\triangle s}$ as an optimization problem over α

$$\min_{\alpha,\xi_i(\alpha)} \frac{1}{2} \sum_{c,c' \in \mathcal{C}(S)} \alpha_c \alpha_{c'} k(c,c') + \frac{C}{n} \sum_{i=1}^{n} \xi_i(\alpha)$$
$$\text{s.t.} \max_{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i} [\triangle(\mathbf{y}_i,\mathbf{y})(1 + F(\mathbf{x}_i,\mathbf{y};\alpha) - F(\mathbf{x}_i,\mathbf{y}_i;\alpha)] \geq 1 - \xi_i(\alpha) \quad \forall i. \tag{1.11}$$

This gives a convex program over the vectors indexed by $\mathcal{C}(S)$, which scales polynomially in terms of the size of the output variables. Every one of the nonlinear inequalities (1.11) implicitly represent exponentially many linear constraints given by

$$\forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \quad \triangle(\mathbf{y}_i,\mathbf{y})(1 + F(\mathbf{x}_i,\mathbf{y};\alpha) - F(\mathbf{x}_i,\mathbf{y}_i;\alpha)) \geq 1 - \xi_i(\alpha). \tag{1.12}$$

If one can perform the max operation in (1.11) via a polynomial time dynamic programming (DP) algorithm, then there exists a polynomial number of constraints that satisfy (1.12). Each such constraint involve a cost variable C, which are coupled within instances $\mathbf{x}_i$ and are upper bounded by ξ_i. The resulting optimization problem, which is closely related to the factored primal of Taskar et al. (2003), is a polynomial sized QP with polynomial number of highly coupled constraints. Unfortunately, this coupling prohibits the use of decomposition methods such as Sequential Minimal Optimization (Platt, 1998) and may render the optimization intractable for large dataset. In the next section, we present a method for optimizing the dual problem which benefits the decomposition methods.

1.3.2 Dual problem

Using the standard Lagrangian techniques, i .e .introducing a Lagrange parameter $\alpha_{(i\mathbf{y})}$ enforcing the margin constraint for label $\mathbf{y} \neq \mathbf{y}_i$ and input $\mathbf{x}_i$, writing out the Lagrangian, differentiating it with respect to the primal parameters $\mathbf{w}$ and ξ, and substituting the optimality equations of the primals into the Lagrangian results, in a dual quadratic program (QP). Let $j((\mathbf{x}_i,\mathbf{y}),(\mathbf{x}_j,\bar{\mathbf{y}})) = \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \rangle$ denote the kernel function [1]. Then, the QP of $\mathrm{SVM}_1^{\triangle s}$ and $\mathrm{SVM}_2^{\triangle s}$ is given in the following proposition.

Proposition 4 *The dual problem to $SVM_1^{\triangle s}$ and $SVM_2^{\triangle s}$ is given by the program*

$$\boldsymbol{\alpha}^* = \operatorname*{argmax}_{\boldsymbol{\alpha}} -\frac{1}{2} \sum_{i,j} \sum_{\mathbf{y} \neq \mathbf{y}_i, \bar{\mathbf{y}} \neq \mathbf{y}_j} \alpha_{(i\mathbf{y})} \alpha_{(j\bar{\mathbf{y}})} j((\mathbf{x}_i,\mathbf{y}),(\mathbf{x}_j,\bar{\mathbf{y}})) + \sum_i \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{(i\mathbf{y})}, \tag{1.13}$$
$$s.\ t.\ \boldsymbol{\alpha} \geq 0,$$

1. Note that j can be computed from the inner products involving values of Ψ due to the linearity of the inner product, and is a valid kernel.

where $SVM_1^{\triangle s}$ has additional box constraints

$$\sum_{\mathbf{y}\neq\mathbf{y}_i} \frac{\alpha_{(i\mathbf{y})}}{\triangle(\mathbf{y}_i,\mathbf{y})} \leq \frac{C}{n}, \quad \forall i = 1,\ldots,n$$

and $SVM_2^{\triangle s}$ has a modified kernel function

$$j((\mathbf{x}_i,\mathbf{y}),(\mathbf{x}_j,\bar{\mathbf{y}})) = \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}})\rangle + \frac{n\delta_{ij}}{C\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}\sqrt{\triangle(\mathbf{y}_j,\bar{\mathbf{y}})}} \tag{1.14}$$

The optimality equation of $\mathbf{w}$ is given by

$$\mathbf{w}^* = \sum_j \sum_{\mathbf{y}} \alpha^t_{(j\mathbf{y})} \delta\Psi_j(\mathbf{y}) \tag{1.15}$$

1.4 Cutting Plane Algorithm

The main computational challenge in optimizing (1.13) is posed by the extremely large number of variables $(n|\mathcal{Y}| - n)$. If $\mathcal{Y}$ is a product space, its cardinality grows exponential in the size of $\mathbf{y}$ (for instance in sequences of length l, $|\mathcal{Y}| = |\Sigma|^l$), rendering the optimization of (1.13) by standard quadratic programming solvers intractable. The max-margin problem in structured-output prediction has two properties that can be exploited for efficient optimization. First, we expect only a very small fraction of the constraints (and therefore a small number of parameters) to be active, due to the hinge loss but more importantly due to the overlap of information among classes represented via the joint feature map. The analysis of sparsity is presented in Section 1.4.2. Secondly, the constraint matrix is (at least) block diagonal (diagonal for the $\text{SVM}_2^{\triangle s}$ variant), resulting in dual variables to be coupled only within a block of variables associated with the same training instance.

