Support Vector Machine Learning for Interdependent and Structured Output Spaces

by

Ioannis Tsochantaridis

B. S., University of Patras, 1995

M. Sc., Brown University, 2000

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island

May 2005

This dissertation by Ioannis Tsochantaridis is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date 4/29/05

Thomas Hofmann, Advisor

Recommended to the Graduate Council

Date 5/12/05

Tom Dean, Reader

Date 5/11/05

Thorsten Joachims, Reader

Date 5/12/05

Pascal Van Hentenryck, Reader

Approved by the Graduate Council

Date 5/18/05

Karen Newman, Dean of the Graduate School

Acknowledgements

The ideas and parts of the text presented in this thesis were developed in collaboration with the following people: Thomas Hofmann, Thorsten Joachims, Yasemin Altun and Lijuan Cai. Parts of this thesis have previously appeared as [19, 2, 46].

Contents

List of Tables viii

List of Figures ix

1 Introduction 1

1.1 Motivation 1

1.2 Thesis 5

1.3 Overview 7

2 Background 9

2.1 Introduction 9

2.2 Linear Classifiers 10

2.3 Support Vector Machines 11

2.4 Sparseness 14

2.5 Kernels 15

2.5.1 Properties of kernels 17

2.6 Multiclass Classification 18

2.6.1 Indirect Multi-class Classification 19

2.6.2 Direct Multi-class Classification 20

3 Joint Kernel Support Vector Machines 22

3.1 Introduction 22

3.2 Discriminant Function and Joint Feature Maps 23

3.3 Perceptron Learning 25

3.4 Hard-Margin Support Vector Machines 27
3.5 Cutting Plane Algorithm 29
3.5.1 Algorithm 29
3.5.2 Analysis 33
3.5.3 Discussion 36
3.6 Soft-Margin Support Vector Machines 37
3.7 Loss-Sensitive Support Vector Machines 38
3.7.1 Slack Re-scaling 39
3.7.2 Margin Re-scaling 40
3.7.3 Algorithm 41
3.7.4 Analysis 42
3.7.5 Discussion 49

4 Special Cases **51**
4.1 Introduction 51
4.2 Multiclass Classification 52
4.2.1 Modeling 52
4.2.2 Algorithms 53
4.2.3 Sparseness 53
4.3 Multiclass Classification with Output Features 53
4.3.1 Modeling 53
4.3.2 Algorithms 58
4.3.3 Sparseness 58
4.3.4 Application: Classification with Taxonomies 58
4.4 Label Sequence Learning 59
4.4.1 Modeling 60
4.4.2 Algorithms 61
4.4.3 Sparseness 62
4.4.4 Application: Named Entity Recognition 63
4.4.5 Discussion 64
4.5 Weighted Context-Free Grammars 65

4.5.1 Modeling 65
4.5.2 Algorithms 66
4.5.3 Sparseness 67
4.5.4 Application: Natural Language Parsing 69
4.6 String-to-String Mappings 70
4.6.1 Modeling 70
4.6.2 Algorithms 71
4.6.3 Sparseness 71
4.6.4 Example: Synthetic Data 72
4.6.5 Discussion 73

5 Conclusions **76**
5.1 Summary 76
5.2 Future Work 77

A Derivation of Dual Formulations **78**

B Notation **82**

List of Tables

4.1 Classification with taxonomies 59
4.2 Named entity recognition 63
4.3 Named entity recognition 64
4.4 Contex-free grammars 68
4.5 String-to-string mapping 71
4.6 Loss functions 74
4.7 Joint fearure maps 75

B.1 Notation 82

List of Figures

1.1 Text classification 2
1.2 Hierarchical text classification 3
1.3 Named entity recognition 4
1.4 Parsing 5
1.5 Machine Translation 6

2.1 Text classification feature map: bag-of-words model 11
2.2 Hard-margin support vector machines 12
2.3 Soft-margin support vector machines 13
2.4 Non-linear kernels 16

3.1 Illustration of a natural language parsing model 23
3.2 Cutting plane algorithm 30

4.1 Classification with taxonomies 57
4.2 Label sequence learning 60

Chapter 1

Introduction

1.1 Motivation

In recent years, the field of machine learning [33] has evolved to produce a set of powerful methods to complement knowledge engineering for solving classification problems among others. The task of classification is to assign objects to a set of predefined classes. In text classification, in particular, the object could be a news article, and the class an identifier that corresponds to the topic of the article, e.g. politics, art, sports, etc. The document shown in Figure 1.1, for example, obviously belongs to the general class "sports". In knowledge engineering, a human expert would construct a set of rules in order to automatically perform the classification task. One such rule, for the example above, could be the following: "if the document contains the word *football* then assign it to the class *sports*". Although, it may be relatively easy for a human to assign an object to a class, it is not always as simple to provide a set of rules that perform the classification with acceptable accuracy. It may also be impractical to provide such rules for a wide variety of classes. Furthermore, for classification tasks, where the objects to be classified are not as intuitive as natural language text, human experts may be able to specify object/class pairs, but not a precise relationship between objects and the corresponding classes.

The machine learning approach to solving classification problems is to automatically infer a classification rule based on such a set of object/class pairs. This is a

04 July 2004 19:45 (Portugal Local Time) - (Luz)

Greece kings of Europe

Greece pulled off arguably the biggest shock at a major football championship as a goal by Angelos Charisteas gave them victory at UEFA EURO 2004™. Having begun the tournament as 80-1 outsiders, the achievement of Otto Rehhagel's team of European journeymen is hard to put into perspective, but once again the belief they have shown in themselves throughout this amazing adventure was too strong for their opponents.

Figure 1.1: Text Classification. The first paragraph of an article that appeared on the web. A human can easily recognize the text as belonging to the class 'sports'.

general problem of learning functions between input and outputs based on a set of input/output pairs. The latter, which are called training examples, define the underlying relationship between inputs and outputs. The goal of machine learning is to provide computationally efficient algorithms that can construct a function, from a certain function class, that captures the underlying relationship, so that it does not only classify correctly the training examples but it also generalizes well to unknown examples. Human expertise is very critical to the success of this learning process in specifying the representation of the inputs, and selecting an appropriate function class. Going back to the text classification example, the underlying document representation could be the frequency of the words in the document (bag-of-words model), and the function class that of linear classifiers.

So far we have considered classification problems where the outputs are simple, that is, they can be characterized by an arbitrary identifier. However, in many real-world applications the outputs are often complex, in that (i) there are dependencies between classes, and (ii) the classes are sequences, trees, or have some other internal structure. To be more precise let's consider a couple of examples. In text classification, when there is a large number of classes, the classes are often organized in non-flat taxonomies. Figure 1.2 shows part of the hierarchical taxonomy provided

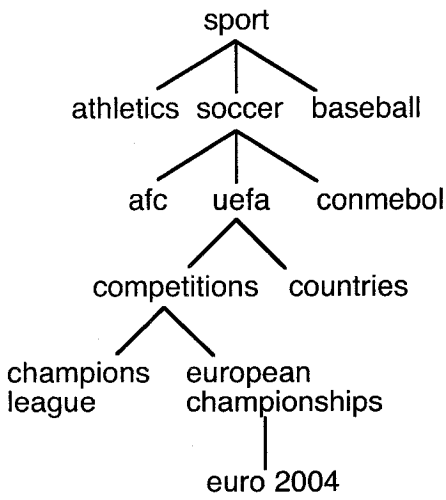


Figure 1.2: Hierarchical text classification. Part of the hierarchical taxonomy produced by the open directory project (DMOZ). Leaves represent the most specific classes.

by the open directory project (DMOZ) to organize web-page content. The document shown in Figure 1.1 belongs to the class "euro 2004", which is about the specific european football championship that took place in 2004, and falls under the classes "european championships", "competitions", "uefa", "soccer", on a path from specific to more general classes, that eventually leads to the root class "sport". There is clearly useful information encoded in this taxonomy, in particular there is a dependency between classes, that can be used when learning a classifier. This is particularly useful when there are very few available training examples, as it is often the case for very specific classes such as the class "euro 2004"

Another set of classification problems, that involve structured outputs, come from the area of natural language processing. In the problem of named entity recognition, we are interested in classifying each word in a sentence as being a named entity (person, organization, location, etc) or not. Consider the example shown in Figure 1.3. Due to the fact that the word "UEFA" is a named entity, in particular the name of an organization, it is likely that the following word, "Euro" is a continuation of the name of the organization, also a named entity. Clearly, the sequential dependencies,

04 July 2004 19:45 (Portugal Local Time) - (Luz)

Greece kings of Europe

[Greece/**LOCATION**] pulled off arguably the biggest shock at a major football championship as a goal by [Angelos Charisteas/**PERSON**] gave them victory at [UEFA EURO 2004™/**ORGANIZATION**]. Having begun the tournament as 80-1 outsiders, the achievement of [Otto Rehhagel/**PERSON**]'s team of European journeymen is hard to put into perspective, but once again the belief they have shown in themselves throughout this amazing adventure was too strong for their opponents.

Figure 1.3: Named entity recognition for an abstract of a news article taken from the web. Words immediately following named entities are very likely to be named entities too.

implied by the ordering of the words in a sentence, suggest that we should consider this problem as a sequence learning problem, rather than considering each word in isolation. In the problem of natural language parsing, as the example in Figure 1.4 shows, the output is the parse of a sentence, which is a tree structure of grammar rules that span parts of the input sentence. Lastly, in the problem of machine translation (see Figure 1.5) the output is a sequence of words in a target language, based on the input, which similarly is a sequence of words in a source language.

Support vector machines is a machine learning method that has been very successful in classification problems with simple outputs. The reason is that they provide a computationally efficient algorithm that can make use of powerful input representation, and construct highly accurate classifiers. Apart from the experimental verification of the generalization performance of support vector machines in a variety of tasks, there is also strong theoretical justification. Although support vector machines have focused on designing flexible and powerful input representations, that deal with structured inputs, for example, sequences, trees, or graphs, existing support vector machines ignore the dependencies/structure in the output, as they can only consider simple outputs.

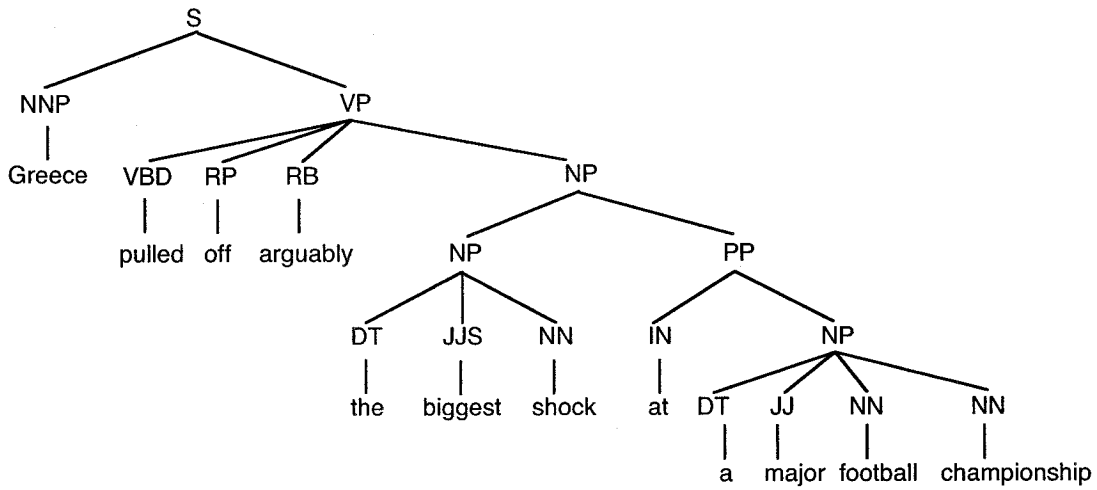


Figure 1.4: Parsing. The parse tree of a sentence.

1.2 Thesis

The main topic addressed in this dissertation is the generalization of support vector machine learning, so that it can be efficiently applied to classification problems that involve interdependent and structured outputs, such as the ones described in the previous section.

These problems fall into two generic cases: first, problems where outputs themselves can be characterized by certain output-specific attributes, and learning should occur across outputs as much as across inputs, for example, text classification, where the outputs (classes) are organized in non-flat taxonomies, and consequently specific classes may inherit attributes from more general ones (predecessors in the taxonomy); second, problems where the outputs are structured, that is, they describe a configuration over components, with possible dependencies among these components, for example, parsing, where an output (parse of a sentence) describes a tree configuration over components (grammar rules) that span part of the input (sentence).

An indirect approach, in order to use existing support vector machines for such classification problems, would be to ignore the interdependencies and structure of outputs, treating, for example, any taxonomy as flat in the former case, and learning

Greece kings of Europe

by Adam Szreter from Estádio da Luz

Greece pulled off arguably the biggest shock at a major football championship as a goal by Angelos Charisteas gave them victory at UEFA EURO 2004™. Having begun the tournament as 80-1 outsiders, the achievement of Otto Rehhagel's team of European journeymen is hard to put int perspective, but once again the belief they have shown in themselves throughout this amazing adventure was too strong for their opponents.

Griechenland holt europäische Krone

Von Andreas Alf, Estadio da Luz

Die Sensation ist vollkommen: Der neue Europameister heißt Griechenland. Ein Kopfballtor von Angelos Charisteas in der 57. Minute besiegelte den 1:0-Finalsieg gegen Gastgeber Portugal in Lissabons Estadio da Luz und ließ die Hellenen endgültig den Fußball-Olymp emporsteigen. Die favorisierte "Seleccao" fand gegen die abermals unglaublich diszipliniert spielende Mannschaft von Trainer Otto Rehhagel kein Mittel. Der deutsche Coach ist im Land der Götter nun unsterblich.

Figure 1.5: Machine translation. Given the English text, one would like to obtain the equivalent German text.

the grammar rules independently in the latter, thus, losing much useful information.

We approach these problems by generalizing multiclass support vector machines [12] to the broad problem of learning for interdependent and structured outputs. In order to do so, we specify discriminant functions that exploit the dependencies and structure of outputs. In that respect, our approach follows the work of [7, 10] on perceptron learning with a similar class of discriminant functions for problems with structured outputs. Note that the naive approach of treating each structured output as a separate class in a multiclass classification scenario is often intractable, since it involves a very large number of classes, that is, it depends on the size of the output space; in parsing, for example, the output space comprises all admissible parse trees, whose number is exponential in the length of the input.

Furthermore, we appropriately generalize the well-known notion of a separation margin and derive a corresponding maximum-margin formulation . A similar maximum margin formulation has been proposed in [8]. Yet, it depends on the size of the output space, therefore it requires some external process to enumerate a small number of candidate outputs for a given input. For a large class of problems, however, there is a polynomial-time algorithm that produces the highest scoring output with

respect to a given discriminant function. For this class of problems, we propose a cutting plane algorithm that does not depend on the size of the output space, and solves the corresponding optimization problem in polynomial time, taking advantage of the sparseness of the maximum margin solution.

In addition, for classification problems that involve interdependent and structured outputs, more appropriate loss functions than the missclassification loss (0-1 loss) function are needed to penalize for incorrect predictions. In natural language parsing, for example, a predicted parse tree that is almost correct, and differs from the correct tree in only a few components (rules), should be penalized less than an output that is completely different. Another support vector machine formulation for structured output problems, proposed in [45], essentially makes use of a loss that is proportional to the number of components in which an incorrect output differs from the correct one. Our framework, however, allows for the incorporation of arbitrary loss functions in the learning algorithm. This is very useful, since loss functions that are commonly used for the evaluation of a given problem, can be directly incorporated in the learning algorithm, and thus improve the generalization performance with respect to the specific loss function. In parsing, for example, the correctness of an parse tree is typically measured by its F_1 score, the harmonic mean of precision and recall as calculated based on the overlap of components between the two trees.

Besides the respective theoretical results, we empirically evaluate our approach for a number of specific problem instantiations, namely, classification with class taxonomies, label-sequence learning, learning weighted context-free grammars and string-to-string mappings.

1.3 Overview

Having presented the topic of this dissertation, the rest goes as follows. Chapter 2 gives a short introduction to support vector machines (SVM), and in particular multiclass SVMs, upon which the proposed generalization is based. The core of the thesis is Chapter 3. We first present a novel framework for learning linear discriminant

functions over joint input-output spaces. We then propose a generic optimization algorithm that generalizes support vector machine learning for interdependent and structured output spaces. We finally provide an analysis of the algorithm, and prove the sparseness of the proposed solution. In the following chapter we discuss several interesting special cases, instantiations of the general framework, and demonstrate the versatility and effectiveness of our approach by experimental evaluation. We conclude in Chapter 5 with a summary of contributions and suggestions for future work.

Chapter 2

Background

2.1 Introduction

In this chapter we will give a short introduction to support vector machines (SVMs) [4]. There are several books written on support vector machines [47, 13, 41], and the reader should refer to these sources of information for an extensive study.

Support vector machines rely on two significant ideas. First, the notion of the separation margin of a set of training examples, that suggests the use the maximum margin principle in choosing the appropriate classifier (the one that separates the training data by the largest margin) for a certain classification problem. The maximum margin principle, comes with several advantages. It leads to robust classifiers with competitive performance over a wide variety of applications. It also provides a nice theoretical argument for learning theory to study the performance of the resulting classifiers and prove strong generalization results. Furthermore, it constructs sparse solutions that involve only a subset of the training examples, and consequently to computationally attractive algorithms.

Second, the use of kernel functions, that allow support vector machines to implicitly employ powerful input representations, and thus efficiently construct non-linear classifiers, as well as classifiers for complex inputs such as sequences, trees, or graphs (cf. Section 2.5).

The maximum margin principle and the use of kernel functions for classification

are not exclusive to support vector machines. There are other classification algorithms that are proven to produce maximum margin classifiers (cf. boosting [16]), or that can be adapted to make use of kernels (cf. principal component analysis [42]). Support vector machines, nevertheless, put them together in an artful way to produce powerful machine learning methods. In the following sections we will illustrate these ideas, and highlight some significant results.

2.2 Linear Classifiers

For a moment, let us go back to the problem of text classification, presented in the introduction, and, say, that we are interested in learning to classify documents as to whether they belong to the class "euro 2004" or not. The learning algorithm, given a training set of input/output pairs (documents about "euro 2004" and documents not related to "euro 2004"), should find an appropriate discriminant function f from a restricted function class that performs the specific classification task with some accuracy.

A common representation of the inputs in text classification is the bag-of-words model [40, 21], illustrated in Figure 2.1. It comprises a (sparse) vector space representation, that ignores the sequence of the words in a document. A document is only represented by the frequencies of the words it contains (vector $\mathbf{x}$). As this is a binary classification problem, the output y is conveniently represented by either 1 or -1.