We propose a general cutting plane algorithm (Kelley, 1960) for cost-sensitive support vector machines. This algorithm exploits the above mentioned properties of the maximum-margin problem, so that only a small number of constraints are examined explicitly and a small size QP is solved at each iteration of the algorithm. In a nutshell, the algorithm starts with no constraints (which corresponds to the most relaxed version of the primal) and iterative adds constraints via a variable selection approach in the dual formulation leading to tighter relaxations.

Pseudo-code of the algorithm is depicted in Algorithm 1.1. The algorithm maintains working sets S_i for each instance 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 finds the (potentially) "most violated" constraint of $\mathbf{x}_i$, involving some output value $\hat{\mathbf{y}}$. If the scaled margin violation of this constraint exceeds the current value of ξ_i by more than ϵ, the dual variable corresponding to $\hat{\mathbf{y}}$ is added to the working set, leading to the cut-off of the current primal solution from the feasible set (see Figure 1.3). Once a constraint has been added, the quadratic program is

Algorithm 1.1 Cost-sensitive support vector machines ($\text{SVM}_1^{\triangle s}$ and $\text{SVM}_2^{\triangle s}$).

```
1:   input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, $C$, $\epsilon$
2:   output: $\boldsymbol{\alpha}$
3:   $S_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
4:   repeat
5:     for $i = 1, \ldots, n$ do
6:       $H(\mathbf{y}) \equiv \begin{cases} (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w}\rangle) \triangle(\mathbf{y}_i, \mathbf{y}) & (\text{SVM}_1^{\triangle s}) \\ (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w}\rangle) \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})} & (\text{SVM}_2^{\triangle s}) \end{cases}$
         where $\mathbf{w} \equiv \sum_j \sum_{\mathbf{y}' \in S_j} \alpha_{(j\mathbf{y}')} \delta\Psi_j(\mathbf{y}')$.
7:       compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$
8:       compute $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$
9:       if $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$ then
10:        $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
10a:         /* Variant (a): perform full optimization */
           $\alpha_S \leftarrow$ optimize the dual of $\text{SVM}_1^{\triangle s}$ or $\text{SVM}_2^{\triangle s}$ over $S$, $S = \cup_i S_i$
10b:         /* Variant (b): perform subspace ascent */
           $\alpha_{S_i} \leftarrow$ optimize the dual of $\text{SVM}_1^{\triangle s}$ or $\text{SVM}_2^{\triangle s}$ over $S_i$
13:      end if
14:    end for
15:  until no $S_i$ has changed during iteration
```

solved with respect to S or S_i (leading to smaller QP problems) depending on the ratio of the complexity of the constraint selection in step 7 and the complexity of solving the relaxed QP. Since at each iteration only one constraint is added, it is possible to initialize the QP solver to the current solution, which greatly reduces the runtime. If $\hat{\mathbf{y}}$ satisfies the soft-margin constraint up to ϵ precision, it implies that the rest of constraints are approximately satisfied as well and no further improvement is necessary for $\mathbf{x}_i$.

Due to the generality of the algorithm, by implementing the feature mapping $\Psi(\mathbf{x}, \mathbf{y})$ (either explicit or via a joint kernel function), the cost function $\triangle(\mathbf{y}, \mathbf{y}')$ and the maximization in step 7 accordingly one achieves a max-margin classifier for all the special cases considered in Section 1.2 as well as others such as string-to-string matching.

1.4.1 Finding most violated constraint

We now describe efficient algorithms to compute the maximization in step 7 of Algorithm 1.1. Note that the crucial computation involves finding

$$\underset{\mathbf{y} \neq \mathbf{y}_i}{\text{argmax}} \triangle(\mathbf{y}, \mathbf{y}^i) F(\mathbf{x}_i, \mathbf{y}), \tag{1.16}$$

since the other terms computed by simple operations. There exists well-known DP algorithms for finding $\text{argmax}_{\mathbf{y}} F(\mathbf{x}, \mathbf{y})$ for various dependency structures, such as the Viterbi algorithm for sequences and CKY algorithm for parsing (Manning and Schuetze, 1999). When $\triangle$ is 0/1 loss, (1.16) can be found by modifying these algorithms to find the n-best labeling of an observation, for $n = 2$ (Schwarz and

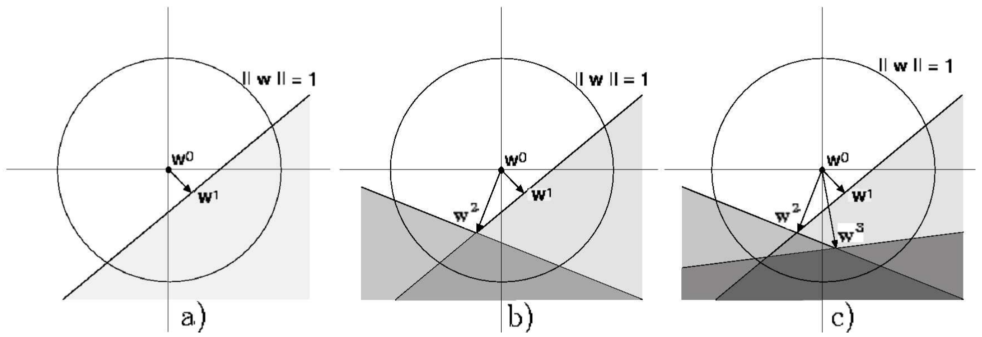


Figure 1.3 Cutting plane algorithm. Successive steps of the cutting plane algorithm. In first step no constraints have been added (no shading), $\mathbf{w}^0 = 0$ is the current solution. (a) Second step: The (potentially) most violated constraint has been added. It cuts off the current solution $\mathbf{w}^0$ from the feasible region (shaded).(b) Third step: One more violated constraint is added, and the new solution is computed. (c) Fourth step: The process is repeated until there are no more violating constraints.

Chow, 1990). In cases where $\triangle(\mathbf{y}_i, \cdot)$ only takes on a finite number of values, a generic strategy is a two stage approach, where one first computes the maximum over those $\mathbf{y}$ for which the cost $\triangle(\mathbf{y}_i, \mathbf{y})$ is constant, and then maximizes over the finite number of levels. However, this strategy can scale the computational complexity by the size of $\mathbf{y}$ (e. g. when the cost is the Hamming loss). We now present the recursion rules of a simple modification of the DP algorithms to compute (1.16) for Hamming loss and $1 - F_1$ score. The resulting algorithms are as efficient as the original DP algorithm (up to a small constant). This approach can easily be generalized to any cost function that decomposes into factors that are linear in the cost of subcomponents of $\mathbf{y}$.

Note that Hamming loss is given by $\triangle(\mathbf{y}, \bar{\mathbf{y}}) = \sum_{t=1}^{T} \bar{\delta}(y^t, \bar{y}^t)$, where y^t denotes the t^{th} component of $\mathbf{y}$ (e. g. t^{th} position in a sequence of length T) and $\bar{\delta}(a, b)$ is 0 is $a = b$ and 1 otherwise. Let $c(t, \sigma, \sigma'; \mathbf{w})$ be the local contribution of assigning σ to the t^{th} component with respect to $\mathbf{w}$ given the previous variable assignments σ' . Suppressing the dependence on $\mathbf{y}^i$ and $\mathbf{w}$, the recursive rules are given by

$$\begin{aligned}
S_t(\sigma) &= \max_{\sigma'} \left(S_{t-1}(\sigma') + \bar{\delta}(y_i^t, \sigma) F_{t-1}(\sigma') + c(t, \sigma, \sigma')[D_{t-1}(\sigma') + \bar{\delta}(y_i^t, \sigma)]\right) \\
A_t(\sigma) &= \operatorname*{argmax}_{\sigma'} \left(T_{t-1}(\sigma') + \bar{\delta}(y_i^t, \sigma) F_{t-1}(\sigma') + c(t, \sigma, \sigma')[D_{t-1}(\sigma') + \bar{\delta}(y_i^t, \sigma)]\right) \\
D_t(\sigma) &= D_{t-1}(A_t(\sigma)) + \bar{\delta}(y_i^t, \sigma) \\
F_t(\sigma) &= F_{t-1}(A_t(\sigma)) + c(t, \sigma, A_t(\sigma)).
\end{aligned}$$

where all the variables at $t = 0$ is 0. Then, the best labeling is achieved by reconstructing the path from A via $\operatorname{argmax}_\sigma S_T(\sigma)$ in reverse direction.