In the linear discrimination case, the discriminant functions are restricted to be linear in the components of $\mathbf{x}$. The considered class of functions (hyperplanes)

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \quad \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \tag{2.1}$$

correspond to decision functions of the form

$$f(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b) \tag{2.2}$$

Solving this problem amounts to finding appropriate values for the parameters $\mathbf{w}$, b.

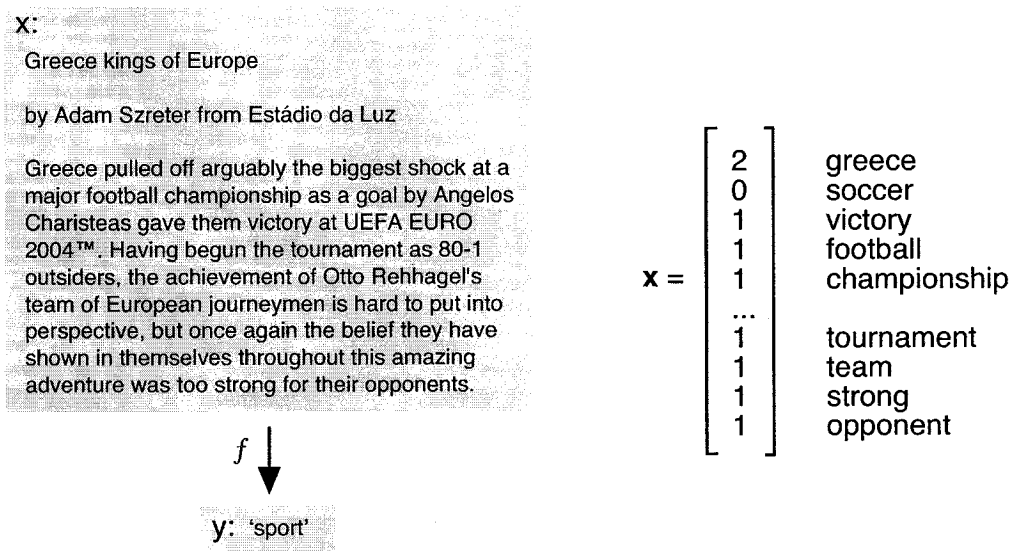


Figure 2.1: Text classification feature map: bag of words model. The text shown in the text is represented by the vector of the word frequences. The word sequence is ignored.

The margin of a training example $(\mathbf{x}, y)$ with respect to a separating hyperplane $(\mathbf{w}, b)$ is defined as

$$\gamma = y(\langle \mathbf{w}, \mathbf{x} \rangle + b) \tag{2.3}$$

which for canonical parameterization of the hyperplanes ($\|\mathbf{w}\| = 1$) is the Euclidean distance of the training input $\mathbf{x}$ to the hyperplane and b is a bias parameter. Geometrically, a separating hyperplane splits the input space into two half spaces, one for each class, and the optimal hyperplane is orthogonal to the shortest segment connecting the convex hulls of the two classes and intersects it at its midpoint.

2.3 Support Vector Machines

The support vector machine is a classification method that is based on the principle of margin maximization. SVMs generalize the linear discrimination method by choosing the maximum margin hyperplane with respect to a training sample $\mathcal{S}$. For linearly separable data $\mathcal{S} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} \in \mathbb{R}^d \times \{1, \ldots, k\}$, and

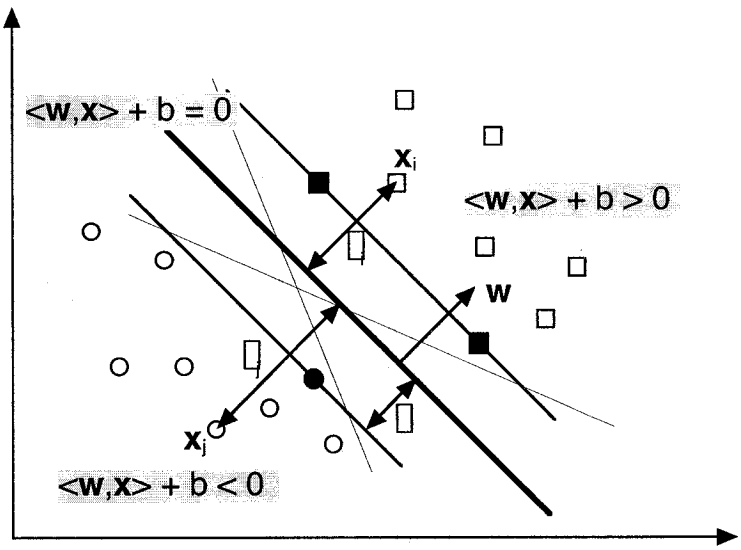


Figure 2.2: Hard-magin support vector machines. The data points of the two classes (circles and squares) are perfectly separable. Among all possible separating hyperplanes, we choose the one that maximizes the margin of separation between the classes. The black circles and squares are the ones that define the support of the maximum margin separating hyperplane. γ is the maximum margin achieved over all possible hyperplanes that separate the data.

$\mathbf{w}$-parameterized hyperplanes, there are in general many separating hyperplanes that separate the training data perfectly. These hyperplanes form the so-called version space. The maximum margin principle suggests to choose $\mathbf{w}^*$ among the parameters in version space so that the margin of separation between the classes is maximized (see Figure 2.2)

$$\mathbf{w}^* = \underset{\mathbf{w}:\|\mathbf{w}\|=1}{\operatorname{argmax}} \left\{ \min_{i=1,\ldots,n} \gamma_i(\mathbf{w}) \right\} \tag{2.4}$$

This problem can be equivalently expressed as a convex quadratic program, leading to the hard margin SVM formulation:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 \tag{2.5a}$$

$$\text{subject to: } \gamma_i(\mathbf{w}) \geq 1 \quad \forall i \in \{1,\ldots,n\} \tag{2.5b}$$

Obviously for data that is not separable the version space is empty. For example, the dataset shown in Figure 2.3 contain points that belong to two distinct classes.

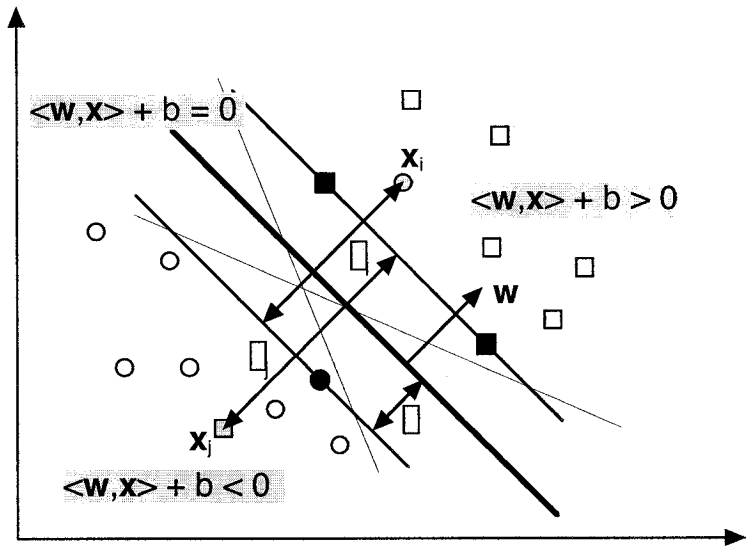


Figure 2.3: Soft-margin support vector machines. Example where the classes are non separable by a hyperplane. In this case we choose the hyperplane that at the same time maximizes the margin and minimizes a penalty for the violations of the margin. The variable ξ_i, ξ_j are the slack variables.

This dataset is not separable by a hyperplane. In order to be able to deal with non separable datasets one introduces slack variables ξ_i, one for every training example, and augments the objective function by an additional penalty term. The penalty term is often proportional to the sum of the slack variables (L_1 penalty) giving the so-called hinged-loss, or else proportional to the squared slack variables (L_2 penalty).

Linear penalties lead to the following standard quadratic program for soft-margin SVMs [11]

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{2.6a}$$

$$\text{subject to: } \gamma_i(\mathbf{w}) \geq 1 - \xi_i,\ \xi_i \geq 0 \quad \forall i \in \{1, \ldots, n\} \tag{2.6b}$$

and quadratic penalties to the following quadratic program

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2n}\sum_{i=1}^{n}\xi_i^2 \tag{2.7a}$$

$$\text{subject to: } \gamma_i(\mathbf{w}) \geq 1 - \xi_i \quad \forall i \in \{1, \ldots, n\} \tag{2.7b}$$

In both soft margin SVM formulations C is a trade-off between the norm of the weight vector $\mathbf{w}$, that controls the complexity of the classifier, and an upper bound on the classification error.

2.4 Sparseness

We will now derive equivalent quadratic programs to the hard-margin and the soft-margin quadratic programs, given in the previous section, for the case of binary classification. In binary classification the margin is defined as in Equation 2.3. Let us denote by α_i the Lagrange multiplier enforcing the margin constraint for example $(\mathbf{x}_i, y_i)$. Using standard Lagrangian duality techniques (see [3, Ch. 3]), one arrives to the following dual quadratic programs for the hard margin case

$$\max_{\boldsymbol{\alpha}} \; -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_i y_i \alpha_i \tag{2.8a}$$

$$\text{subject to:} \quad \alpha_i \geq 0 \quad \forall i \tag{2.8b}$$

the soft-margin with linear penalties

$$\max_{\boldsymbol{\alpha}} \; -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_i y_i \alpha_i \tag{2.9a}$$

$$\text{subject to:} \quad 0 \leq \alpha_i \leq C, \quad \forall i \tag{2.9b}$$

and the soft-margin with quadratic penalties

$$\max_{\boldsymbol{\alpha}} \; -\frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j \left[\langle \mathbf{x}_i, \mathbf{x}_j \rangle + \frac{n\delta_{ij}}{C} \right] + \sum_i y_i \alpha_i \tag{2.10a}$$

$$\text{subject to:} \quad \alpha_i \geq 0 \quad \forall i \tag{2.10b}$$

respectively, where

$$\delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.11}$$

The optimal weight vector w^* can be written as an expansion of the training examples

$$\mathbf{w}^* = \sum_i y_i \alpha_i \mathbf{x}_i \tag{2.12}$$

Lagrange multipliers were introduced in the Lagrangian dual derivation to enforce the margin constraints, and the Karush-Kahn-Tucker (KKT) complementary conditions (sufficient and necessary optimality conditions) for the hard margin case

$$\alpha_i(\gamma_i - 1) = 0 \quad \forall i \tag{2.13}$$

and the ones for the soft margin case

$$\alpha_i(\gamma_i - 1 + \xi_i) = 0 \quad \forall i \tag{2.14}$$

need to be satisfied at the optimal solution $\mathbf{w}^*$. According to the KKT conditions the training examples with non-zero Lagrange multipliers are the ones that either have a functional margin of one (support vectors), or violate the margin (outliers). Notice that only the training examples with non-zero Lagrange multipliers α_i, contribute to the optimal solution. Assuming that only a relatively small fraction of the training data will be support vectors or outliers, this will result in a sparse representation of the optimal weight vector, and efficient optimization algorithms that solve for the optimal weight vector $\mathbf{w}^*$.

2.5 Kernels

Linear discriminant functions are inherently too limited in terms of the type of functional relationships that they can learn [32]. To the left of Figure 2.4, for example, the class of circular examples are inside a circle and the square examples outside. This training data are obviously not separable by linear discriminant functions. Nevertheless they can be separated by a circle, that is, by linear discriminant functions that use second order features of the inputs, i.e., monomials of degree 2 (see Figure 2.4). It is in general useful to use mappings

$$\Phi : \mathcal{X} \rightarrow \mathcal{H} \tag{2.15}$$

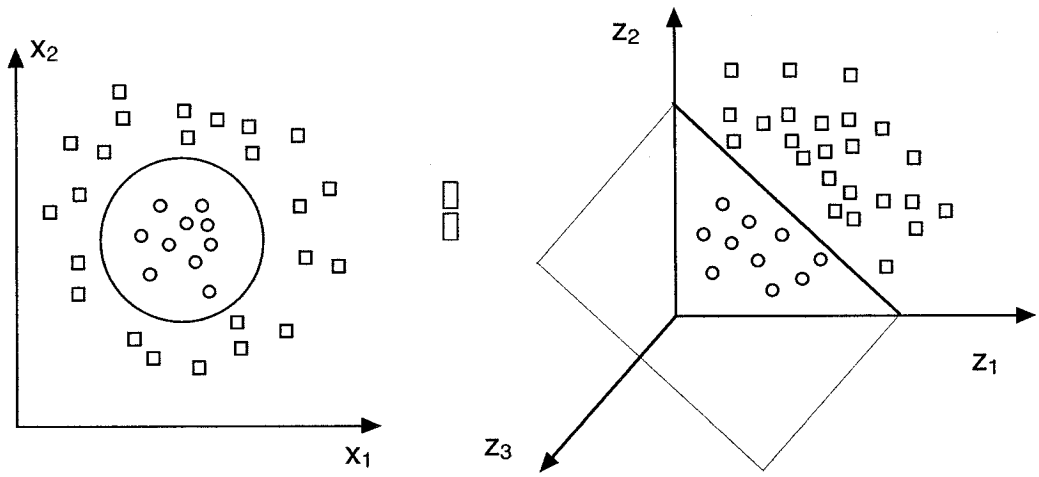


Figure 2.4: Non-linear kernels. The circular points cannot be separated from the squares with a linear function. But if we map the data to a higher dimensional space (e.g. $(z_1, z_2, z_3) = (x_1^2, x_2^2, c)$) they can separated using a hyperplane.

of the data from the input space $\mathcal{X}$ to a feature space $\mathcal{H}$ where the data can be separated linearly. This mapping Φ is called a feature map. However, this may not always be as efficient as in the above example, especially when the feature space need to be high-dimensional, for example, in the case where we would like to use monomials of degree much higher than two.

An important property of the dual formulations for support vector machines is that the data appear only in inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ between pairs of training examples (cf. the dual objective functions 2.8a, 2.9a, and 2.10a) . Thus, if we have a way to compute the inner product $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ efficiently, we can solve the problem linearly in the feature space without having to explicitly construct the feature map Φ. One just needs to replace the input space inner products $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ with the feature space inner products $\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ in the dual formulations.

A function $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ with the property that there exists a map Φ into an inner product space such that

$$K(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \tag{2.16}$$

is called a (positive definite) kernel. For a complete study of kernels and their applications the reader is referred to [41, 18]. Here we will briefly describe a couple

of commonly used kernels, and some basic properties of the kernels used to combine kernels.

Two popular kernels are the polynomial kernel

$$K(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + c)^d \tag{2.17}$$

where $d \in \mathbb{N}, c \geq 0$ and the Gaussian kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{2.18}$$

where $\sigma > 0$. Polynomial kernels correspond to mappings into the feature space of all possible d-th order monomials in input coordinates (as seen for $d = 2$ in the example earlier). Gaussian kernels correspond to mappings into (possible infinite) feature spaces, that whenever possible give rise to smooth discriminant functions, the smoothness depending on the parameter σ.

2.5.1 Properties of kernels

It is easy to see that the sum of kernels and the pointwise product of kernels are themselves kernels (cf. [41])

Proposition 1 *If K_1 and K_2 are kernels, and $\alpha_1, \alpha_2 \geq 0$, then $\alpha_1 K_1 + \alpha_2 K_2$ is a kernel.*

Proposition 2 *If K_1 and K_2 are kernels, then $K_1 K_2$, defined by $(K_1 K_2)(\mathbf{x}, \mathbf{x}') \equiv K_1(\mathbf{x}, \mathbf{x}') K_2(\mathbf{x}, \mathbf{x}')$, is a kernel.*

Similar properties hold for the tensor product and the direct sum of kernels, (possibly) defined on different domains.

Proposition 3 *If K_1 and K_2 are kernels defined respectively on $\mathcal{X}_1$ and $\mathcal{X}_2$, then their tensor product*

$$(K_1 \otimes K_2)((\mathbf{x}_1, \mathbf{x}_2)(\mathbf{x}_1', \mathbf{x}_2')) = K_1(\mathbf{x}_1, \mathbf{x}_1') K_2(\mathbf{x}_2, \mathbf{x}_2') \tag{2.19}$$

is a kernel on $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$.

Proposition 4 *If K_1 and K_2 are kernels defined respectively on $\mathcal{X}_1$ and $\mathcal{X}_2$, then their direct sum*

$$(K_1 \oplus K_2)((\mathbf{x}_1, \mathbf{x}_2)(\mathbf{x}'_1, \mathbf{x}'_2)) = K_1(\mathbf{x}_1, \mathbf{x}'_1) + K_2(\mathbf{x}_2, \mathbf{x}'_2) \tag{2.20}$$

is a kernel on $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$.

Based on the properties of sum and product of kernels, [17, 48] proposed the general family of convolution kernels. Convolution kernels involve a recursive computation over d components of structured inputs as follows

$$K(\mathbf{x}, \bar{\mathbf{x}}) = \sum_{\mathbf{x} \in R^{-1}(x), \bar{\mathbf{x}} \in R^{-1}(\bar{x})} \prod_{c=1}^{d} K_c(\mathbf{x}_c, \mathbf{y}_c) \tag{2.21}$$

Convolution kernels provide a method for constructing kernels for structure inputs like strings, trees and graphs (c.f. [9, 29]).

Kernels allow support vector machines to deal with complex input representations. However, the output space representation is simple, as that in binary classification. At this point, let's move from the binary classification to multiclass classification, and pave the way for more complex outputs in the sequel.

2.6 Multiclass Classification

In multiclass classification the output space is $\mathcal{Y} = \{1, \ldots, k\}$, and the solution involves several separating hyperplanes. There are basically two different approaches for multiclass classification. The first one is an indirect approach that combines several binary classifiers into a multiclass classifier. The second one involves a direct approach of learning the multiple hyperplanes simultaneously.

2.6.1 Indirect Multi-class Classification

One-Against-All

The first combination scheme involves learning k binary classifiers F_r $(r = 1, \ldots, k)$, one for each class, that discriminate between patterns of a particular class and patterns of any of the remaining classes. The decision function is given by

$$f(\mathbf{x}) = \underset{r=1,\ldots,k}{\operatorname{argmax}} F_r(\mathbf{x}) \tag{2.22}$$

Geometrically this amounts to associating a hyperplane with each class and assigning a new pattern $\mathbf{x}$ to the hyperplane that is furthest from it. The input space is split into k simply connected, convex regions. The main disadvantage of this scheme is that the training is performed independently for each classifier, and it may be hard to separate each class from the all the other $k - 1$ classes.

All-Against-All

The second scheme involves learning $\binom{k}{2}$ binary classifiers and employing some voting scheme to derive the decision function. This scheme has also the disadvantage that the training is performed independently for each classifier, but also the computational complexity of training $k(k-1)/2$ classifiers may be prohibitive.