Note that F_1 score, which is the harmonic mean of precision and recall, is given by $\triangle(\mathbf{y}, \bar{\mathbf{y}}) = 2a/(p + o)$, where a is the number of correctly predicted subcomponents, p is the number of predicted subcomponents and o is the number of correct subcomponents. Define $\hat{c}$ such that $\hat{c}(t, \sigma, \sigma') = 1$ if labeling t^{th} component with σ increases the number of predicted components given previous labeling σ' and 0 otherwise. Then the recursive rules are given by

$$\begin{aligned}
R_t(\sigma) &= \max_{\sigma'} \left[R_{t-1}(\sigma') + c(t,\sigma,\sigma') \left(1 - 2\frac{D_t(\sigma)}{N_t(\sigma)+o} \right) + 2\frac{F_{t-1}(\sigma')}{N_t(\sigma)+o} V \right] \\
A_t(\sigma) &= \operatorname*{argmax}_{\sigma'} \left[R_{t-1}(\sigma') + c(t,\sigma,\sigma') \left(1 - 2\frac{D_t(\sigma)}{N_t(\sigma)+o} \right) + 2\frac{F_{t-1}(\sigma')}{N_t(\sigma)+o} V \right] \\
D_t(\sigma) &= D_{t-1}(A_t(\sigma)) + \delta(y_i^t, \sigma) \\
F_t(\sigma) &= F_{t-1}(A_t(\sigma)) + c(t,\sigma,A_t(\sigma)) \\
N_t(\sigma) &= N_{t-1}(A_t(\sigma)) + \hat{c}(t,\sigma,A_t(\sigma)).
\end{aligned}$$

where $V = \frac{D_{t-1}(\sigma')\hat{c}(t,\sigma,\sigma')}{N_{t-1}(\sigma')+o} - \delta(\sigma, y_i^t)$. The best labeling is achieved by reconstructing the path from A via $\operatorname{argmax}_\sigma R_T(\sigma)$ in reverse direction.

1.4.2 Analysis

We now show that Algorithm 1.1 computes arbitrary close approximations to SVMs by evaluating only a polynomial number of constraints so that the exponentially many constraints are guaranteed to be satisfied up to an ϵ-precision without explicit evaluation. We start by providing lower bounds of the improvements in SVM$_2^{\triangle s}$ and SVM$_1^{\triangle s}$ at each iteration of Algorithm 1.1. The proof can be found in the appendix.

Proposition 5 *Let $\triangle_i \equiv \max_{\mathbf{y}}\{\triangle(\mathbf{y}_i, \mathbf{y})\}$ and $R_i \equiv \max_{\mathbf{y}}\{\|\delta\Psi_i(\mathbf{y})\|\}$. The dual objective improves by at least*

$$\frac{1}{2}\frac{\epsilon^2}{\triangle_i R_i^2 + \frac{n}{C}} \quad \textit{and} \ \min\left\{ \frac{C\epsilon}{2n}, \frac{\epsilon^2}{8\triangle_i^2 R_i^2} \right\} \tag{1.19}$$

in step 10 of Algorithm 1.1 for SVM$_2^{\triangle s}$ and SVM$_1^{\triangle s}$ respectively.

Theorem 6 *With $\bar{R} = \max_i R_i$, $\bar{\triangle} = \max_i \triangle_i$ and for a given $\epsilon > 0$, Algorithm 1.1 terminates after incrementally adding at most*

$$\max\left\{ \frac{2n\bar{\triangle}}{\epsilon}, \frac{8C\bar{\triangle}^3\bar{R}^2}{\epsilon^2} \right\}, \quad \textit{and} \quad \frac{C\bar{\triangle}^2\bar{R}^2 + n\bar{\triangle}}{\epsilon^2} \tag{1.20}$$

constraints to the working set S for the SVM$_1^{\triangle s}$ and SVM$_2^{\triangle s}$ respectively.

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 the amounts suggested by Proposition 5. Hence after t constraints, the dual objective will be at least t times these amounts. 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 $C\bar{\triangle}$ and $\frac{1}{2}C\bar{\triangle}$ for SVM$_1^{\triangle s}$ and SVM$_2^{\triangle s}$ respectively.* ■

To make the bounds more concrete, let us now examine how R is bounded for the special cases studied in Section 1.2. For hierarchical classification, we define $r_i \equiv \|\Phi(\mathbf{x}_i)\|$ and $S \equiv \max_{\mathbf{y}\in\mathcal{Y}} \|\Lambda(y)\|$. Using Proposition 1 and simple derivation, we can show that $\|\Psi(\mathbf{x}_i, y) - \Psi(\mathbf{x}_i, y')\|^2$ is upper bounded by

$$2\langle\Psi(\mathbf{x}_i, y), \Psi(\mathbf{x}_i, y)\rangle = 2\|\Phi(\mathbf{x}_i)\|^2\|\Lambda(y)\|^2 \leq 2r_i^2 S^2. \tag{1.21}$$

For label sequence learning, we define $r_i \equiv \max_t \|\Phi(\mathbf{x}_i^t)\|$ and $l = \max_i l_{\mathbf{x}_i}$. Then $\|\Psi(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \mathbf{y}')\|^2$ is upper bounded by

$$2\|\sum_t \Phi(\mathbf{x}_i^t) \otimes \Lambda^c(y^t)\|^2 + \eta^2\|\sum_t \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})\|^2 \leq 2l^2(R_i^2 + \eta^2) \tag{1.22}$$

In parsing, a sequence $\mathbf{x}$ of length l has $l-1$ internal and l pre-terminal nodes. Thus, $\|\Psi(\mathbf{x}, \mathbf{y})\|_1 = 2l-1$. Then, $\|\Psi(\mathbf{x}, \mathbf{y}) - \Psi(\mathbf{x}, \mathbf{y}')\|_2 \leq \sqrt{4l^2 + 4(l-1)^2} < 2\sqrt{2}l$.