Error Correcting Codes

An alternative scheme involves learning $m \leq \binom{k}{2}$ binary classifiers, each one for a different partition of the classes. Each classifier thus assigns a new pattern $\mathbf{x}$ to a set of classes (imposed by the specific partition), and the decision function selects the class that prevails in most partitions. Ideas from error correcting codes [14] can be used in the choice of the partitions in order to add some robustness to the decision function.

2.6.2 Direct Multi-class Classification

A more natural way to solve the multiclass classification problem is to construct a decision function by considering all classes simultaneously. One such scheme [50, 12] involves $r, r = 1, \ldots, k$ linear discriminant functions

$$F_r(\mathbf{x}) = \langle \mathbf{w}_r, \mathbf{x} \rangle \tag{2.23}$$

one for each class, and a decision function derived by the so-called winner-take-all rule

$$f(\mathbf{x}) = \underset{r=1,\ldots,k}{\operatorname{argmax}} F_r(\mathbf{x}) \tag{2.24}$$

that computes a separate score $F_r(\mathbf{x})$ for every class, and predicts the class with the highest score. In order to generalize SVMs to this multiclass setting, the margin of a training example $(\mathbf{x}_i, y_i)$ is defined as

$$\gamma_i = F_{y_i}(\mathbf{x}_i) - \max_{y \neq y_i} F_y(\mathbf{x}_i) \tag{2.25}$$

which is the difference between the score of the correct class and the highest score of all incorrect classes. This definition of the margin introduces a non-linear constraint for each training example in the quadratic program for SVMs, that can be expanded in $k - 1$ linear constraints of the form

$$F_{y_i}(\mathbf{x}_i) - F_y(\mathbf{x}) = \langle \mathbf{w}_{y_i} - \mathbf{w}_y, \mathbf{x} \rangle \geq 1 \quad \forall y \neq y_i \tag{2.26}$$

so that it indeed corresponds to a quadratic program with $n(k-1)$ constraints.

In order to allow margin violations, [50] proposed to introduce a slack variable $\xi_{(iy)}$ for each one of the $n(k-1)$ constraints leading to the following quadratic programs for linear penalties

$$\min_{\mathbf{w}_r} \quad \frac{1}{2} \sum_{r=1}^{k} \|\mathbf{w}_r\|^2 + \frac{C}{n(k-1)} \sum_{i=1}^{n} \sum_{y \neq y_i} \xi_{(iy)} \tag{2.27a}$$

$$\text{subject to:} \quad \langle \eta \mathbf{w}_{y_i} - \mathbf{w}_y, \mathbf{x}_i \rangle \geq 1 - \xi_{(iy)}, \ \xi_{(iy)} \geq 0 \quad \forall i \in \{1, \ldots, n\}, \forall y \neq y_i \tag{2.27b}$$

and the following quadratic program for quadratic penalties

$$\min_{\mathbf{w}_r} \quad \frac{1}{2}\sum_{r=1}^{k}\|\mathbf{w}_r\|^2 + \frac{C}{2n(k-1)}\sum_{i=1}^{n}\sum_{y\neq y_i}\xi_{(iy)}^2 \tag{2.28a}$$

$$\text{subject to:} \quad \langle \mathbf{w}_{y_i} - \mathbf{w}_y, \mathbf{x}_i\rangle \geq 1 - \xi_{(iy)} \quad \forall i \in \{1,\ldots,n\},\ \forall y \neq y_i \tag{2.28b}$$

The above approach, introducing a slack variable for every linear constraint, often leads to a hard to solve optimization problem, so [12] proposed to introduce a single slack variable ξ_i for every non-linear constraint, which results in a tighter upper bound on the empirical risk and offers additional algorithmic advantages. This leads to the following quadratic program for linear penalties

$$\min_{\mathbf{w}_r} \quad \frac{1}{2}\sum_{r=1}^{k}\|\mathbf{w}_r\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{2.29a}$$

$$\text{subject to:} \quad \langle \mathbf{w}_{y_i} - \mathbf{w}_y, \mathbf{x}_i\rangle \geq 1 - \xi_i \quad \forall i \in \{1,\ldots,n\},\ \forall y \neq y_i \tag{2.29b}$$

$$\xi_i \geq 0 \quad \forall i \in \{1,\ldots,n\} \tag{2.29c}$$

and the following quadratic program for quadratic penalties

$$\min_{\mathbf{w}_r} \quad \frac{1}{2}\sum_{r=1}^{k}\|\mathbf{w}_r\|^2 + \frac{C}{2n}\sum_{i=1}^{n}\xi_i^2 \tag{2.30a}$$

$$\text{subject to:} \quad \langle \mathbf{w}_{y_i} - \mathbf{w}_y, \mathbf{x}_i\rangle \geq 1 - \xi_i \quad \forall i \in \{1,\ldots,n\},\ \forall y \neq y_i \tag{2.30b}$$

Having presented multiclass support vector machines we are ready to proceed to the core of the thesis. In the next chapter we will present a generalization of multiclass support vector machines to the problem of learning for interdependent and structured outputs, motivated by the applications presented in Chapter 1.

Chapter 3

Joint Kernel Support Vector Machines

3.1 Introduction

In Chapter 1 we presented a series of applications in areas such as information retrieval, and natural language processing, where learning general functional dependencies between arbitrary input and output spaces is particularly relevant. Recent progress in machine learning has mainly focused on designing flexible and powerful input representations, facilitated by the development of kernel-based methods, a brief introduction of which we gave in Chapter 2. However, the complementary issue of designing classification algorithms that can deal with more complex outputs has been largely ignored.

In this chapter we first explain the main ideas of joint input-output feature maps, loss functions, and decoding, which are the necessary ingredients of the general framework, through an illustrative example. These depend on the structure of the problem and the assumed decomposition. Different structures, consequently decompositions, leading to different joint feature maps, loss functions, and decoding algorithms will be discussed in detail in Chapter 4 (see Tables 4.6, 4.7). Here we will focus on the general algorithms starting from simple a on-line algorithm, and building incrementally

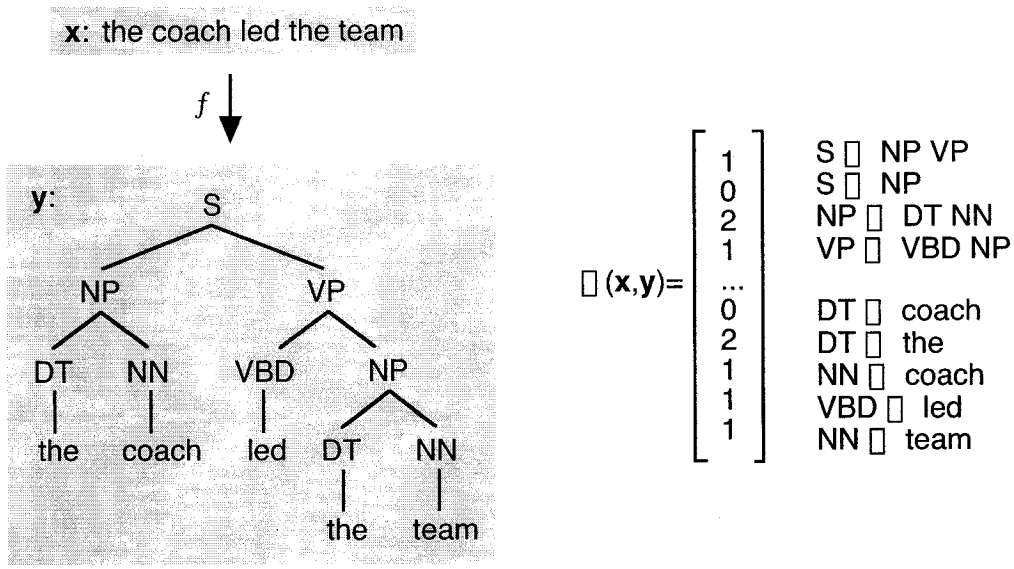


Figure 3.1: Illustration of a natural language parsing model. We want to learn a function f that maps a sentence x to its parse tree. To the right we show the computed input-output feature representation.

hard-margin SVMs, soft margin SVMs, and finally loss-sensitive SVMs. We will focus on a general cutting plane approach proposed to solve the above SVM problems, and discuss its convergence and sparseness properties.

3.2 Discriminant Function and Joint Feature Maps

We are interested in the general problem of learning functions $f : \mathcal{X} \rightarrow \mathcal{Y}$ between input spaces $\mathcal{X}$ and arbitrary discrete output spaces $\mathcal{Y}$ based on a training sample of input-output pairs. As an illustrating example, which we will continue to use as a prototypical application in the sequel, consider the case of natural language parsing, where the function f maps a given sentence $\mathbf{x}$ to a parse tree $\mathbf{y}$. This is depicted graphically in Figure 1.4.

The approach we pursue is to learn a *discriminant function* $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ over input/output pairs from which we can derive a prediction by maximizing F over the outputs for a specific given input $\mathbf{x}$. Hence, the general form of our function

(hypothesis) f is

$$f(\mathbf{x}) = \text{argmax}_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x}, \mathbf{y}; \mathbf{w}) \tag{3.1}$$

where $\mathbf{w}$ denotes a parameter vector. It might be useful to think of F as a compatibility function that measures how compatible pairs $(\mathbf{x}, \mathbf{y})$ are, or, alternatively, $-F$ can be thought of 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.

We assume F to be linear in some *combined feature representation* of inputs and outputs $\Psi(\mathbf{x}, \mathbf{y})$, i.e.

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle \tag{3.2}$$

The specific form of Ψ depends on the nature of the problem, and special cases will be discussed subsequently. However, in this chapter we present learning algorithms and theoretical results for the general case. Since we want to exploit the advantages of kernel-based method, we will pay special attention to cases where the inner product in the joint representation can be efficiently computed via a joint kernel function

$$J_\Psi((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}', \mathbf{y}') \rangle \,. \tag{3.3}$$

We generalize the notion of a separation margin, by defining the margin of a training example with respect to a discriminant as

$$\gamma_i = F(\mathbf{x}_i, \mathbf{y}_i) - \max_{\mathbf{y}\in\mathcal{Y}\setminus\mathbf{y}_i} F(\mathbf{x}_i, \mathbf{y}) \tag{3.4}$$

This corresponds to the separation margin employed by multiclass support vector machines (cf. Section 4.2) as proposed by [50, 12], generalized to the problem for learning interdependent and structured outputs. Nevertheless, there are cases where the number of outputs is extremely large, and some sort of decomposition of the feature map is required, to ensure that f can be computed from F efficiently, i.e. via a dynamic programing algorithm. Such decompositions have been proposed for problems with structured outputs in the context of perceptron learning [7], and support vector machine and boosting re-ranking algorithms [10, 8].

As pointed out in [8], we can think of the (linear) discriminant function F in terms of a sum of potentials, i.e. a log probability of an exponential family model, only that the global (or input specific) normalization Z is ignored. This implies that

$$F(\mathbf{x},\mathbf{y}) - F(\mathbf{x},\bar{\mathbf{y}}) = \log \frac{P(\mathbf{y}|\mathbf{x})}{P(\bar{\mathbf{y}}|\mathbf{x})}, \quad \text{where} \quad P(\mathbf{y}|\mathbf{x}) = \frac{\exp[\langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{y})\rangle]}{Z(\mathbf{x})} \tag{3.5}$$

Maximum likelihood estimation aims at maximizing the average logarithm of the conditional probability of the correct output given the input [24]. In contrast, the proposed maximum margin approach aims at maximizing the difference of log probabilities between the correct output and the highest ranked incorrect one(s). While the former requires to take $\sum_{\mathbf{y}} \exp[\langle \mathbf{w}, \Psi(\mathbf{x},\mathbf{y})\rangle]$ into account, the latter does not.

We will discuss two learning approaches for the joint input-output kernel setting: (i) An on-line algorithm in the spirit of the perceptron algorithm and (ii) a maximum margin formulation generalizing support vector machines.

3.3 Perceptron Learning

We will first focus on an on-line learning approach, which generalizes perceptron learning [39], and was first proposed in the context natural language processing in [10].

In a nutshell, the perceptron algorithm works as follows. In an on-line fashion, training inputs $\mathbf{x}_i$ are presented. Then the function $f(\mathbf{x}_i)$ is computed, which produces the highest scored output $\hat{\mathbf{y}}$. If the predicted label sequence is correct $\hat{\mathbf{y}} = \mathbf{y}_i$, no update is performed. Otherwise, the weight vector is updated based on the difference vector $\delta\Psi_i(\hat{\mathbf{y}}) = \Psi(\mathbf{x}_i,\mathbf{y}_i) - \Psi(\mathbf{x}_i,\hat{\mathbf{y}})$, namely $\mathbf{w}^{\text{new}} \leftarrow \mathbf{w}^{\text{old}} + \delta\Psi_i(\hat{\mathbf{y}})$. This scheme can be proven to converge under a standard separability assumption.

In order to avoid an explicit evaluation of the feature map as well as a direct (i.e. primal) representation of the discriminant function, we would like to derive an equivalent dual formulation of the perceptron algorithm. Notice that in the standard perceptron learning case, $\Psi(\mathbf{x},1) = -\Psi(\mathbf{x},-1)$, so it is sufficient to store only those training inputs that have been used during a weight update. In the generalized perceptron algorithm one also needs to store the incorrectly predicted (decoded) outputs

Algorithm 1 Dual perceptron for learning via joint feature maps.

input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$
output: $\boldsymbol{\alpha}$
initialize all $\alpha_{(i\mathbf{y})} = 0$
repeat
  for all training patterns $\mathbf{x}_i$ **do**
    / decode */*
    compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{x}_i, \mathbf{y})$,
    where $F(\mathbf{x}_i, \mathbf{y}) = \sum_{j\bar{\mathbf{y}}} \alpha_{(j\bar{\mathbf{y}})} \langle \delta\Psi_j(\hat{\mathbf{y}}), \Psi(\mathbf{x}_i, \mathbf{y}) \rangle$
    if $\hat{\mathbf{y}} \neq \mathbf{y}_i$ **then**
      / update */*
      $\alpha_{(i\hat{\mathbf{y}})} \leftarrow \alpha_{(i\hat{\mathbf{y}})} + 1$
    end if
  end for
until no more errors

$f(\mathbf{x}_i)$. More precisely, one only needs to store how the decoded $f(\mathbf{x}_i)$ differs from the correct $\mathbf{y}_i$, which typically results in a more compact representation.

The dual formulation of the discriminant function is as follows. One maintains a set of dual parameters $\alpha_{(i\mathbf{y})}$ such that

$$F(\mathbf{x}, \mathbf{y}) = \sum_{i,\bar{\mathbf{y}}} \alpha_{(i\bar{\mathbf{y}})} \langle \delta\Psi_i(\bar{\mathbf{y}}), \Psi(\mathbf{x}, \mathbf{y}) \rangle \tag{3.6}$$

Once an update is necessary for training example $(\mathbf{x}_i, \mathbf{y}_i)$ and incorrect output $\hat{\mathbf{y}}$, one simply increments $\alpha_{(i\hat{\mathbf{y}})}$. Of course, as a practical matter of implementation, one will only represent the non-zero $\alpha_{(i\mathbf{y})}$. Notice that this requires to keep track of the α values themselves as well as the pairs $(\mathbf{x}_i, \mathbf{y})$ for which $\alpha_{(i\mathbf{y})} > 0$.

The above formulation is valid for any joint feature map Ψ and can be generalized to arbitrary joint kernel functions J_Ψ by replacing the inner product with the corresponding values of J_Ψ.

In order to prove the convergence of this algorithm, it suffices to apply Theorem 1 in [10], which is a simple generalization of Novikoff's theorem [34].

Theorem 1 *Assume a training set* $\{(\mathbf{x}_i, \mathbf{y}_i)\}$, $i = 1, \ldots, n$, *and for each training output a set of candidate outputs* $\mathcal{Y}_i \subseteq \mathcal{Y} \setminus \mathbf{y}_i$. *If there exists a weight vector* $\mathbf{w}$ *such*

that $\|\mathbf{w}\| = 1$ *and*

$$\langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}_i) \rangle - \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle \geq \gamma, \quad \textit{for all } \mathbf{y} \in \mathcal{Y}_i \tag{3.7}$$

then the number of update steps performed by the above perceptron algorithm is bounded from above by

$$\frac{R^2}{\gamma^2} \tag{3.8}$$

where $R = \max_i \|\Psi(\mathbf{x}_i, \mathbf{y})\|$ *for* $\mathbf{y} \in \mathcal{Y}_i$.

3.4 Hard-Margin Support Vector Machines

Here, we consider the case where there exists a function f parameterized by $\mathbf{w}$ such that the empirical risk (the error on the training data) is zero. The condition of zero training error can then be compactly written as a set of nonlinear constraints

$$\forall i \in \{1, \ldots, n\} : \quad \max_{\mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i} \{\langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle\} \leq \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}_i) \rangle \tag{3.9}$$

Every one of the nonlinear inequalities in Eq. (3.9) can be equivalently replaced by $|\mathcal{Y}| - 1$ linear inequalities, resulting in a total of $n|\mathcal{Y}| - n$ linear constraints,

$$\forall i \in \{1, \ldots, n\}, \ \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \quad \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}_i) - \Psi(\mathbf{x}_i, \mathbf{y}) \rangle \geq 0 \tag{3.10}$$

If the set of inequalities in Eq. (3.10) is feasible, there will typically be more than one solution $\mathbf{w}$. To specify a unique solution, we select the $\mathbf{w}$ for which the score of the correct output $\mathbf{y}_i$ is uniformly most different from the closest runner-up $\hat{\mathbf{y}}_i(\mathbf{w}) = \text{argmax}_{\mathbf{y} \neq \mathbf{y}_i} \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle$. This generalizes the maximum-margin principle employed in Support Vector Machines [47] to the more general case considered in this thesis. Denoting the margin by γ and restricting the L_2 norm of $\mathbf{w}$ to make the problem well-posed, this leads to the following optimization problem.

$$\max_{\gamma, \mathbf{w}: \|\mathbf{w}\| = 1} \gamma \tag{3.11a}$$

$$\text{s.t. } \forall i \in \{1, \ldots, n\}, \ \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \quad \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq \gamma \tag{3.11b}$$

This problem can be equivalently expressed as a convex quadratic program

$$\text{SVM}_0: \quad \min_{\mathbf{w}} \ \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.12a}$$

$$\text{s.t. } \forall i,\ \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \ \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq 1 \tag{3.12b}$$

The key challenge is that the size of these problems can be immense, since we have to deal with $n|\mathcal{Y}| - n$ margin inequalities. In many cases, $|\mathcal{Y}|$ may be extremely large, in particular, if $\mathcal{Y}$ is a product space of some sort (e.g. in parsing), its cardinality may grow exponentially in the description length of $\mathbf{y}$. This makes standard quadratic programming solvers unsuitable for this type of problem.