1.5 Alternative margin formulations

In addition to the margin approaches of $\mathrm{SVM}_1^{\triangle s}$ and $\mathrm{SVM}_2^{\triangle s}$, a second way to incorporate the cost function into the optimization problem is to re-scale the margin as proposed by Taskar et al. (2003) for the special case of the Hamming loss,

$$\mathrm{SVM}_1^{\triangle m}: \qquad \min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \max_{\mathbf{y}\in\mathcal{Y}\setminus\mathbf{y}_i} (\triangle(\mathbf{y}_i, \mathbf{y}) - \langle\mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle)_+.$$

$\mathrm{SVM}_2^{\triangle m}$ can be defined similarly with the squared-hinge loss. Both $\mathrm{SVM}_1^{\triangle m}$ and $\mathrm{SVM}_2^{\triangle m}$ are upper bounds on $\mathcal{R}_{\mathcal{S}}^{\triangle}(\mathbf{w}^*)$. Dynamic programming algorithms similar to ones in Section 1.4.1 can be given to compute

$$\max_{\mathbf{y}\in\mathcal{Y}\setminus\mathbf{y}_i} (\triangle(\mathbf{y}_i, \mathbf{y}) - \langle\mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle\,. \tag{1.23}$$

and Algorithm 1.1 can be used for these formulations simply by changing line 6.

An advantage of $\mathrm{SVM}_1^{\triangle s}$ over $\mathrm{SVM}_1^{\triangle m}$ is its scaling invariance.

Proposition 7 *Suppose $\triangle' \equiv \eta\triangle$ with $\eta > 0$, i.e. $\triangle'$ is a scaled version of the original loss $\triangle$. Then the optimal weight vector $\mathbf{w}^*$ for $SVM_1^{\triangle' s}$ is also optimal for $SVM_1^{\triangle s}$ and vice versa, if we rescale $C' = C/\eta$.*

Proof *If $\mathbf{w}$ is fixed in $SVM_1^{\triangle s}$ and $SVM_1^{\triangle' s}$ then the optimal values for ξ_i^* in each of the problems are related to another by a scale change of η. By scaling C with the inverse of η, this effectively cancels.* ■

This is not the case for $\mathrm{SVM}_1^{\triangle m}$. One needs, for example, to rescale the feature map Ψ by a corresponding scale factor as well, which seems to indicate that one has to calibrate the scaling of the loss and the scaling of the feature map more carefully in the $\mathrm{SVM}_1^{\triangle m}$ formulation.

	flt 0/1	tax 0/1	flt $\triangle$	tax $\triangle$	
4 training instances per class (3-fold x-val)					
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(5-fold x-val)					
acc	20.20	20.46	20.20	21.73	+7.57 %
$\triangle$-loss	1.54	1.51	1.39	1.33	+13.67 %

Table 1.1 Results on the WIPO-alpha corpus. '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.

More importantly, $\text{SVM}_1^{\triangle m}$ can potentially give significant weight to output values $\mathbf{y} \in \mathcal{Y}$ that are not even close to being confusable with the target values $\mathbf{y}_i$, i. e. $F(\mathbf{x}_i, \mathbf{y}_i) - F(\mathbf{x}_i, \mathbf{y})$ might be large but smaller than $\triangle(\mathbf{y}_i, \mathbf{y})$. $\text{SVM}_1^{\triangle m}$, on the other hand, only depends on $\mathbf{y}$ for which $F(\mathbf{x}_i, \mathbf{y}_i) - F(\mathbf{x}_i, \mathbf{y}) \leq 1$, i. e. outputs only receive a relatively high discriminant score.

1.6 Experiments

We present experiments on hierarchical classification, label sequence learning and parsing, reported in Tsochantaridis et al. (2005).

1.6.1 Hierarchical Classification

Experiments were performed using a 1710 document subsample of the World Intellectual Property Organization (WIPO)-alpha document collection. The title and claim tags were indexed. Document parsing, tokenization and term normalization have been performed with the MindServer retrieval engine[2]. The cost function $\triangle$ is the height of the first common ancestor of the arguments y, y' in the taxonomy. Results in Table 1.1 show that the proposed hierarchical SVM learning architecture improves the performance over the standard multiclass SVM in terms of classification accuracy as well as in terms of the tree loss.