In the following, we will 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. We will show that the algorithm can compute arbitrary close approximations to SVMs in polynomial time for a large range of structures. Since the algorithm operates on the dual program, we will first derive the Wolfe dual.

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 $(\mathbf{x}_i, \mathbf{y}_i)$. Again, using standard Lagragian duality techniques, one arrives at the following dual quadratic program (QP).

Proposition 5 *The objective of the dual problem of SVM_0 from Eq.* (3.12) *is given by*

$$\Theta(\boldsymbol{\alpha}) \equiv -\frac{1}{2}\sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \sum_j \sum_{\bar{\mathbf{y}}\neq\mathbf{y}_j} \alpha_{(i\mathbf{y})}\alpha_{(j\bar{\mathbf{y}})} J_{\delta\Psi}((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \bar{\mathbf{y}})) + \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})} \tag{3.13}$$

where $J((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \bar{\mathbf{y}})) = \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \rangle$. The dual QP can be formulated as

$$\boldsymbol{\alpha}^* = argmax_{\boldsymbol{\alpha}} \Theta(\boldsymbol{\alpha}), \quad \text{subject to: } \boldsymbol{\alpha} \geq 0 \tag{3.14}$$

Proof *The primal Lagrangian of the problem SVM_0 is*

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})}[\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle - 1] \tag{3.15}$$

where $\boldsymbol{\alpha} \geq 0$ are the Lagrange mulipliers. The corresponding dual is found by differentiating with respect to $\mathbf{w}$, imposing stationarity

$$\frac{\partial L(\mathbf{w}, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \delta\Psi_i(\mathbf{y}) = 0 \tag{3.16}$$

and re-substituting this relation into the primal we obtain the dual objective

$$L(\mathbf{w}, \boldsymbol{\alpha}) = \sum_i \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{(i\mathbf{y})} \tag{3.17}$$

$$-\frac{1}{2} \sum_i \sum_{\mathbf{y} \neq \mathbf{y}_i} \sum_j \sum_{\bar{\mathbf{y}} \neq \mathbf{y}_i} \alpha_{(i\mathbf{y})} \alpha_{(j\bar{\mathbf{y}})} \left\langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \right\rangle \tag{3.18}$$

■

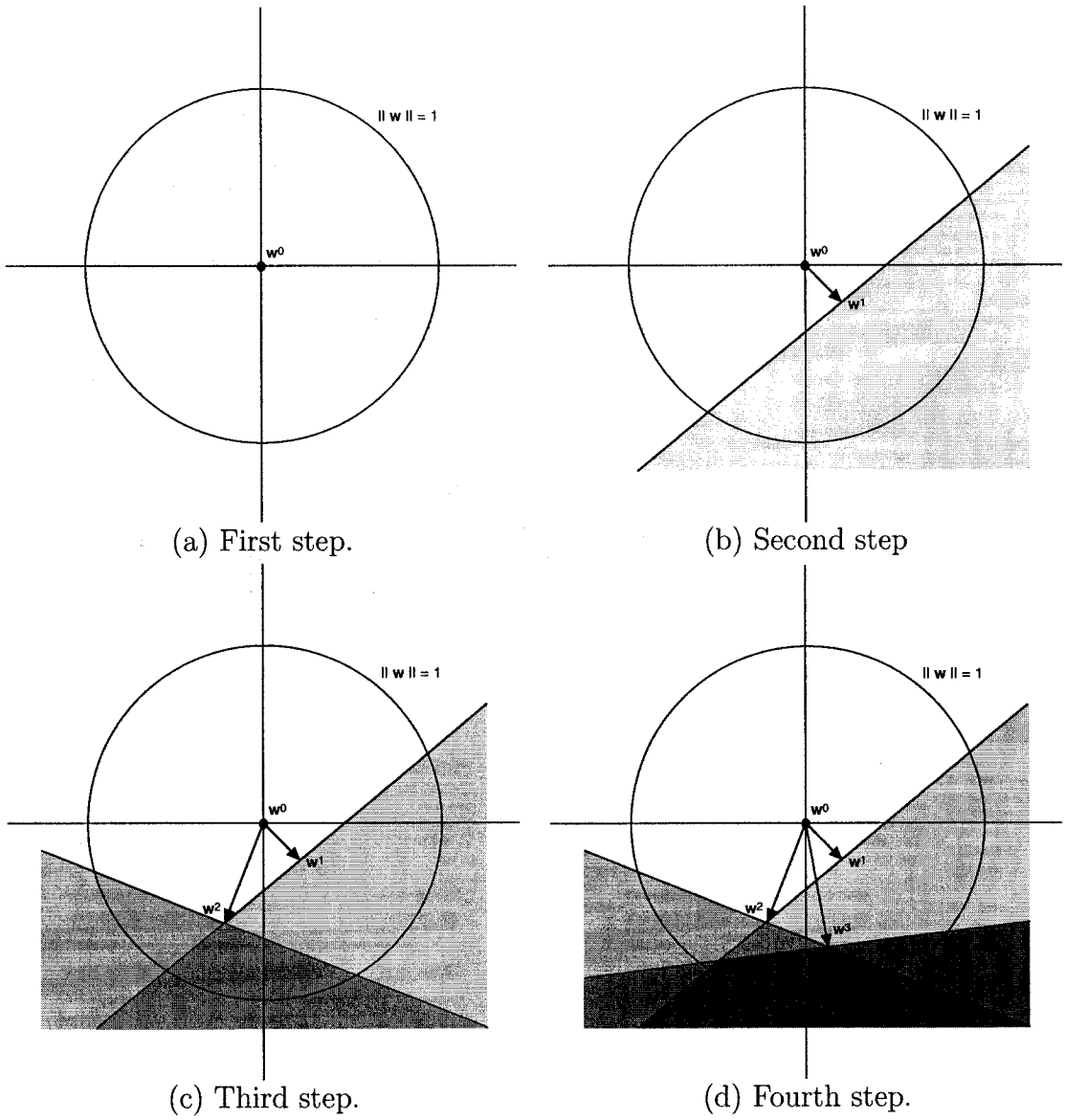
Notice that the $J_{\delta\Psi}$ function that generates the quadratic form in the dual objective can be computed from inner products involving values of Ψ, which is a simple consequence of the linearity of the inner product. $J_{\delta\Psi}$ can hence be alternatively computed from a joint kernel function J_Ψ over $\mathcal{X} \times \mathcal{Y}$.

3.5 Cutting Plane Algorithm

The algorithm we propose aims at finding a small set of constraints from the full-sized optimization problem that ensures a sufficiently accurate solution. More precisely, we will construct a nested sequence of successively tighter relaxations of the original problem using a cutting plane method [23], implemented as a variable selection approach in the dual formulation. We will later show that this is a valid strategy, since there always exists a polynomially sized subset of constraints so that the solution of the relaxed problem defined by this subset fulfills all constraints from the full optimization problem up to a precision of ϵ. This means that the remaining – potentially exponentially many – constraints are guaranteed to be violated by no more than ϵ, without the need for explicitly adding these constraints to the optimization problem.

3.5.1 Algorithm

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 supports a natural problem decomposition.



(a) First step. (b) Second step

(c) Third step. (d) Fourth step.

Figure 3.2: Cutting plane algorithm. Successive steps of the cutting plane algorithm. (a) First step: No constraints have been added. $\mathbf{w}^0$ is the current solution. (b) Second step: The (potentially) most violated constraint has been added. It cuts off the current solution $\mathbf{w}^0$ from the feasible region (shaded).(c) Third step: One more violated constraint is added, and the new solution is computed. (d) Fourth step: The process is repeated until there are no more violating constraints.

More specifically, notice that the constraint martrix derived for SVM_0 is diagonal, since the non-negativity constraints involve only a single α-variable at a time.

Pseudocode of the algorithm is depicted in Algorithm 2. 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 proceeds by finding the (potentially) "most violated" constraint, involving some output value $\hat{\mathbf{y}}$. If the margin violation of this constraint is more than ϵ, the dual variable corresponding to $\hat{\mathbf{y}}$ is added to the working set. 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 violated constraint. Once a constraint has been added, the solution is re-computed with respect to S (see Figure 3.2). 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 constraint matrix, which implies that variables $\alpha_{(j\bar{\mathbf{y}})}$ with $j \neq i$, $\bar{\mathbf{y}} \in S_j$ 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. With respect to the optimization in step 9, we would like to point out that in some applications the constraint selection in step 7 may be more expensive than solving the relaxed QP. Hence it may be advantageous to solve the full relaxed QP in every iteration, instead of just optimizing over a subspace of the dual variables.

A convenient property of both algorithms is that they have a very general and well-defined interface independent of the choice of Ψ. 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), as well as the maximization in step 7. All of those, in particular the constraint/cut selection method, are treated as black boxes. While the modeling of $\Psi(\mathbf{x}, \mathbf{y})$ is typically straightforward, solving the maximization problem for constraint selection typically requires exploiting the structure of Ψ. We need to identify the

Algorithm 2 Hard-margin support vector machines (SVM_0).

input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, ϵ
output: $\boldsymbol{\alpha}$
$S_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
repeat
  for $i = 1, \ldots, n$ **do**
    / prepare cost function for optimization */*
    set up cost function
    $H(\mathbf{y}) \equiv 1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle$ where $\mathbf{w} \equiv \sum_j \sum_{\mathbf{y}' \in S_j} \alpha_{(j\mathbf{y}')} \delta\Psi_j(\mathbf{y}')$
    / find cutting plane */*
    compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$
    if $H(\hat{\mathbf{y}}) > \epsilon$ **then**
      / add constraint to the working set */*
      $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
      / Variant (a): perform full optimization */*
      $\boldsymbol{\alpha}_S \leftarrow$ optimize the dual over S, $S = \cup_i S_i$
      / Variant (b): perform subspace ascent */*
      $\boldsymbol{\alpha}_{S_i} \leftarrow$ optimize the dual over S_i
    end if
  end for
until no S_i has changed during iteration

highest scoring $\mathbf{y}$ that is incorrect,

$$\hat{\mathbf{y}} \equiv \text{argmax}_{\mathbf{y} \neq \mathbf{y}_i} \{1 - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle\} \tag{3.19}$$

It is therefore sufficient to identify the best solution $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle$ as well as the second best solution $\tilde{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y} \setminus \hat{\mathbf{y}}} \langle \mathbf{w}, \Psi(\mathbf{x}_i, \mathbf{y}) \rangle$. The second best solution is necessary to detect margin violations in cases where $\hat{\mathbf{y}} = \mathbf{y}_i$, but $\langle \mathbf{w}, \delta\Psi_i(\tilde{\mathbf{y}}) \rangle < 1$. This means that for all problems where we can solve the inference problem in Eq. (3.1) for the top two $\mathbf{y}$, we can also apply our learning algorithms with the zero-one loss. In the case of parsing, for example, we can use CKY algorithm that returns the two highest scoring parses of a sequence.

We will now proceed by analyzing the presented algorithm. In particular, we will show correctness and sparse approximation properties, as well as bounds on the runtime complexity.

3.5.2 Analysis

What we would like to accomplish first is to obtain a lower bound on the achievable improvement of the dual objective by selecting a single variable $\alpha_{(i\tilde{y})}$ and adding it to the dual problem (cf. step 9 in Algorithm 2). In order to derive useful bounds, it suffices to restrict attention to simple one-dimensional families of solutions that are defined by improving an existing solution along a specific direction $\boldsymbol{\eta}$. By proving that one can make sufficient progress along a specific direction, this clearly implies that one can make at least that much progress as by optimizing over a larger subspace that includes the direction $\boldsymbol{\eta}$. A first step towards excecuting this idea is the following lemma.

Lemma 1 *Let $\boldsymbol{J}$ be a symmetric, positive semi-definite matrix, and define a concave objective in $\boldsymbol{\alpha}$*

$$\Theta(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}'\boldsymbol{J}\boldsymbol{\alpha} + \langle \mathbf{h}, \boldsymbol{\alpha} \rangle \tag{3.20}$$

which we assume to be bounded from above. Assume that a solution $\boldsymbol{\alpha}^t$ and an optimization direction $\boldsymbol{\eta}$ is given such that $\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle > 0$. Then optimizing Θ starting from $\boldsymbol{\alpha}^t$ along the chosen direction $\boldsymbol{\eta}$ will increase the objective by

$$\max_{\beta>0} \left\{ \Theta(\boldsymbol{\alpha}^t + \beta\boldsymbol{\eta}) \right\} - \Theta(\boldsymbol{\alpha}^t) = \frac{1}{2}\frac{\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle^2}{\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta}} > 0 \tag{3.21}$$

Proof *The difference obtained by a particular β is given by*

$$\delta\Theta(\beta) \equiv \beta \left[\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle - \frac{\beta}{2}\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta} \right] \tag{3.22}$$

as can be verified by elementary algebra. Solving for β one arrives at

$$\frac{d}{d\beta}\delta\Theta = 0 \iff \beta^* = \frac{\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle}{\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta}} \tag{3.23}$$

Notice that this requires $\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta} > 0$. Obviously, the positive semi-definiteness of $\boldsymbol{J}$ guarantees $\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta} \geq 0$ for any $\boldsymbol{\eta}$. Moreover $\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta} = 0$ together with $\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle > 0$ would imply that $\lim_{\beta\to\infty} \Theta(\boldsymbol{\alpha}^t + \beta\boldsymbol{\eta}) = \infty$, which is in contraction with the assumption that Θ is bounded. Plugging the value for β^* back into the above expression for $\delta\Theta$ yields the claim. ■

Corollary 1 *Under the same assumption as in Lemma 1 and for the special case of an optimization direction* $\boldsymbol{\eta} = \boldsymbol{e}_r$, *the objective improves by*

$$\delta\Theta(\beta^*) = \frac{1}{2J_{rr}}\left(\frac{\partial\Theta}{\partial\alpha_r}\right)^2 > 0 \tag{3.24}$$

Proof *Notice that* $\boldsymbol{\eta} = \boldsymbol{e}_r$ *implies* $\langle\nabla\Theta, \boldsymbol{\eta}\rangle = \frac{\partial\Theta}{\partial\alpha_r}$ *and* $\boldsymbol{\eta}'\boldsymbol{J}\boldsymbol{\eta} = J_{rr}$. ■

We now apply the above lemma and corollary to get the following proposition:

Proposition 6 *For step 9 in Algorithm 2 the improvement* $\delta\Theta$ *of the dual objective is lower bounded by*

$$\delta\Theta \geq \frac{\epsilon^2}{2R_i^2}, \quad \textit{where} \quad R_i \equiv \max_{\mathbf{y}} \|\delta\Psi_i(\mathbf{y})\| \tag{3.25}$$

Proof *Using the notation in Algorithm 2 one can apply Corollary 1 with multi-index* $r = (i\hat{\mathbf{y}})$, $\boldsymbol{h} = \mathbf{1}$, *and* $\boldsymbol{J}$ *such that*

$$J_{(i\hat{\mathbf{y}})(j\mathbf{y})} = \langle\delta\Psi_i(\hat{\mathbf{y}}), \delta\Psi_j(\mathbf{y})\rangle \tag{3.26}$$

Notice that the partial derivative of Θ *with respect to* $\alpha_{(i\hat{\mathbf{y}})}$ *is given by*

$$\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\boldsymbol{\alpha}^t) = 1 - \sum_{(j\mathbf{y})}\alpha_{(j\mathbf{y})}J_{(i\hat{\mathbf{y}})(j\mathbf{y})} = 1 - \langle\mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}})\rangle, \tag{3.27}$$

since the optimality equations for the primal variables yield the identities

$$\mathbf{w}^* = \sum_{(j\mathbf{y})}\alpha_{(j\mathbf{y})}\delta\Psi_j(\mathbf{y}) \tag{3.28}$$

Now, applying the condition of step 9, namely

$$(1 - \langle\mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}})\rangle) > \epsilon \tag{3.29}$$

leads to the bound

$$\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\boldsymbol{\alpha}^t) \geq \epsilon \tag{3.30}$$

Finally, $J_{rr} = \|\delta\Psi_i(\hat{\mathbf{y}})\|^2$ *and inserting this expression and the previous bound into the expression from Corollary 1 yields*

$$\frac{1}{2J_{rr}}\left(\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}\right)^2 \geq \frac{\epsilon^2}{2\|\delta\Psi_i(\hat{\mathbf{y}})\|^2} \geq \frac{\epsilon^2}{2R_i^2} \tag{3.31}$$

The claim follows by observing that jointly optimizing over a set of variables that include α_r *can only further increase the value of the dual objective.* ■

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

Theorem 2 *With* $R \equiv \max_{i,\mathbf{y}} \|\delta\Psi_i(\mathbf{y})\|$, $\gamma > 0$ *the geometric margin of a separable training set, and for a given* $\epsilon > 0$, *Algorithm 2 terminates after incrementally adding at most*

$$\frac{R^2}{\gamma^2\epsilon^2} \tag{3.32}$$

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 9, the objective will increase by at least the amount suggested by Proposition 6. 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 is* $\frac{1}{2\gamma^2}$. ■

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 7 can be computed in polynomial time, the overall algorithm has a runtime polynomial in $n, \bar{R}, 1/\epsilon$, since at least one constraint will be added while cycling through all n instances and since step 9 is polynomial. This shows that the algorithm considers only a small number of constraints, if one allows an extra ϵ slack, and that the solution depends on the precision parameter ϵ. The upper bound on the number of active constraints in such an approximate solution depends on the chosen representation, more specifically, we need to upper bound the difference vectors $\|\Psi_i(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \bar{\mathbf{y}})\|^2$ for arbitrary $\mathbf{y}, \bar{\mathbf{y}} \in \mathcal{Y}$. In the following chapter, we will thus make sure that suitable upper bounds are available.

3.5.3 Discussion

The nested sequence of successively tighter relaxations of the original problem created by the proposed cutting plane algorithm is reminiscent of the chunking technique [4], shown to converge to the optimal solution of the original problem by [36]. Chunking starts with an arbitrary subset of the constraints (working set) and uses a generic quadratic programming solver on that relaxed subproblem. Adding constraints that violate the current solution, the set of constraints grows until in the last iteration it contains all the active constraints of the original problem, that is, the support vectors. For joint kernel support vector learning it may be beneficial to use a specialized support vector machine learner (i.e. SVM^{light} [20]) for solving the increasing in size relaxed problems, rather than a generic quadratic programming solver.