1.6.2 Label Sequence Learning

We performed experiments on a named entity recognition (NER) problem using 5-fold cross validation on a 300 sentence sub-corpus of the Spanish news wire article corpus provided by CoNLL2002. The label set consists of non-name and

2. http://www.recommind.com

Method	HMM	CRF	Perceptron	SVM
Error	9.36	5.17	5.94	5.08

Table 1.2 Results of various algorithms on the Named Entity Recognition task.

	SVM_2	$\text{SVM}_2^{\triangle s}$	$\text{SVM}_2^{\triangle m}$
Test Err	5.1±0.6	5.1±0.8	5.1±0.7
Const	2824±106	2626±225	2628±119

Table 1.3 Results of various joint kernel SVM formulations on NER.

	PCFG	SVM_2	$\text{SVM}_2^{\triangle s}$	$\text{SVM}_2^{\triangle m}$
Test Acc	55.2	58.9	58.9	58.3
Test F_1	86.0	86.2	88.5	88.4
Const	N/A	7494	8043	7117

Table 1.4 Results for learning a weighted context-free grammar on the Penn Treebank.

the beginning and continuation of person names, organizations, locations and miscellaneous names, resulting in a total of $|\Sigma| = 9$ different labels. The joint feature map $\Psi(\mathbf{x}, \mathbf{y})$ is the histogram of label-label interactions plus a set of features describing the observation-label interactions. For both perceptron and SVM, a second degree polynomial kernel was used. No special tuning was performed, and C was set to 1 and ϵ to 0.01. The cost function $\triangle$ is given by the Hamming distance. Results given in Table 1.2 for the zero-one loss, compare the generative HMM with Conditional Random Fields (CRF) Lafferty et al. (2001), perceptron Collins (2002) and the joint kernel 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 1.3 shows that all joint kernel 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. Note that for loss equal to 1 all SVM formulations are equivalent.

1.6.2.1 Parsing

Experiments were performed on a subset of the Penn Treebank corpus, where the 4098 sentences of length at most 10 from sections F2-21 were used as the training set, and the 163 sentences of length at most 10 from F22 were used as the test set. The feature representation include the grammar rules observed in the training data. The cost function is given by $\triangle(\mathbf{y}, \mathbf{y}^i) = 1 - F_1(\mathbf{y}_i, \mathbf{y})$. All results are for $C = 1$ and $\epsilon = 0.01$. All values of C between 10^{-1} to 10^2 gave comparable prediction performance.

The results are given in Table 1.4. For the zero-one loss (i.e. predicting the complete tree correctly), the generative PCFG model using the maximum likelihood estimate (MLE) achieves better accuracy than the max-margin approaches. However, optimizing the SVM for the F_1-loss outperforms PCFG significantly in terms of F_1-scores. Table 1.4 also shows that the total number of constraints added to the working set is small, roughly twice the number of training examples in all cases.

1.7 Conclusions

We presented a maximum-margin approach to learn functional dependencies for interdependent and structured output spaces. The key idea is to model the problem as a (kernelized) linear discriminant function over a joint feature space of inputs and outputs. We showed that this framework is applicable to a wide range of problems, in particular hierarchical classification, label sequence learning and parsing. We presented an efficient algorithm with polynomial time convergence. This algorithm combines the advantages of maximum-margin classifiers and kernels with the efficiency of dynamic programming algorithms. Experimental evaluation shows the competitiveness of our method.

A Proof of Proposition 5

Lemma 8 *For a symmetric, positive semi-definite matrix, $\boldsymbol{J}$, let*

$$\boldsymbol{\Theta}(\boldsymbol{\alpha}) = -\tfrac{1}{2}\boldsymbol{\alpha}'\boldsymbol{J}\boldsymbol{\alpha} + \langle \mathbf{h}, \boldsymbol{\alpha}\rangle \tag{A.1}$$

be concave in $\boldsymbol{\alpha}$ and bounded from above. Assume $\langle \nabla\boldsymbol{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle > 0$ for a solution $\boldsymbol{\alpha}^t$ and an optimization direction $\boldsymbol{\eta}$. Let $\beta \leq D$ for some $D > 0$. Then the improvement of the objective along $\boldsymbol{\eta}$ starting from $\boldsymbol{\alpha}^t$ $\delta\boldsymbol{\Theta}(\beta) \equiv \max_{0<\beta\leq D}\{\boldsymbol{\Theta}(\boldsymbol{\alpha}^t + \beta\boldsymbol{\eta})\} - \boldsymbol{\Theta}(\boldsymbol{\alpha}^t)$ is bounded by

$$\frac{1}{2}\min\left\{D, \frac{\langle \nabla\boldsymbol{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle}{\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta}}\right\}\langle \nabla\boldsymbol{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle \tag{A.2}$$