Alternatively, the use of decomposition methods [35, 37, 20], where the working set size is smaller than the number of the active constraints, may be favorable, especially since one can exploit the block diagonal structure of the optimization problem and perform the optimization at every iteration on the subspace defined by a single training example. However, even though decomposition methods lead to a strict improvement of the objective function, available convergence proofs [28] depend on the selection of the working set. More flexible working set selection schemes, such as sequential or random subspace selection, that are computational appropriate in the case of joint kernel support vector machines, may not lead to convergence, and [28] suggests that it may not be an easy task to prove more generalized convergence without the properties of a systematic working set selection.

At the extreme of decomposition methods there is an algorithm for joint kernel support vector learning, similar to sequential minimal optimization (SMO) [37], where the size of the working set is 1. It exploits the fact that once a constraint violated by the current solution has been identified there is an analytical solution of the optimal update of the corresponding lagrange multiplier.

3.6 Soft-Margin Support Vector Machines

To allow errors in the training set, we introduce slack variables and propose to optimize a soft-margin criterion. As in the case of multiclass SVMs, there are at least two ways of introducing slack variables [50, 12]. One may introduce a single slack variable ξ_i for violations of the *nonlinear* constraints (i.e. every instance $\mathbf{x}_i$) [12] or one may penalize margin violations for every *linear* constraint (i.e. every instance $\mathbf{x}_i$ and output $\mathbf{y} \neq \mathbf{y}_i$) [50]. Since the former results in a tighter upper bound on the empirical risk and offers some advantages in the proposed optimization scheme, we have focused on this formulation, which generalizes the multiclass SVM of [12]. Adding a penalty term that is linear in the slack variables to the objective results in the quadratic program

$$\text{SVM}_1: \quad \min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{3.33a}$$

$$\text{s.t. } \forall i,\ \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i: \ \langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle \geq 1 - \xi_i,\ \xi_i \geq 0 \tag{3.33b}$$

Alternatively, we can also penalize margin violations by a quadratic term leading to the following optimization problem:

$$\text{SVM}_2: \quad \min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2n}\sum_{i=1}^{n}\xi_i^2 \tag{3.34a}$$

$$\text{s.t. } \forall i,\ \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i: \ \langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle \geq 1 - \xi_i \tag{3.34b}$$

In both cases, $C > 0$ is a constants that controls the trade-off between training error minimization and margin maximization. Both optimization problems correspond to minimizing an upper bound on the zero-one loss, since $\frac{1}{n}\sum_{i=1}^{n}\xi_i$ and $\frac{1}{n}\sum_{i=1}^{n}\xi_i^2$ are upper bounds on the empirical risk for this loss function.

In the respective dual problems for the soft margin formulations, linear penalties introduce additional (box) constraints, whereas the squared penalties modify the kernel function.

Proposition 7 *The dual problem to SVM_1 is given by the program in Proposition 3.14 with additional box constraints*

$$\sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})} \leq \frac{C}{n}, \quad \forall i = 1, \ldots, n \tag{3.35}$$

Proof *Appendix A.* ■

Proposition 8 *The dual problem to SVM_2 is given by the program in Proposition 3.14 with modified kernel function*

$$J((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \bar{\mathbf{y}})) = \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \rangle + \frac{n\delta_{ij}}{C} \tag{3.36}$$

Proof *Appendix A.* ■

The soft-margin SVM formulation is a special case of the loss-sensitive SVM formulation we describe in the following section. One just needs to replace the arbitrary loss function $\triangle$ with the $0-1$ loss function to arrive from loss-sensitive SVMs to soft-margin SVMs. Therefore, we will skip the presentation of the soft-margin SVMs algorithm and analysis, and present the more general loss sensitive SVM algorithm and analysis in the following section.

3.7 Loss-Sensitive Support Vector Machines

The standard zero-one loss function typically used in classification is not appropriate for most kinds of structured outputs. For example, in natural language parsing, a parse tree that is almost correct and differs from the correct parse tree in only a few nodes should be treated differently from a parse that is completely different. Typically, the correctness of a predicted parse tree is measured by its F_1 score (see [22]), the harmonic mean of precision and recall as calculated based on the overlap of nodes between the trees. To be able to model such crucial distinctions, we propose two approaches that generalize the above formulations to the case of arbitrary loss functions $\triangle$.

3.7.1 Slack Re-scaling

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 severley than a violation involving an output value with smaller loss. This can be accomplished by multiplying the margin violation by the loss, or equivalently, by scaling the slack variable with the inverse loss, which yields

$$\text{SVM}_1^{\triangle s}: \qquad \min_{\mathbf{w},\boldsymbol{\xi}} \ \frac{1}{2}\langle \mathbf{w}, \mathbf{w} \rangle + \frac{C}{n}\sum_{i=1}^{n} \xi_i \tag{3.37a}$$

$$\text{s.t. } \forall i, \forall \mathbf{y} \in \mathcal{Y} \setminus \mathbf{y}_i : \ \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq 1 - \frac{\xi_i}{\triangle(\mathbf{y}_i, \mathbf{y})} \tag{3.37b}$$

A justification for this formulation is given by the subsequent proposition.

Proposition 9 *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* $\mathcal{R}_S^{\triangle}(\mathbf{w}^*)$.

Proof *Notice first that* $\xi_i^* = \max\{0, \max_{\mathbf{y} \neq \mathbf{y}_i}\{\triangle(\mathbf{y}_i, \mathbf{y})\,(1 - \langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y}) \rangle)\}\}$.

Case 1: If $f(\mathbf{x}_i; \mathbf{w}^*) = \mathbf{y}_i$ *then* $\xi_i^* \geq 0 = \triangle(\mathbf{y}_i, f(\mathbf{x}_i; \mathbf{w}))$ *and the loss is trivially upper bounded.*

Case 2: If $\hat{\mathbf{y}} \equiv f(\mathbf{x}_i; \mathbf{w}^*) \neq \mathbf{y}_i$, *then* $\langle \mathbf{w}, \delta\Psi_i(\hat{\mathbf{y}}) \rangle \leq 0$ *and thus* $\frac{\xi_i^*}{\triangle(\mathbf{y}_i, \mathbf{y})} \geq 1$ *which is equivalent to* $\xi_i^* \geq \triangle(\mathbf{y}_i, \mathbf{y})$.

Since the bound holds for every training instance, it also holds for the average. ■

The optimization problem $\text{SVM}_2^{\triangle s}$ 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.

In the loss-sensitive case with slack re-scaling, the loss function is introduced in the box constraints for L_1 penalties and in the kernel function for L_2 penalties.

Proposition 10 *The dual problem to* $SVM_1^{\triangle s}$ *is given by the program in Proposition 3.14 with additional box constraints*

$$\sum_{\mathbf{y} \neq \mathbf{y}_i} \frac{\alpha_{(i\mathbf{y})}}{\triangle(\mathbf{y}_i, \mathbf{y})} \leq \frac{C}{n}, \quad \forall i = 1, \ldots, n \tag{3.38}$$

Proof *Appendix A.* ■

Proposition 11 *The dual problem to* $SVM_2^{\triangle s}$ *is given by the program in Proposition 3.14 with modified kernel function*

$$J((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \bar{\mathbf{y}})) = \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \rangle + \frac{n\delta_{ij}}{C\sqrt{\triangle(\mathbf{y}_i, \mathbf{y})}\sqrt{\triangle(\mathbf{y}_j, \bar{\mathbf{y}})}} \tag{3.39}$$

Proof *Appendix A.* ■

3.7.2 Margin Re-scaling

In addition to this *slack re-scaling* approach, a second way to include loss functions is to re-scale the margin as proposed by [45] for the special case of the Hamming loss. It is straightforward to generalize this method to general loss functions. The margin constraints in this setting take the following form:

$$\forall i,\ \forall \mathbf{y} \in \mathcal{Y}: \quad \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \geq \triangle(\mathbf{y}_i, \mathbf{y}) - \xi_i \tag{3.40}$$

The set of constraints in Eq. (3.40) combined with the objective in Eq. (3.33a) yield an optimization problem $\text{SVM}_1^{\triangle m}$ which also results in an upper bound on $\mathcal{R}_{\mathcal{S}}^{\triangle}(\mathbf{w}^*)$.

Proposition 12 *Denote by* $(\mathbf{w}^*, \xi^*)$ *the optimal solution to* $SVM_1^{\triangle m}$. *Then* $\frac{1}{n}\sum_{i=1}^n \xi_i^*$ *is an upper bound on the empirical risk* $\mathcal{R}_{\mathcal{S}}^{\triangle}(\mathbf{w}^*)$.

Proof *The essential observation is that* $\xi_i^* = \max\{0, \max_{\mathbf{y}}\{\triangle(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y}) \rangle\}\}$ *which is guaranteed to upper bound* $\triangle(\mathbf{y}_i, \mathbf{y})$ *for* $\mathbf{y}$ *such that* $\langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y}) \rangle \leq 0$. ■

The optimization problem $\text{SVM}_2^{\triangle m}$ can be derived analogously, where $\triangle(\mathbf{y}_i, \mathbf{y})$ is replaced by $\sqrt{\triangle(\mathbf{y}_i, \mathbf{y})}$.

In the non-separable case with margin re-scaling, the loss function is introduced in the linear part of the objective function.

Proposition 13 *The dual problems to* $SVM_1^{\triangle m}$ *and* $SVM_2^{\triangle m}$ *are given by the dual problems to* SVM_1 *and* SVM_2 *respectively with objective*

$$\Theta(\boldsymbol{\alpha}) \equiv -\frac{1}{2}\sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \sum_j \sum_{\bar{\mathbf{y}}\neq\mathbf{y}_j} \alpha_{(i\mathbf{y})}\alpha_{(j\bar{\mathbf{y}})} J((\mathbf{x}_i, \mathbf{y}), (\mathbf{x}_j, \bar{\mathbf{y}})) + \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{(i\mathbf{y})} \triangle(\mathbf{y}_i, \mathbf{y}) \tag{3.41}$$

Proof *Appendix A.* ■

3.7.3 Algorithm

We propose a general cutting plane algorithm for loss-sensitive support vector machines. The algorithm proposed in Section 3.5 is a special case of either version of the general algorithm. Once again we base the optimization in the dual program formulations. Notice that the constraint matrix derived for the SVM_2^* variants is diagonal, as the one for SVM_0, whereas in the SVM_1^* case, dual variables are coupled, but the couplings only occur within a block of variables associated with the same training instance. Hence, the constraint matrix is (at least) block diagonal in all cases, where each block corresponds to a specific training instance.

Pseudo-code of the algorithm is depicted in Algorithm 3. 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 proceeds by finding the (potentially) most violated constraint, involving some output value $\hat{\mathbf{y}}$. If the appropriately 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. This corresponds to 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 recomputed with respect to S (or alternatively S_i).

The presented general algorithm is implemented in the software package SVM^{struct} by Thorsten Joachims[1]. 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. The algorithm has a very general and well-defined interface independent of the choice of Ψ 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 7. 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 typically straightforward, solving the maximization problem for constraint selection

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

typically requires exploiting the structure of Ψ.

In the slack re-scaling setting, it turns out that for a given example $(\mathbf{x}_i, \mathbf{y}_i)$ we need to identify the maximum over

$$\hat{\mathbf{y}} \equiv \text{argmax}_{\mathbf{y}\in\mathcal{Y}} \{(1 - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle) \triangle(\mathbf{y}_i, \mathbf{y})\} \tag{3.42}$$

We will discuss several cases for how to solve this problem in Chapter 4. Typically, it can be solved by an appropriate modification of the prediction problem in Eq. (3.1), which recovers f from F. For example, in the case of grammar learning with the F_1 score as the loss function via $\triangle(\mathbf{y}_i, \mathbf{y}) = (1 - F_1(\mathbf{y}_i, \mathbf{y}))$, the maximum can be computed using a modified version of the CKY algorithm. More generally, 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 loss is constant, $\triangle(\mathbf{y}_i, \mathbf{y}) = const$, and then maximizes over the finite number of levels.

In the margin re-scaling setting, one needs to solve the maximization problem

$$\hat{\mathbf{y}} \equiv \text{argmax}_{\mathbf{y}\in\mathcal{Y}} \{\triangle(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle\} \tag{3.43}$$

In cases where the loss function has an additive decomposition that is compatible with the feature map, one can fold the loss function contribution into the weight vector $\langle \mathbf{w}', \delta\Psi_i(\mathbf{y})\rangle = \langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle - \triangle(\mathbf{y}_i, \mathbf{y})$ for some $\mathbf{w}'$. This means the class of cost functions defined by $F(\mathbf{x}, \cdot; \mathbf{w})$ and $F(\mathbf{x}, \cdot; \mathbf{w}) - \triangle(\mathbf{y}, \cdot)$ may actually be identical.

3.7.4 Analysis

While it is relatively straightforward to obtain a lower bound on the achievable improvement of the dual objective by selecting and adding a single dual variable $\alpha_{(i\mathbf{y})}$ to the dual problem when using quadratic penalties, the SVM_1^* formulation introduces an additional complication in the form of upper bounds on non-overlapping subsets of variables, namely, the set of variables $\alpha_{(i\mathbf{y})}$ in the current working set that correspond to the same training instance. Hence, we may not be able to answer the above question by optimizing over $\alpha_{(i\mathbf{y})}$ alone, but rather have to deal with a larger optimization problem over a whole subspace.

Algorithm 3 Loss-sensitive support vector machines (SVM_1^* and SVM_2^*).

input: $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_n, \mathbf{y}_n)$, C, ϵ
output: $\boldsymbol{\alpha}$
$S_i \leftarrow \emptyset$ for all $i = 1, \ldots, n$
repeat
  for $i = 1, \ldots, n$ **do**
    / prepare cost function for optimization */*
    set up cost function

$$H(\mathbf{y}) \equiv \begin{cases} (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle) \triangle(\mathbf{y}_i, \mathbf{y}) & (\text{SVM}_1^{\triangle s}) \\ \triangle(\mathbf{y}_i, \mathbf{y}) - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle & (\text{SVM}_1^{\triangle m}) \\ (1 - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle) \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})} & (\text{SVM}_2^{\triangle s}) \\ \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})} - \langle \delta\Psi_i(\mathbf{y}), \mathbf{w} \rangle & (\text{SVM}_2^{\triangle m}) \end{cases}$$

    where $\mathbf{w} \equiv \sum_j \sum_{\mathbf{y}' \in S_j} \alpha_{(j\mathbf{y}')} \delta\Psi_j(\mathbf{y}')$.
    / find cutting plane */*
    compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}} H(\mathbf{y})$
    / determine value of current slack variable */*
    compute $\xi_i = \max\{0, \max_{\mathbf{y} \in S_i} H(\mathbf{y})\}$
    if $H(\hat{\mathbf{y}}) > \xi_i + \epsilon$ **then**
      / add constraint to the working set */*
      $S_i \leftarrow S_i \cup \{\hat{\mathbf{y}}\}$
      / Variant (a): perform full optimization */*
      $\alpha_S \leftarrow$ optimize the dual of SVM_1^* or SVM_2^* over S, $S = \cup_i S_i$
      / Variant (b): perform subspace ascent */*
      $\alpha_{S_i} \leftarrow$ optimize the dual of SVM_1^* or SVM_2^* over S_i
    end if
  end for
until no S_i has changed during iteration

Corollary 2 *Under the same assumptions as in Lemma 1 and enforcing the constraint $\beta \leq D$ for some $D > 0$, the objective improves by*

$$\max_{0<\beta\leq D} \{\Theta(\boldsymbol{\alpha}^t + \beta\boldsymbol{\eta})\} - \Theta(\boldsymbol{\alpha}^t) = \begin{cases} \frac{\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle^2}{2\boldsymbol{\eta}' \boldsymbol{J} \boldsymbol{\eta}} & \langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle \leq D\boldsymbol{\eta}' \boldsymbol{J} \boldsymbol{\eta} \\ D\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle - \frac{D^2}{2} \boldsymbol{\eta}' \boldsymbol{J} \boldsymbol{\eta} & \textit{otherwise} \end{cases} \tag{3.44}$$

Moreover the improvement can be upper bounded by

$$\max_{0<\beta\leq D} \{\Theta(\boldsymbol{\alpha}^t + \beta\boldsymbol{\eta})\} - \Theta(\boldsymbol{\alpha}^t) \geq \frac{1}{2} \min \left\{ D, \frac{\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle}{\boldsymbol{\eta}' \boldsymbol{J} \boldsymbol{\eta}} \right\} \langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle \tag{3.45}$$

Proof *We distinguish two cases of whether $\beta^* \leq D$ or $\beta^* > D$. In the first case, we can simple apply lemma 1 since the additional constraint is inactive and does not change the solution. In the second case, the concavity of Θ implies that $\beta = D$ achieves the maximum of $\delta\Theta$ over the constrained range. Plugging in this result for β^* into $\delta\Theta$ yields the second case in the claim.*

Finally, the bound is obtained by exploiting that in the second case

$$\beta^* > D \iff D < \frac{\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle}{\boldsymbol{\eta}' \boldsymbol{J} \boldsymbol{\eta}} \tag{3.46}$$

Replacing one of the D factors in the D^2 term of the second case with this bound yields an upper bound. The first (exact) case and the bound in the second case can be compactly combined as shown in the formula of the claim. ■

Corollary 3 *Under the same assumption as in Corollary 2 and for the special case of an optimization direction $\boldsymbol{\eta} = \boldsymbol{e}_r$, the objective improves at least by*

$$\max_{0<\beta\leq D} \Theta(\boldsymbol{\alpha}^t + \beta \boldsymbol{e}_r) - \Theta(\boldsymbol{\alpha}^t) \geq \frac{1}{2} \min \left\{ D, \frac{\frac{\partial\Theta}{\partial\alpha_r}(\boldsymbol{\alpha}^t)}{J_{rr}} \right\} \frac{\partial\Theta}{\partial\alpha_r}(\boldsymbol{\alpha}^t) \tag{3.47}$$

Proof *Notice that $\boldsymbol{\eta} = \boldsymbol{e}_r$ implies $\langle \nabla\Theta, \boldsymbol{\eta} \rangle = \frac{\partial\Theta}{\partial\alpha_r}$ and $\boldsymbol{\eta}' \boldsymbol{J} \boldsymbol{\eta} = J_{rr}$.* ■

Proposition 14 ($\mathbf{SVM}_2^{\triangle s}$) *For $SVM_2^{\triangle s}$ step 10 in Algorithm 3 the improvement $\delta\Theta$ of the dual objective is lower bounded by*

$$\delta\Theta \geq \frac{1}{2} \frac{\epsilon^2}{\triangle_i R_i^2 + \frac{n}{C}}, \quad \text{where} \quad \triangle_i \equiv \max_{\mathbf{y}} \{\triangle(\mathbf{y}_i, \mathbf{y})\} \text{ and } R_i \equiv \max_{\mathbf{y}} \{\|\delta\Psi_i(\mathbf{y})\|\} \tag{3.48}$$

Proof *Using the notation in Algorithm 3 one can apply Corollary 1 with multi-index $r = (i\hat{\mathbf{y}})$, $\boldsymbol{h} = \mathbf{1}$, and $\boldsymbol{J}$ such that*

$$J_{(i\hat{\mathbf{y}})(j\mathbf{y})} = \langle \delta\Psi_i(\hat{\mathbf{y}}), \delta\Psi_j(\mathbf{y}) \rangle + \frac{\delta_{ij} n}{C \sqrt{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})}} \tag{3.49}$$

Notice that the partial derivative of Θ with respect to $\alpha_{(i\hat{\mathbf{y}})}$ is given by

$$\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\boldsymbol{\alpha}^o) = 1 - \sum_j \sum_{\mathbf{y}} \alpha^t_{(j\mathbf{y})} J_{(i\hat{\mathbf{y}})(j\mathbf{y})} = 1 - \langle \mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}}) \rangle - \frac{\xi_i^*}{\sqrt{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})}}, \tag{3.50}$$

since the optimality equations for the primal variables yield the identities

$$\mathbf{w}^* = \sum_j \sum_{\mathbf{y}} \alpha^t_{(j\mathbf{y})} \delta\Psi_j(\mathbf{y}), \quad \textit{and} \quad \xi^*_i = \sum_{\mathbf{y} \neq \mathbf{y}_i} \frac{n\alpha^t_{(i\mathbf{y})}}{C\sqrt{\triangle(\mathbf{y}_i, \mathbf{y})}} \tag{3.51}$$

Now, applying the condition of step 10

$$\sqrt{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})}\,(1 - \langle \mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}})\rangle) > \xi^*_i + \epsilon \tag{3.52}$$

leads to the bound

$$\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\boldsymbol{\alpha}^t) \geq \frac{\epsilon}{\sqrt{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})}} \tag{3.53}$$

Finally,

$$J_{rr} = \|\delta\Psi_i(\hat{\mathbf{y}})\|^2 + \frac{n}{C\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \tag{3.54}$$

and inserting this expression and the previous bound into the expression from Corollary 1 yields

$$\frac{1}{2J_{rr}} \left(\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}} \right)^2 \geq \frac{\epsilon^2}{2\left(\triangle(\mathbf{y}_i, \hat{\mathbf{y}})\|\delta\Psi_i(\hat{\mathbf{y}})\|^2 + \frac{n}{C}\right)} \geq \frac{\epsilon^2}{2\left(\triangle_i R_i^2 + \frac{n}{C}\right)} \tag{3.55}$$

The claim follows by observing that jointly optimizing over a set of variables that include α_r can only further increase the value of the dual objective. ■

Proposition 15 ($\mathbf{SVM}_2^{\triangle m}$) *For $SVM_2^{\triangle m}$ step 10 in Algorithm 3 the improvement $\delta\Theta$ of the dual objective is lower bounded by*

$$\delta\Theta \geq \frac{1}{2}\frac{\epsilon^2}{R_i^2 + \frac{n}{C}} \quad \textit{where} \quad R_i = \max_{\mathbf{y}} \|\delta\Psi_i(\mathbf{y})\| \tag{3.56}$$

Proof *By re-defining*

$$\delta\tilde{\Psi}_i(\mathbf{y}) \equiv \frac{\delta\Psi_i(\mathbf{y})}{\sqrt{\triangle(\mathbf{y}_i, \mathbf{y})}} \tag{3.57}$$

we are back to Proposition 14 with

$$\max_{\mathbf{y}}\{\triangle(\mathbf{y}_i, \mathbf{y})\|\delta\tilde{\Psi}_i(\mathbf{y})\|^2\} = \max_{\mathbf{y}}\{\|\delta\Psi_i(\mathbf{y})\|^2\} = R_i^2 \tag{3.58}$$

since

$$\langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle \geq \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})} - \xi_i \iff \langle \mathbf{w}, \delta\tilde{\Psi}_i(\mathbf{y})\rangle \geq 1 - \frac{\xi_i}{\sqrt{\triangle(\mathbf{y}_i, \mathbf{y})}} \tag{3.59}$$

■

Proposition 16 ($\mathbf{SVM}_1^{\triangle s}$) *For $SVM_1^{\triangle s}$ step 10 in Algorithm 3 the improvement $\delta\Theta$ of the dual objective is lower bounded by*

$$\delta\Theta \geq \min\left\{\frac{C\epsilon}{2n}, \frac{\epsilon^2}{8\triangle_i^2 R_i^2}\right\} \quad \text{where} \quad \triangle_i = \max_{\mathbf{y}}\{\triangle(\mathbf{y}_i, \mathbf{y})\} \text{ and } R_i = \max_{\mathbf{y}}\{\|\delta\Psi_i(\mathbf{y})\|\} \tag{3.60}$$

Proof

Case I:

If the working set does not contain an element $(i\mathbf{y})$, then we can optimize over $\alpha_{(i\hat{\mathbf{y}})}$ under the constraint that

$$\alpha_{(i\hat{\mathbf{y}})} \leq \triangle(\mathbf{y}_i, \hat{\mathbf{y}})\frac{C}{n} = D \tag{3.61}$$

Notice that

$$\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\boldsymbol{\alpha}^t) = 1 - \langle \mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}})\rangle > \frac{\xi_i^* + \epsilon}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \geq \frac{\epsilon}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \tag{3.62}$$

where the first inequality follows from the pre-condition for selecting $(i\hat{\mathbf{y}})$ and the last one from $\xi_i^* \geq 0$. Moreover, notice that $J_{(i\hat{\mathbf{y}})(i\hat{\mathbf{y}})} \leq R_i^2$. Evoking Corollary 3 with the obvious identifications yields

$$\delta\Theta \geq \frac{1}{2}\min\left\{D, \frac{1}{J_{rr}}\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\boldsymbol{\alpha}^t)\right\}\frac{\partial\Theta}{\partial\alpha_{(i\hat{\mathbf{y}})}}(\boldsymbol{\alpha}^t) \tag{3.63a}$$

$$> \frac{1}{2}\min\left\{\frac{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})C}{n}, \frac{\epsilon}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})R_i^2}\right\} \tag{3.63b}$$

$$= \frac{\epsilon}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})}\min\left\{\frac{C\epsilon}{2n}, \frac{\epsilon^2}{2R_i^2\triangle(\mathbf{y}_i, \hat{\mathbf{y}})^2}\right\} \tag{3.63c}$$

The second term can be further bounded to yield the claim.

Case II:

If there are already active constraints for instance i in the current working set, i.e. $S_i \neq \emptyset$, then we may need to reduce dual variables $\alpha_{(i\mathbf{y})}$ in order to get some slack for increasing the newly added $\alpha_{(i\hat{\mathbf{y}})}$. We thus investigate search directions $\boldsymbol{\eta}$ such that $\eta_{(i\hat{\mathbf{y}})} = 1$, $\eta_{(i\mathbf{y})} = -\frac{\alpha_{(i\mathbf{y})}}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})}\frac{n}{C} \leq 0$ for $(i\mathbf{y}) \in S_i$, and $\eta_{(j\mathbf{y}')} = 0$ in all other cases. For such $\boldsymbol{\eta}$, we guarantee that $\boldsymbol{\alpha}^t + \beta\boldsymbol{\eta} \geq 0$ since $\beta \leq \frac{C}{n}\triangle(\mathbf{y}_i, \hat{\mathbf{y}})$.

In finding a suitable direction to derive a good bound, we have two (possibly conflicting) goals. First of all, we want the directional derivative to be positively bounded away from zero. Notice that

$$\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle = \sum_{\mathbf{y}} \eta_{(i\mathbf{y})} \left(1 - \langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y}) \rangle\right) \tag{3.64}$$

Furthermore, by the restrictions imposed on $\boldsymbol{\eta}$, $\eta_{(i\mathbf{y})} < 0$ *implies that the* $(i\mathbf{y})$*-constraint is active and hence*

$$\triangle(\mathbf{y}_i, \mathbf{y}) \left(1 - \langle \mathbf{w}^*, \delta\Psi_i(\mathbf{y}) \rangle\right) = \xi_i^* \tag{3.65}$$

Moreover the pre-condition of step 10

$$\triangle(\mathbf{y}_i, \hat{\mathbf{y}}) \left(1 - \langle \mathbf{w}^*, \delta\Psi_i(\hat{\mathbf{y}}) \rangle\right) = \xi_i^* + \delta \tag{3.66}$$

where $\delta \geq \epsilon > 0$. *Hence*

$$\langle \nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta} \rangle = \frac{\xi_i^*}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \left(1 - \frac{n}{C} \sum_{\mathbf{y}} \frac{\alpha_{(i\mathbf{y})}}{\triangle(\mathbf{y}_i, \mathbf{y})}\right) + \frac{\delta}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \geq \frac{\epsilon}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \tag{3.67}$$

The second goal is to make sure the curvature along the chosen direction is not too large.

$$\boldsymbol{\eta}' \boldsymbol{J} \boldsymbol{\eta} = J_{(i\hat{\mathbf{y}})(i\hat{\mathbf{y}})} - 2 \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \frac{\alpha_{(i\mathbf{y})}^t}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \frac{n}{C} J_{(i\hat{\mathbf{y}})(i\mathbf{y})} + \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \sum_{\mathbf{y}' \neq \hat{\mathbf{y}}} \frac{\alpha_{(i\mathbf{y})}^t}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \frac{n}{C} \frac{\alpha_{(i\mathbf{y}')}^t}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \frac{n}{C} J_{(i\mathbf{y})(i\mathbf{y}')} \tag{3.68a}$$

$$\leq R_i^2 + 2 \frac{n R_i^2}{C \triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \alpha_{(i\mathbf{y})}^t + \frac{n^2 R_i^2}{C^2 \triangle(\mathbf{y}_i, \hat{\mathbf{y}})^2} \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \sum_{\mathbf{y}' \neq \hat{\mathbf{y}}} \alpha_{(i\mathbf{y})}^t \alpha_{(i\mathbf{y}')}^t \tag{3.68b}$$

$$\leq R_i^2 + 2 \frac{R_i^2 \triangle_i}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} + \frac{R_i^2 \triangle_i^2}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})^2} \tag{3.68c}$$

$$\leq \frac{4 R_i^2 \triangle_i^2}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})^2} \tag{3.68d}$$

This follows from the fact that

$$\sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \alpha_{(i\mathbf{y})}^t \leq \triangle_i \sum_{\mathbf{y} \neq \hat{\mathbf{y}}} \frac{\alpha_{(i\mathbf{y})}^t}{\triangle(\mathbf{y}_i, \mathbf{y})} \leq \frac{C \triangle_i}{n} \tag{3.69}$$

Evoking Corollary 2 yields

$$\delta\Theta \geq \frac{1}{2}\min\left\{D, \frac{\langle\nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle}{\boldsymbol{\eta}' \boldsymbol{J} \boldsymbol{\eta}}\right\}\langle\nabla\Theta(\boldsymbol{\alpha}^t), \boldsymbol{\eta}\rangle \tag{3.70a}$$

$$\geq \frac{1}{2}\min\left\{\frac{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})C}{n}, \frac{\frac{\epsilon}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})}}{\frac{4R_i^2\triangle_i^2}{\triangle(\mathbf{y}_i,\hat{\mathbf{y}})^2}}\right\}\frac{\epsilon}{\triangle(\mathbf{y}_i, \hat{\mathbf{y}})} \tag{3.70b}$$

$$= \min\left\{\frac{C\epsilon}{2n}, \frac{\epsilon^2}{8R_i^2\triangle_i^2}\right\} \tag{3.70c}$$

■

Proposition 17 ($\mathbf{SVM}_1^{\triangle m}$) *For $SVM_1^{\triangle m}$ step 10 in Algorithm 3 the improvement $\delta\Theta$ of the dual objective is lower bounded by*

$$\delta\Theta \geq \frac{\epsilon^2}{8R_i^2} \quad \textit{where} \quad R_i = \max_{\mathbf{y}} \|\delta\Psi_i(\mathbf{y})\| \tag{3.71}$$

Proof *By re-defining*

$$\delta\tilde{\Psi}_i(\mathbf{y}) \equiv \frac{\delta\Psi_i(\mathbf{y})}{\triangle(\mathbf{y}_i, \mathbf{y})} \tag{3.72}$$

we are back to Proposition 14 with

$$\max_{\mathbf{y}}\{\triangle(\mathbf{y}_i, \mathbf{y})^2\|\delta\tilde{\Psi}_i(\mathbf{y})\|^2\} = \max_{\mathbf{y}}\{\|\delta\Psi_i(\mathbf{y})\|^2\} = R_i^2 \tag{3.73}$$

since

$$\langle\mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle \geq \triangle(\mathbf{y}_i, \mathbf{y}) - \xi_i \iff \langle\mathbf{w}, \delta\tilde{\Psi}_i(\mathbf{y})\rangle \geq 1 - \frac{\xi_i}{\triangle(\mathbf{y}_i, \mathbf{y})} \tag{3.74}$$

■

Theorem 3 *With $\bar{R} = \max_i R_i$, $\bar{\triangle} = \max_i \triangle_i$ and for a given $\epsilon > 0$, Algorithm 3 terminates after incrementally adding at most*

$$\max\left\{\frac{2n\bar{\triangle}}{\epsilon}, \frac{8C\bar{\triangle}^3\bar{R}^2}{\epsilon^2}\right\}, \quad \max\left\{\frac{2n\bar{\triangle}}{\epsilon}, \frac{8C\bar{\triangle}\bar{R}^2}{\epsilon^2}\right\}, \quad \frac{C\bar{\triangle}^2\bar{R}^2 + n\bar{\triangle}}{\epsilon^2} \quad \textit{and} \quad \frac{C\bar{\triangle}\bar{R}^2 + n\bar{\triangle}}{\epsilon^2} \tag{3.75}$$

constraints to the working set S for the $SVM_1^{\triangle s}$, $SVM_1^{\triangle m}$, $SVM_2^{\triangle s}$ and $SVM_2^{\triangle m}$ 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 Propositions 14, 15, 16 and 17 respectively. 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^* and SVM_2^* respectively.* ■

3.7.5 Discussion

Let us discuss some of the advantages and disadvantages of the two formulations presented. An appealing property of the slack re-scaling approach is its scaling invariance.

Proposition 18 *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.* ■

In contrast, the margin re-scaling formulation is not invariant under scaling of the loss function. One needs, for example, to rescale the feature map Ψ by a corresponding scale factor as well. This seems to indicate that one has to calibrate the scaling of the loss and the scaling of the feature map more carefully in the $\text{SVM}_1^{\triangle m}$ formulation. The $\text{SVM}_1^{\triangle s}$ formulation on the other hand, represents the loss scale explicitly in terms of the constant C.

A second disadvantage of the margin scaling approach is that it potentially gives 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. If one interprets $F(\mathbf{x}_i, \mathbf{y}_i) - F(\mathbf{x}_i, \mathbf{y})$ as a log odds ratio of an exponential family model (cf. [44]) then the margin constraints may be dominated by

incorrect values $\mathbf{y}$ that are exponentially less likely than the target value. To be more precise, notice that in the $\text{SVM}_1^{\triangle s}$ formulation, the penalty part only depends on $\mathbf{y}$ for which $\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle \leq 1$. These are outputs that all receive a relatively "high" (i.e. 1-close to the optimum) value of $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$. However, in $\text{SVM}_1^{\triangle m}$, ξ_i^* has to majorize $\triangle(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle$ for all $\mathbf{y}$. This means ξ_i^* can be dominated by a value $\hat{\mathbf{y}} = \text{argmax}_{\mathbf{y}} \{\triangle(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle\}$ which has a large loss, but whose value of $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ comes nowhere near the optimal value of F. Nevertheless, Proposition 12 provides some justification for using the constraints set in Eq. (3.40), whereas this choice was not explicitly justified in [45].

Having presented the general algorithm for joint kernel support vector machines, we will proceed to present several interesting special cases, in order to focus on the details of specific instantiations of the generic framework, and demonstrate the versatility and effectiveness of the approach.

Chapter 4

Special Cases

4.1 Introduction

In the sequel we will discuss a number of interesting special cases, of the general framework described in the previous chapter. Furthermore to demonstrate the effectiveness and versatility of our approach, we have applied it to the problems of taxonomic text classification [5, 46], named entity recognition [2, 46], natural language parsing [46], and string-to-string mapping.

To model each particular problem and to be able to run the algorithm and bound its complexity, we need to examine the following three questions for each case:

- *Modeling*: How can we define suitable feature maps $\Psi(\mathbf{x}, \mathbf{y})$ and loss functions $\triangle(\mathbf{y}, \mathbf{y}')$ for specific problems?
- *Algorithms*: How can we compute the required maximization over $\mathcal{Y}$ for given $\mathbf{x}$?
- *Sparseness*: How can we bound $\|\Psi(\mathbf{x}, \mathbf{y}) - \Psi(\mathbf{x}, \mathbf{y}')\|$?

First we give some useful definitions. We define the canonical (binary) representation of outputs $y \in \mathcal{Y} = \{1, \ldots, k\}$ by unit vectors

$$\Lambda^c(y) \equiv (\delta(y, 1), \delta(y, 2), \ldots, \delta(y, k))' \in \{0, 1\}^k \tag{4.1}$$

so that $\langle \Lambda^c(y), \Lambda^c(y') \rangle = \delta(y, y')$. It will turn out to be convenient to use tensor products $\otimes$ and direct sums $\oplus$ to combine feature maps over $\mathcal{X}$ and $\mathcal{Y}$. We define the $\otimes$ and $\oplus$ operations in the following manner

$$\otimes : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^{dk}, \quad [\mathbf{a} \otimes \mathbf{b}]_{i+(j-1)d} \equiv [\mathbf{a}]_i [\mathbf{b}]_j \tag{4.2}$$

$$\oplus : \mathbb{R}^d \times \mathbb{R}^k \to \mathbb{R}^{d+k}, \quad [\mathbf{a} \oplus \mathbf{b}]_i \equiv \begin{cases} [\mathbf{a}]_i & \text{if } i \leq d \\ [\mathbf{b}]_{i-d} & \text{if } i > d \end{cases} \tag{4.3}$$

4.2 Multiclass Classification

A special case of Eq. (3.1) is multiclass classification, where $y \in \mathcal{Y} = \{1, \ldots, k\}$ and $\mathbf{w} = (\mathbf{w}_1', \ldots, \mathbf{w}_k')'$ is a stack of vectors, $\mathbf{w}_r$ being a weight vector associated with the r-th class. The classification rule is given by

$$f(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} F(\mathbf{x}, y; \mathbf{w}), \qquad F(\mathbf{x}, y; \mathbf{w}) = \langle \mathbf{w}_y, \Phi(\mathbf{x}) \rangle \tag{4.4}$$

Here $\Phi(\mathbf{x}) \in \mathbb{R}^d$ denotes an arbitrary feature representation of the inputs, which in many cases may be defined implicitly via a kernel function.