For special cases where $\boldsymbol{\eta} = \boldsymbol{e}_r$ and where $\boldsymbol{\eta} = \boldsymbol{e}_r, \beta < D = \infty$, (A.2) *is equivalent to*

$$\frac{1}{2}\min\left\{D, \frac{\frac{\partial\boldsymbol{\Theta}}{\partial\alpha_r}(\boldsymbol{\alpha}^t)}{J_{rr}}\right\}\frac{\partial\boldsymbol{\Theta}}{\partial\alpha_r}(\boldsymbol{\alpha}^t) \quad \text{and} \quad \frac{1}{2J_{rr}}\left(\frac{\partial\boldsymbol{\Theta}}{\partial\alpha_r}(\boldsymbol{\alpha}^t)\right)^2. \tag{A.3}$$

Proof *Writing out $\delta\boldsymbol{\Theta}(\beta^*) = \beta\langle \nabla\boldsymbol{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle - \frac{\beta^2}{2}\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta}$, solving for β to maximize $\delta\boldsymbol{\Theta}(\beta^*)$ yields $\beta^* = \langle \nabla\boldsymbol{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle/(\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta})$. Substituting this value in $\delta\boldsymbol{\Theta}(\beta^*)$ gives the bound on improvement with unconstraint β,*

$$\delta\boldsymbol{\Theta}(\beta^*) = \langle \nabla\boldsymbol{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle^2/(2\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta}) > 0. \tag{A.4}$$

If $D < \beta^*$, i .e. $\langle \nabla\boldsymbol{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle/(\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta}) > D$, due to the concavity of $\boldsymbol{\Theta}$, $\beta = D$ and

$$\delta\boldsymbol{\Theta}(D) = D\left(\langle \nabla\boldsymbol{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle - \frac{D}{2}\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta}\right). \tag{A.5}$$

Combining the two cases yields (A.2). *When $\boldsymbol{\eta} = \boldsymbol{e}_r$, $\langle \nabla\boldsymbol{\Theta}, \boldsymbol{\eta}\rangle = \frac{\partial\boldsymbol{\Theta}}{\partial\alpha_r}$ and $\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta} = J_{rr}$, which yields the first term in* (A.3). *The second term is achieved by substituting ∞ for D.* ■

Proof of Proposition 5. We first prove the bound for $\mathrm{SVM}_2^{\triangle s}$. Using (1.15), (1.14), $\xi_i^* = \sum_{\mathbf{y}\neq\mathbf{y}_i}\frac{n\alpha^t_{(i\mathbf{y})}}{C\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}}$ given by the optimality equations for the primal variables and the condition of step 10, namely $\sqrt{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}\,(1 - \langle \mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}})\rangle) > \xi_i^* + \epsilon$ yields $\frac{\partial\boldsymbol{\Theta}}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\boldsymbol{\alpha}^t) \geq \frac{\epsilon}{\sqrt{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}}$. Inserting this and $J_{rr} = \|\delta\Psi_i(\hat{\mathbf{y}})\|^2 + \frac{n}{C\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}$ in second term of (A.3) yields the first term of the (1.19).

For $\mathrm{SVM}_1^{\triangle s}$, consider two cases:
Case I:
If the working set does not contain an element $(i\mathbf{y})$, $S_i = 0$, then we can optimize over $\alpha_{(i\hat{\mathbf{y}})}$ such that $\alpha_{(i\hat{\mathbf{y}})} \leq \triangle(\mathbf{y}_i, \hat{\mathbf{y}})\frac{C}{n} = D$. Then, via the condition of step 10 and $\xi_i^* \geq 0$, $\frac{\partial \mathbf{\Theta}}{\partial \alpha_{(i\hat{\mathbf{y}})}}(\boldsymbol{\alpha}^t) = 1 - \langle \mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}})\rangle > \frac{\xi_i^*+\epsilon}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})} \geq \frac{\epsilon}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}$. Substituting this and $J_{(i\hat{\mathbf{y}})(i\hat{\mathbf{y}})} \leq R_i^2$ in the first term of (A.3) yields

$$\delta\mathbf{\Theta} \geq \min\left\{\frac{C\epsilon}{2n}, \frac{\epsilon^2}{2R_i^2\triangle(\mathbf{y}_i,\hat{\mathbf{y}})^2}\right\} \tag{A.6}$$