4.2.1 Modeling

The above decision rule can be equivalently represented by making use of a joint feature map as follows. We can define a joint feature map for the multiclass problem by

$$\Psi(\mathbf{x}, y) \equiv \Phi(\mathbf{x}) \otimes \Lambda^c(y) \tag{4.5}$$

It is is easy to show that this results in an equivalent formulation of multiclass classification as expressed in the following proposition.

Proposition 19 *$F(\mathbf{x}, y; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle$, where F is defined in Eq. (4.4) and Ψ in Eq. (4.5).*

Proof *For all* $y \in \mathcal{Y}$

$$\langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle = \langle \mathbf{w}, \Psi(\mathbf{x}) \otimes \Lambda(y) \rangle = \sum_{r=1}^{k} \delta(y, r) \langle \mathbf{w}_y, \Phi(\mathbf{x}) \rangle = \langle \mathbf{w}_y, \Phi(\mathbf{x}) \rangle \ . \tag{4.6}$$

■

4.2.2 Algorithms

It is usually assumed that the number of classes k in simple multiclass problems is small enough, so that an exhaustive search can be performed to maximize any objective over $\mathcal{Y}$. Similarly, we can find the second best $y \in \mathcal{Y}$.

4.2.3 Sparseness

In order to bound the norm of the difference feature vectors, we prove the following simple result.

Proposition 20 *Define* $R_i \equiv \|\Phi(\mathbf{x}_i)\|$. *Then* $\|\Psi(\mathbf{x}_i, y) - \Psi(\mathbf{x}_i, y')\|^2 \leq 2R_i^2$.

Proof

$$\|\Psi(\mathbf{x}_i, y) - \Psi(\mathbf{x}_i, y')\|^2 \leq \|\Psi(\mathbf{x}_i, y)\|^2 + \|\Psi(\mathbf{x}_i, y')\|^2 = 2\|\Phi(\mathbf{x}_i)\|^2 \tag{4.7}$$

where the first step follows from the Cauchy-Schwarz inequality and the second step exploits the sparseness of Λ^c. ■

4.3 Multiclass Classification with Output Features

4.3.1 Modeling

The first generalization we propose is to make use of more interesting output feature map Λ than the canonical representation in Eq. (4.1). Apparently, we could use the same approach as in Eq. (4.5) to define a joint feature function, but use a more general form for Λ.

We first show that for any joint feature map Ψ constructed via the direct tensor product $\otimes$ the following relation holds:

Proposition 21 *For $\Psi = \Phi \otimes \Lambda$ the inner product can be written as*

$$\langle \Psi(\mathbf{x}, y), \Psi(\mathbf{x}', y') \rangle = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \langle \Lambda(y), \Lambda(y') \rangle \tag{4.8}$$

Proof *By simple algebra*

$$\begin{aligned} \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}', y') \rangle &= \langle \Phi(\mathbf{x}) \otimes \Lambda(y), \Phi(\mathbf{x}') \otimes \Lambda(y') \rangle & (4.9) \\ &= \sum_{r=1}^{k} \sum_{s=1}^{k} \delta(y, r)\delta(y', s) \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle & (4.10) \\ &= \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \langle \Lambda(y), \Lambda(y') \rangle \ . & (4.11) \end{aligned}$$

■

This implies that for feature maps Φ that are implicitly defined via kernel functions K, $K(\mathbf{x}, \mathbf{x}') \equiv \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$, one can define a joint kernel function as follows,

$$J((\mathbf{x}, y), (\mathbf{x}', y')) = \langle \Psi(\mathbf{x}, y), \Psi(\mathbf{x}', y') \rangle = \langle \Lambda(y), \Lambda(y') \rangle K(\mathbf{x}, \mathbf{x}') \tag{4.12}$$

Of course, nothing prevents us from expressing the inner product in output space via yet another kernel function $L(y, y') = \langle \Lambda(y), \Lambda(y') \rangle$. Notice that the kernel L is simply the identity in the standard multiclass case. How can this kernel be chosen in concrete cases? It basically may encode any type of prior knowledge one might have about the similarity between classes. It is illuminating to note the following proposition.

Proposition 22 *Define $\Psi(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \otimes \Lambda(\mathbf{y})$ with $\Lambda(\mathbf{y}) \in \mathbb{R}^p$, then the discriminant function $F(\mathbf{x}, \mathbf{y}; \mathbf{w})$ can be written as:*

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \sum_{r=1}^{p} \lambda_r(y) \langle \mathbf{w}_y, \Phi(\mathbf{x}) \rangle \tag{4.13}$$

where $\mathbf{w} = (\mathbf{w}_1', \ldots, \mathbf{w}_p')'$ *is the stack of vectors* $\mathbf{w}_r \in \mathbb{R}^d$*, one vector for each basis function of* Λ*.*

Proof

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle = \langle \mathbf{w}, \Phi(\mathbf{x}) \otimes \Lambda(\mathbf{y}) \rangle = \sum_{r=1}^{p} \lambda_r(y) \langle \mathbf{w}_y, \Phi(\mathbf{x}) \rangle \tag{4.14}$$

■

We can give this a simple interpretation: For each output feature λ_r a corresponding weight vector $\mathbf{w}_r$ is introduced. The discriminant function can then be represented as a weighted sum of contributions coming from the different features. In particular, in the case of binary features $\Lambda : \mathcal{Y} \to \{0, 1\}^p$, this will simply be a sum over all contributions $\langle \mathbf{w}_r, \Phi(\mathbf{x}) \rangle$ of features that are active for the class y, i.e. for which $\lambda_r(y) = 1$.

It is also important to note that the orthogonal representation provides a maximally large hypothesis class and that nothing can be gained in terms of representational power by including *additional* features.

Corollary 4 *Assume a mapping* $\Lambda(y) = (\tilde{\Lambda}(y)', \Lambda^c(y)')'$*,* $\tilde{\Lambda}(y) \in \mathbb{R}^p$ *and define* $\tilde{\Psi}(\mathbf{x}, y) = \Phi(\mathbf{x}) \otimes \Lambda(y)$ *and* $\Psi(\mathbf{x}, y) = \Phi(\mathbf{x}) \otimes \Lambda^c(y)$*. Now, for every* $\tilde{\mathbf{w}}$ *there is a* $\mathbf{w}$ *such that* $\left\langle \tilde{\mathbf{w}}, \tilde{\Psi}(\mathbf{x}, y) \right\rangle = \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle$ *and vice versa.*

Proof

$$\left\langle \tilde{\mathbf{w}}, \tilde{\Psi}(\mathbf{x}, y) \right\rangle = \sum_{r=1}^{p+k} \lambda_r(y) \langle \tilde{\mathbf{v}}_r, \Phi(\mathbf{x}) \rangle \tag{4.15}$$

$$= \left\langle \sum_{r=1}^{p+k} \lambda_r(y) \tilde{\mathbf{v}}_r, \Phi(\mathbf{x}) \right\rangle \tag{4.16}$$

$$= \langle \mathbf{v}_y, \Phi(\mathbf{x}) \rangle \tag{4.17}$$

$$= \langle \mathbf{w}, \Psi(\mathbf{x}, y) \rangle \tag{4.18}$$

where we have defined $\mathbf{v}_y = \sum_{r=1}^{p+k} \lambda_r(y) \tilde{\mathbf{v}}_r$*. The reverse direction is trivial and requires setting* $\tilde{\mathbf{v}}_r = 0$ *for* $r = 1, \ldots, p$*.* ■

In the light of this corollary, we would like to emphasize that the rationale behind the use of class features is not to increase the representational power of the hypothesis space, but to re-parameterize (or even constrain) the hypothesis space such that a more suitable representation for $\mathcal{Y}$ is produced. We would like to *generalize across classes* as we want to generalize across input patterns in the standard formulation of classification problems. Obviously, orthogonal representations (corresponding to diagonal kernels) will provide no generalization whatsoever across different classes y. The choice of a good output feature map Λ is thus expected to provide an inductive bias, namely that learning can occur across a set of classes sharing a common property.

Let us discuss some special cases of interest.

Classification with Taxonomies Assume that class labels y are arranged in a taxonomy. We will define a taxonomy as a set of elements $\mathcal{Z} \supseteq \mathcal{Y}$ equipped with a partial order $\prec$. The partially ordered set $(\mathcal{Z}, \prec)$ might, for example, represent a tree or a lattice (see Figure 4.1). Now we can define binary features for classes as follows: Associate one feature λ_z with every element in $\mathcal{Z}$ according to

$$\lambda_z(y) = \begin{cases} 1 & \text{if } y \prec z \text{ or } y = z \\ 0 & \text{otherwise} \end{cases} \tag{4.19}$$

This includes multiclass classification as a special case of an unordered set $\mathcal{Z} = \mathcal{Y}$. In general, however, the features λ_z will be "shared" by all classes below z, e.g. all nodes y in the subtree rooted at z in the case of a tree. One may also introduce a relative weight β_z for every feature and define a β-weighted (instead of binary) output feature map $\tilde{\Lambda}$ as $\tilde{\lambda}_z = \beta_z \lambda_z$. If we reflect upon the implication of this definition in the light of Proposition 22, one observes that this effectively introduces a weight vector $\mathbf{v}_z$ for every element of $\mathcal{Z}$, i.e. for every node in the hierarchy.

Learning with Textual Class Descriptions As a second motivating example, we consider problems where classes are characterized by short glosses, blurbs or other textual descriptions. We would like to exploit the fact that classes sharing some descriptors are likely to be similar, in order to specify a suitable inductive bias. This

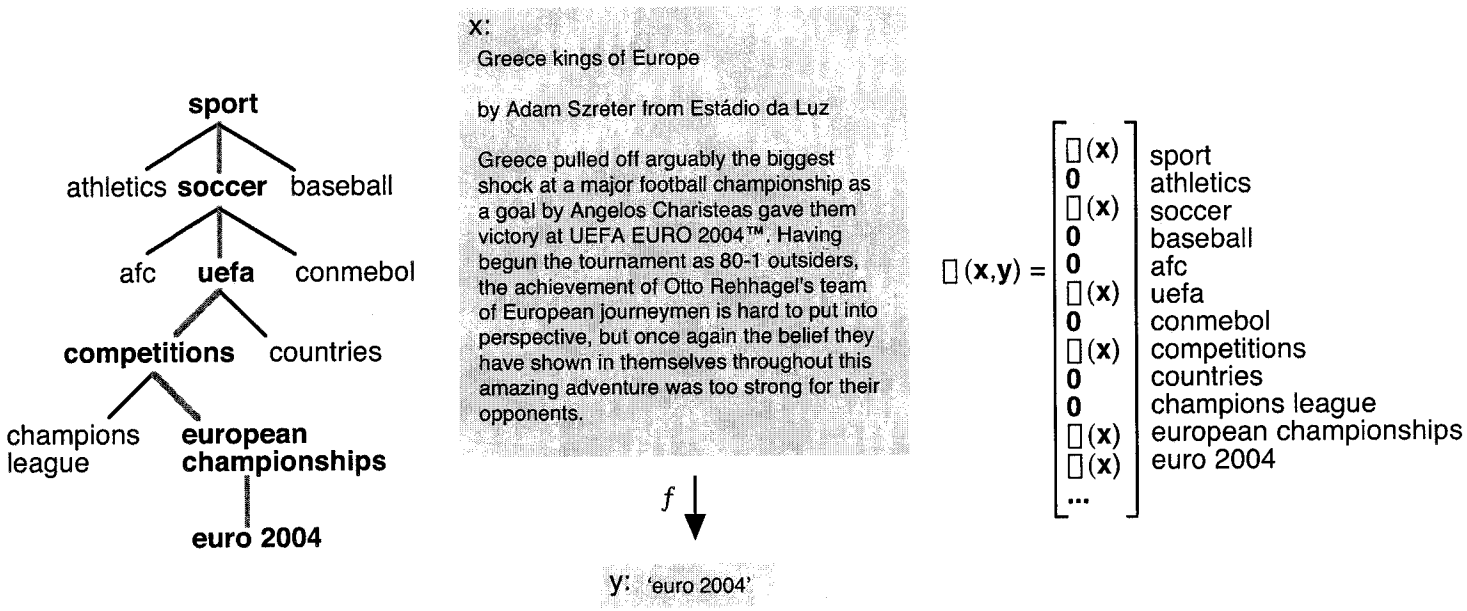


Figure 4.1: Classification with taxonomies. The text in the middle belongs to the class 'euro 2004'. The classes are organized in a tree structure, from the most general (root) to the most specific (leaves). To the right we show the joint feature map. The class 'euro 2004' draws features from its parent classes.

can be achieved, for example, by associating a feature λ with every keyword used to describe classes, in addition to the class identity. Hence standard vector space models like term-frequency of idf representations can be applied to model classes and the inner product $\langle \Lambda(y), \Lambda(y') \rangle$ then defines a similarity measure between classes corresponding to the standard cosine-measure used in information retrieval.

Learning with Class Similarities The above example can obviously be generalized to any situation, where we have access to a positive definite similarity function for pairs of classes. To come up with suitable similarity functions is part of the domain model – very much like determining a good representation of the inputs – and we assume here that it is given.

4.3.2 Algorithms

As in the multiclass case, we assume that the number of classes is small enough to perform an exhaustive search.

4.3.3 Sparseness

Proposition 20 can be generalized in the following way:

Proposition 23 *Define* $R_i \equiv \|\Phi(\mathbf{x}_i)\|$ *and* $S \equiv \max_{\mathbf{y} \in \mathcal{Y}} \|\Lambda(y)\|$. *Then* $\|\Psi(\mathbf{x}_i, y) - \Psi(\mathbf{x}_i, y')\|^2 \leq 2R_i^2 S^2$ *for all* $y, y' \in \mathcal{Y}$.

Proof $\langle \Psi(\mathbf{x}_i, y), \Psi(\mathbf{x}_i, y) \rangle = \|\Phi(\mathbf{x}_i)\|^2 \|\Lambda(y)\|^2 \leq R_i^2 S^2$. *In the first step, we have used Proposition 21.* ■

4.3.4 Application: Classification with Taxonomies

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

	flt 0/1	tax 0/1	flt △	tax △	
4 training instances per class					
acc	28.32	28.32	27.47	29.74	+5.01 %
△-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 %
△-loss	1.54	1.51	1.39	1.33	+13.67 %

Table 4.1: 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, '△' refers to training based on the tree loss.

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 sub-sampled 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.[1] As a suitable loss function Δ, we have used a tree loss function which defines the loss between two classes y and y' as the height of the first common ancestor of y and y' in the taxonomy. The results are summarized in Table 4.1 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.

4.4 Label Sequence Learning

The next problem we would like to formulate in the joint feature map framework is the problem of label sequence learning, or sequence segmentation/annotation. Here, the goal is to predict a label sequence $\mathbf{y} = (y^1, \ldots, y^l)$ for a given observation sequence $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^l)$. In order to simplify the presentation, let us assume all sequences are of the same length l. Let us denote by Σ the set of possible labels for each individual variable y^t, i.e. $\mathcal{Y} = \Sigma^l$. Hence each sequence of labels is considered to be a class of

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

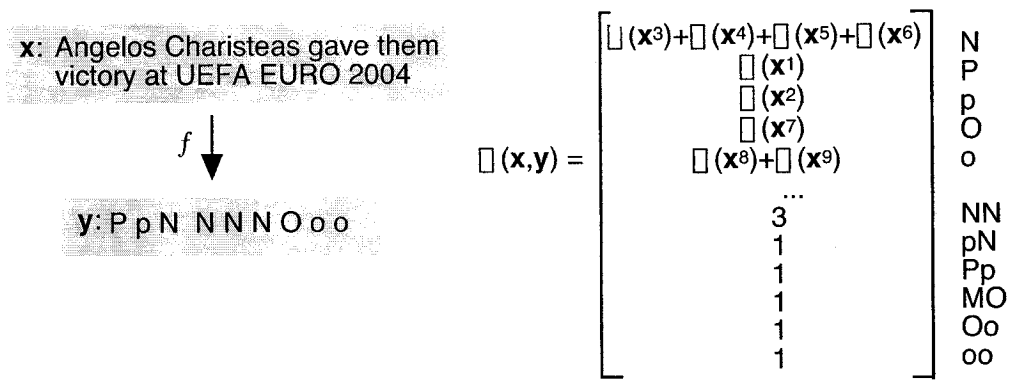


Figure 4.2: Label sequence learning. To the left we see a sentence and the label sequence that corresponds to name tags (e.g. person, organization). To the right we show the feature map. It includes joint features between input and output, via multiple copies of input features associated with a specific label, and features between nearby labels, counts of occurrences of pairs of labels.

its own, resulting in a multiclass classification problem with $|\Sigma|^l$ different classes. To model label sequence learning in this manner would of course not be very useful, if one were to apply standard multiclass classification methods. However, this can be overcome by an appropriate definition of the discriminant function.