Case II:
If $S_i \neq \emptyset$, optimization is performed over $\alpha_{(i\hat{\mathbf{y}})}$ and $\alpha_{(i\mathbf{y})}, \forall \mathbf{y} \in S_i$. We need to upper bound $\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta}$ and lower bound $\langle \nabla\mathbf{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle$. Wlog let $\eta_{(i\hat{\mathbf{y}})} = 1$ and $\eta_{(i\mathbf{y})} = -\frac{\alpha_{(i\mathbf{y})}}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}\frac{n}{C} \leq 0$ for $(i\mathbf{y}) \in S_i$. Then $\boldsymbol{\alpha}^t + \beta\boldsymbol{\eta} \geq 0$ since $\beta \leq \frac{C}{n}\triangle(\mathbf{y}_i, \hat{\mathbf{y}})$.
In order to bound $\langle \nabla\mathbf{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle$, notice that for $\delta \geq \epsilon > 0$,

$$\triangle(\mathbf{y}_i, \hat{\mathbf{y}})\left(1 - \langle \mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}})\rangle\right) = \xi_i^* + \delta, \tag{A.7a}$$

$$\triangle(\mathbf{y}_i, \mathbf{y})\left(1 - \langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y})\rangle\right) = \xi_i^*, \quad \mathbf{y} \in S_i \tag{A.7b}$$

Then via $\langle \nabla\mathbf{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle = \sum_{\mathbf{y}} \eta_{(i\mathbf{y})}\left(1 - \langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y})\rangle\right)$, we get

$$\langle \nabla\mathbf{\Theta}(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle = \frac{\xi_i^*}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}\left(1 - \frac{n}{C}\sum_{\mathbf{y}}\frac{\alpha_{(i\mathbf{y})}}{\triangle(\mathbf{y}_i,\mathbf{y})}\right) + \frac{\delta}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})} \geq \frac{\epsilon}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})} \tag{A.8}$$

Using $\sum_{\mathbf{y}\neq\hat{\mathbf{y}}} \alpha_{(i\mathbf{y})}^t \leq \triangle_i \sum_{\mathbf{y}\neq\hat{\mathbf{y}}} \frac{\alpha_{(i\mathbf{y})}^t}{\triangle(\mathbf{y}_i,\mathbf{y})} \leq \frac{C\triangle_i}{n}$, we bound $\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta}$ as

$$\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta} = J_{(i\hat{\mathbf{y}})(i\hat{\mathbf{y}})} - 2\frac{n}{C}\sum_{\mathbf{y}\neq\hat{\mathbf{y}}}\frac{\alpha_{(i\mathbf{y})}^t J_{(i\hat{\mathbf{y}})(i\mathbf{y})}}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})} + \frac{n^2}{C^2}\sum_{\mathbf{y}\neq\hat{\mathbf{y}}}\sum_{\mathbf{y}'\neq\hat{\mathbf{y}}}\frac{\alpha_{(i\mathbf{y})}^t\alpha_{(i\mathbf{y}')}^t J_{(i\mathbf{y})(i\mathbf{y}')}}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})\triangle(\mathbf{y}_i,\hat{\mathbf{y}})} \tag{A.9a}$$

$$\leq R_i^2 + 2\frac{nR_i^2}{C\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}\sum_{\mathbf{y}\neq\hat{\mathbf{y}}}\alpha_{(i\mathbf{y})}^t + \frac{n^2R_i^2}{C^2\triangle(\mathbf{y}_i,\hat{\mathbf{y}})^2}\sum_{\mathbf{y}\neq\hat{\mathbf{y}}}\sum_{\mathbf{y}'\neq\hat{\mathbf{y}}}\alpha_{(i\mathbf{y})}^t\alpha_{(i\mathbf{y}')}^t \tag{A.9b}$$

$$\leq R_i^2 + 2\frac{R_i^2\triangle_i}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})} + \frac{R_i^2\triangle_i^2}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})^2} \leq \frac{4R_i^2\triangle_i^2}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})^2} \tag{A.9c}$$

Substituting (A.8) and (A.9c) into (A.2) yields

$$\min\left\{\frac{C\epsilon}{2n}, \frac{\epsilon^2}{8R_i^2\triangle_i^2}\right\} \tag{A.10}$$

Combining (A.6) and (A.10) yields the second term of (1.19). ■

References

Y. Altun, D. McAllester, and M. Belkin. Maximum-margin semi-supervised learning for structured variables. In *(NIPS)*, 2005.

M. Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.

M. Collins and N. Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, pages 625–632, Cambridge, MA, 2002. MIT Press.

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

J. E. Kelley. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial Applied Mathematics*, 8:703–712, 1960.

G. S. Kimeldorf and G. Wahba. Some results on Tchebycheffian spline functions. *J. Math. Anal. Applic.*, 33:82–95, 1971.

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

J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *Proceedings of the 21st International Conference (ICML)*, 2004.

C. D. Manning and H. Schuetze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98 - 14, Microsoft Research, 1998.

B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

R. Schwarz and Y. L. Chow. The n-best algorithm: An efficient and exact procedure for finding the n most likely hypotheses. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 81–84, 1990.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 16*, Cambridge, MA, 2003. MIT Press.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 2005.