4.4.1 Modeling

Inspired by Hidden Markov Model (HMM) type of interactions, we propose to define Ψ to include interactions between input features and labels via multiple copies of the input features, as well as features that model interactions between nearby label variables. It is perhaps most intuitive to start from the discriminant function

$$F(\mathbf{x},\mathbf{y};\mathbf{w}) = \sum_{t=1}^{l}\sum_{\sigma\in\Sigma}\langle \mathbf{u}_\sigma, \Phi(\mathbf{x}^t)\rangle \delta(y^t,\sigma) + \eta \sum_{t=1}^{l-1}\sum_{\sigma\in\Sigma}\sum_{\sigma'\in\Sigma} \mathbf{v}_{\sigma\sigma'}\delta(y^t,\sigma)\delta(y^{t+1},\sigma') \tag{4.20a}$$

$$= \left\langle \mathbf{u}, \sum_{t=1}^{l} \Phi(\mathbf{x}^t)\otimes\Lambda^c(y^t)\right\rangle + \eta \left\langle \mathbf{v}, \sum_{t=1}^{l-1} \Lambda^c(y^t)\otimes\Lambda^c(y^{t+1})\right\rangle \tag{4.20b}$$

Here $\mathbf{w} = (\mathbf{u}', \mathbf{v}')'$, Λ^c denotes the orthogonal representation of labels over Σ, and $\eta \geq 0$ is a scaling factor which balances the two types of contributions. It is straightforward

to read off the joint feature map implicit in the definition of the HMM-discriminant from Eq. (4.20b),

$$\Psi(\mathbf{x},\mathbf{y}) = \left[\sum_{t=1}^{l} \Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t)\right] \oplus \left[\eta \sum_{t=1}^{l-1} \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})\right] \tag{4.21}$$

Notice that similar to the multiclass case, we can revoke Proposition 21 in the case of an implicit representation of Φ via a kernel function K and the inner product between labeled sequences can thus be written as

$$\langle \Psi(\mathbf{x},\mathbf{y}), \Psi(\bar{\mathbf{x}},\bar{\mathbf{y}}) \rangle = \sum_{t=1}^{l}\sum_{s=1}^{l} \delta(y^t,\bar{y}^s) K(\mathbf{x}^t,\bar{\mathbf{x}}^s) + \eta^2 \sum_{t=1}^{l-1}\sum_{s=1}^{l-1} \delta(y^t,\bar{y}^s)\delta(y^{t+1},\bar{y}^{s+1}) = \tag{4.22}$$

$$\sum_{t=1}^{l}\sum_{s=1}^{l} \left\langle \Lambda^c(y^t), \Lambda^c(\bar{y}^s) \right\rangle K(\mathbf{x}^t,\bar{\mathbf{x}}^s) + \eta^2 \sum_{t=1}^{l-1}\sum_{s=1}^{l-1} \left\langle \Lambda^c(y^t), \Lambda^c(\bar{y}^s) \right\rangle \left\langle \Lambda^c(y^{t+1}), \Lambda^c(\bar{y}^{s+1}) \right\rangle \tag{4.23}$$

A larger family of discriminant functions can be obtained by using more powerful feature functions Ψ. We would like to mention three ways of extending the previous HMM discriminant. First of all, one can extract features not just from $\mathbf{x}^t$, but from a window around $\mathbf{x}^t$, e.g. replacing $\Phi(\mathbf{x}^t)$ with $\Phi(\mathbf{x}^{t-r},\ldots,\mathbf{x}^t,\ldots,\mathbf{x}^{t+r})$. Since the same input pattern $\mathbf{x}^t$ now occurs in multiple terms, this has been called the use of *overlapping* features [24] in the context of label sequence learning. Secondly, it is also straightforward to include higher order label-label interactions beyond pairwise interactions by including higher order tensor terms, for instance, label triplets $\sum_t \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1}) \otimes \Lambda^c(y^{t+2})$, etc. Thirdly, one can also combine higher order output features with input features, for example, by including terms of the type using $\sum_t \Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})$.

4.4.2 Algorithms

The maximization of $\langle \mathbf{w}, \Psi(\mathbf{x}_i,\mathbf{y}) \rangle$ over $\mathbf{y}$ can be carried out by dynamic programming, since the cost contributions are additive over sites and contain only linear and nearest neighbor quadratic contributions. In particular, in order to find the best label

Algorithm 4 Viterbi decoding.

input: $\mathbf{x} = (\mathbf{x}^1, \ldots, \mathbf{x}^l), \mathbf{w} = (\mathbf{u}', \mathbf{v}')'$
output: $\hat{\mathbf{y}}, F(\mathbf{x}, \hat{\mathbf{y}}; \mathbf{w})$
/* initialization */
for $\sigma \in \Sigma$ **do**
  Score$[1, \sigma] \leftarrow \langle \mathbf{u}_\sigma, \mathbf{x}^1 \rangle$
end for
/* recursion */
for $t = 2, \ldots, l$ **do**
  for $\sigma' \in \Sigma$ **do**
    Score$[t, \sigma'] \leftarrow \max_{\sigma \in \Sigma} \{\text{Score}[t-1, \sigma] v_{\sigma\sigma'}\} \langle \mathbf{u}_{\sigma'}, \mathbf{x}^t \rangle$
    LabelSequenceTable$[t, \sigma'] \leftarrow \text{argmax}_{\sigma \in \Sigma} \{\text{Score}[t-1, \sigma] v_{\sigma\sigma'}\}$
  end for
end for
/* termination */
$\hat{y}^l = \text{argmax}_{\sigma \in \Sigma} \text{Score}[l, \sigma]$
$F(\mathbf{x}, \hat{\mathbf{y}}; \mathbf{w}) = \max_{\sigma \in \Sigma} \text{Score}[l, \sigma]$
/* reconstruction */
for $t = l-1, \ldots, 1$ **do**
  $\hat{y}^t = \text{LabelSequenceTable}[t+1, \hat{y}^{t+1}]$
end for
$\hat{\mathbf{y}} = (y^1, \ldots, y^l)$

sequence $\hat{\mathbf{y}} \neq \mathbf{y}_i$, one can perform Viterbi decoding [15, 43] (see Algorithm 4), which can also determine the second best sequence for the zero-one loss (2-best Viterbi decoding). Viterbi decoding can also be used with other loss functions by computing the maximization for all possible values of the loss function.

4.4.3 Sparseness

Proposition 24 *Define* $R_i \equiv \max_t \|\Phi(\mathbf{x}_i^t)\|$. *Then*

$$\|\Psi(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \mathbf{y}')\|^2 \leq 2l^2(R_i^2 + \eta^2) \tag{4.24}$$

Proof *Notice that*

$$\|\Psi(\mathbf{x}_i, \mathbf{y})\|^2 = \|\sum_t \Phi(\mathbf{x}_i^t) \otimes \Lambda^c(y^t)\|^2 + \eta^2 \|\sum_t \Lambda^c(y^t) \otimes \Lambda^c(y^{t+1})\|^2 \tag{4.25}$$

Method	HMM	CRF	Perceptron	SVM
Error	9.36	5.17	5.94	5.08

Table 4.2: Results of various algorithms on the Named Entity Recognition task.

The first squared norm can be upper bounded by

$$\|\sum_t \Phi(\mathbf{x}_i^t) \otimes \Lambda^c(y^t)\|^2 = \sum_s \sum_t \left\langle \Phi(\mathbf{x}_i^s), \Phi(\mathbf{x}_i^t) \right\rangle \delta(y^s, y^t) \leq l^2 R_i^2 \tag{4.26}$$

and the second one by $\eta^2 l^2$, which yields the claim. ■

4.4.4 Application: Named Entity Recognition

We study our algorithm for label sequence learning 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. A 5-fold cross validation was performed. The label set in this corpus consists 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 [2], 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. An adapted version of the Viterbi algorithm is used to solve the argmax in step 7. 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. Finally we used the Hamming distance as the loss function for loss-sensitive SVM formulations.

The 5-fold cross validation results given in Table 4.2 for the zero-one loss, compare the generative HMM with Conditional Random Fields (CRF) [24], perceptron [7] 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 4.3 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

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

Table 4.3: Results of various joint kernel SVM formulations on the Named Entity Recognition task.

Hamming distance 1 to the correct label sequence. Note that for loss equal to 1 all SVM formulations are equivalent.

4.4.5 Discussion

The predominant formalism for modeling and learning label sequence has been based on Hidden Markov Models (HMMs) and variations thereof. HMMs model sequential dependencies by treating the label sequence as a Markov chain. This avoids direct dependencies between subsequent observations and leads to an efficient dynamic programming formulation for inference and learning. Yet, despite their success, HMMs have at least three major limitations. (i) They are typically trained in a non-discriminative manner. (ii) The conditional independence assumptions are often too restrictive. (iii) They are based on explicit feature representations and lack the power of kernel-based methods. To address these shortcomings several discriminative approaches, that includes maximum entropy Markov models (MEMMs) [31, 38], conditional random fields (CRFs) [24], perceptron re-ranking [7] and label sequence boosting [1], have recently been proposed. The basic commonality between joint kernel SVMs and these methods is their discriminative approach to modeling and the fact that they can account for overlapping features, that is, labels can depend directly on features of past or future observations. Joint kernel SVMs furthermore combine the maximum margin principle and a kernel-based approach to learning non-linear discriminant functions. Perceptron and conditonal random fields [25] can also be kernelized.

4.5 Weighted Context-Free Grammars

In natural language parsing, the task is to predict a labeled tree $\mathbf{y}$ based on a sequence $\mathbf{x} = (x^1, ..., x^l)$ of terminal symbols. For this problem, our approach extends the approaches of [6] and [10] to an efficient maximum-margin algorithm with general loss functions. We assume that each node in the tree corresponds to the application of a context-free grammar rule. The leaves of the tree are the symbols in $\mathbf{x}$, while interior nodes correspond to non-terminal symbols from a given alphabet Σ. For simplicity, we assume that the trees are in Chomsky normal form. This means that each internal node has exactly two children. An exception are pre-terminal nodes (non-leaf nodes that have a terminal symbol as child) which have exactly one child.

4.5.1 Modeling

We consider weighted context-free grammars to model the dependency between x and y. Grammar rules are of the form $y^{rs} \rightarrow y^{rt}y^{(t+1)s}$, $1 \leq t \leq r < s \leq l$ or $y^t \rightarrow x^t$, $t = 1, ..., l$, where $y^t \in \Sigma$ are non-terminal symbols, and $x^t \in T$ are terminal symbols. Each such rule is parameterized by an individual weight w_r. A particular kind of weighted context-free grammars are probabilistic context-free grammars (PCFGs), where the weights $w_{\sigma\rightarrow\sigma'\sigma''}$ $(\sigma, \sigma', \sigma'' \in \Sigma)$ and $w_{\sigma\rightarrow\tau}$ $(\sigma \in \Sigma, \tau \in T)$ are the log-probabilities of expanding nodes y^{ts} and y^t with respective rules. In PCFGs, the individual node probabilities are assumed to be independent, so that the probability $P(\mathbf{x}, \mathbf{y})$ of sequence $\mathbf{x}$ and tree $\mathbf{y}$ is the product of the node probabilities in the tree. The most likely parse tree to yield $\mathbf{x}$ from a designated start symbol is the predicted label $h(\mathbf{x})$. This leads to the following maximization problem.

$$h(\mathbf{x}) = \text{argmax}_{\mathbf{y}\in\mathcal{Y}} P(\mathbf{y}|\mathbf{x}) \tag{4.27}$$

$$= \text{argmax}_{\mathbf{y}\in\mathcal{Y}} \left\{ \sum_{t=1}^{l-1} \sum_{(\sigma,\sigma',\sigma'')\in\Sigma^3} w_{\sigma\rightarrow\sigma'\sigma''}\delta(y^{rs},\sigma)\delta(y^{rt},\sigma')\delta(y^{(t+1)s},\sigma'') + \right. \tag{4.28}$$

$$\left. + \sum_{t=1}^{l} \sum_{(\sigma,\tau)\in\Sigma\times T} w_{\sigma\rightarrow\tau}\delta(y^t,\sigma)\delta(x^t,\tau) \right\} \tag{4.29}$$

More generally, weighted context-free grammars can be used in our framework as follows. $\Psi(x, y)$ contains one feature $\psi_{\sigma \to \sigma'\sigma''}$ for each node of type $\sigma \to \sigma'\sigma''$ and one feature $\psi_{\sigma \to \tau}$ for each node of type $\sigma \to \tau$. As illustrated in Figure 3.1, the number of times a particular rule occurs in the tree is the value of the feature. The weight vector $\mathbf{w}$ contains the corresponding weights so that

$$F(\mathbf{x}, \mathbf{y}; \mathbf{w}) = \sum_{t=1}^{l-1} \sum_{(\sigma,\sigma',\sigma'') \in \Sigma^3} w_{\sigma \to \sigma'\sigma''} \delta(y^{rs}, \sigma) \delta(y^{rt}, \sigma') \delta(y^{(t+1)s}, \sigma'') + \tag{4.30}$$

$$+ \sum_{t=1}^{l} \sum_{(\sigma,\tau) \in \Sigma \times T} w_{\sigma \to \tau} \delta(y^t, \sigma) \delta(x^t, \tau) \tag{4.31}$$

$$= \left\langle \bar{\mathbf{w}}, \sum_{t=1}^{l} \Phi^c(x^t) \otimes \Lambda^c(y^t) \right\rangle + \left\langle \hat{\mathbf{w}}, \sum_{t=1}^{l-1} \Lambda^c(y^{rs}) \otimes \Lambda^c(y^{rt}) \otimes \Lambda^c(y^{(t+1)s}) \right\rangle \tag{4.32}$$

It is easy to read off the joint feature map implicit in the definition of the discriminant from Eq. 4.32

$$\Psi(\mathbf{x}, \mathbf{y}) = \left[\sum_{t=1}^{l} \Phi^c(x^t) \otimes \Lambda^c(y^t) \right] \oplus \left[\sum_{t=1}^{l-1} \Lambda^c(y^{rs}) \otimes \Lambda^c(y^{rt}) \otimes \Lambda^c(y^{(t+1)s}) \right] \tag{4.33}$$

Note that our framework also allows more complex $\Psi(\mathbf{x}, \mathbf{y})$, making it more flexible than PCFGs. In particular, each node weight can be a (kernelized) linear function of the full x and the span of the subtree.

4.5.2 Algorithms

The solution of $\text{argmax}_{\mathbf{y} \in \mathcal{Y}} \langle \mathbf{w}, \Psi(\mathbf{x}, \mathbf{y}) \rangle$ for a given $\mathbf{x}$ can be determined efficiently using a CKY-Parser [30] (see Algorithm 5), which can also return the second best parse for learning with the zero-one loss. To implement other loss functions, like $\triangle(\mathbf{y}_i, \mathbf{y}) = (1 - F_1(\mathbf{y}_i, \mathbf{y}))$, the CKY algorithm can be extended to compute both $\text{argmax}_{\mathbf{y} \in \mathcal{Y}}(1 - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle) \triangle(\mathbf{y}_i, \mathbf{y})$ as well as $\text{argmax}_{\mathbf{y} \in \mathcal{Y}}(\triangle(\mathbf{y}_i, \mathbf{y}) - \langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle)$ by stratifying the maximization over all values of $\triangle(\mathbf{y}_i, \mathbf{y})$.

Algorithm 5 CKY parsing.

input: $\mathbf{x} = (x^1, \ldots, x^l), \mathbf{w} = (\mathbf{u}', \mathbf{v}')'$
output: $\hat{\mathbf{y}} = \{(r, s, \sigma \rightarrow \sigma'\sigma'') | r \leq t < s, t = 1, \ldots, l-1, \sigma, \sigma', \sigma'' \in \Sigma\}, F(\mathbf{x}, \hat{\mathbf{y}}; \mathbf{w})$
/* initialization */
for $t = 1, \ldots, l$ **do**
  for $\sigma \in \Sigma$ **do**
    $\text{Score}[t, t, \sigma] \leftarrow u_{\sigma \rightarrow \tau}$, where $x^t = \tau$
  end for
end for
/* recursion */
for $k = 2, \ldots, l$ **do**
  for $r = 1, \ldots, l - k + 1$ **do**
    $s \leftarrow r + k - 1$
    for $t = r, \ldots, s - 1$ **do**
      for $\sigma, \sigma', \sigma'' \in \Sigma$ **do**
        $z \leftarrow \text{Score}[r, t, \sigma'] + \text{Score}[t + 1, s, \sigma''] + v_{\sigma \rightarrow \sigma'\sigma''}$
        if $z > \text{Score}[r, s, \sigma]$ **then**
          $\text{Score}[r, s, \sigma] \leftarrow z$
          $\text{ParseTable}[r, s, \sigma] \leftarrow (t, \sigma', \sigma'')$
        end if
      end for
    end for
  end for
end for
/* recosntruction */
$\hat{\mathbf{y}} = parse(\text{ParseTable}, 1, l, \sigma^0), F(\mathbf{x}, \hat{\mathbf{y}}; \mathbf{w}) = \text{Score}[1, l, \sigma^0]$,
where $\sigma^0 \in \Sigma$ is the start symbol, and $parse(\text{ParseTable}, r, s, \sigma) =$
$\{(r, s, \sigma \rightarrow \sigma'\sigma''), parse(\text{ParseTable}, r, t, \sigma'), parse(\text{ParseTable}, t + 1, s, \sigma'')\}$

4.5.3 Sparseness

Since the trees branch for each internal node, a tree over a sequence $\mathbf{x}$ of length l has $l - 1$ internal nodes. Furthermore, it has l pre-terminal nodes. This means that the L_1-norm of $\Psi(\mathbf{x}, \mathbf{y})$ is $2l - 1$ and that the L_2-norm of $\Psi(\mathbf{x}, \mathbf{y}) - \Psi(\mathbf{x}, \mathbf{y}')$ is at most $\sqrt{4l^2 + 4(l-1)^2} < 2\sqrt{2}l$.

	Train				Test				Training Efficiency			
Method	Acc	Prec	Rec	F_1	Acc	Prec	Rec	F_1	CPU-h	%SVM	Iter	Const
PCFG	61.4	92.4	88.5	90.4	55.2	86.8	85.2	86.0	0	N/A	N/A	N/A
SVM_2	66.3	92.8	91.2	92.0	58.9	85.3	87.2	86.2	1.2	81.6	17	7494
$\text{SVM}_2^{\triangle s}$	62.2	93.9	90.4	92.1	58.9	88.9	88.1	88.5	3.4	10.5	12	8043
$\text{SVM}_2^{\triangle m}$	63.5	93.9	90.8	92.3	58.3	88.7	88.1	88.4	3.5	18.0	16	7117

Table 4.4: Results for learning a weighted context-free grammar on the Penn Treebank.

4.5.4 Application: Natural Language Parsing

We test the feasibility of our approach for learning a weighted context-free grammar (see Figure 3.1) 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 [22], 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[2].

The results are given in Table 4.4. They show micro-averaged precision, recall, and F_1 for the training and the test set. The first line shows the performance of the 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 $\triangle(y_i, y) = (1 - F_1(y_i, y))$ using $\text{SVM}_2^{\triangle s}$ and $\text{SVM}_2^{\triangle m}$. All results are for $C = 1$ and $\epsilon = 0.01$. All values of C between 10^{-1} to 10^2 gave comparable prediction performance. While the zero-one loss (which is also implicitly used in Perceptrons [9, 7]) achieves better accuracy (i.e. predicting the complete tree correctly), the F_1-score is only marginally better compared to the PCFG model. However, optimizing the SVM for the F_1-loss gives substantially better F_1-scores, outperforming the PCFG 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 4.4 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 of the algorithm. This might be sped up, since we were using a

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

rather naive algorithm in the experiments.

4.6 String-to-String Mappings

We now focus on the problem of string-to-string mapping, where the goal is to predict an output string $\mathbf{y} = (y^1, \ldots, y^l)$ for a given input string $\mathbf{x} = (x^1, \ldots, x^m)$. Let us denote by Σ and T the alphabets of each individual character x^t, $(t = 1, ..., l)$ and y^s $(s = 1, ..., m)$ respectively, i.e. $\mathcal{X} = \Sigma^*$ and $\mathcal{Y} = T^*$.

4.6.1 Modeling

We propose to learn mappings from strings to strings with a combined string kernel computing joint features from an input string $\mathbf{x}$ and an output string $\mathbf{y}$, e.g. the joint occurrence of some substring in $\mathbf{x}$ and some other substring in $\mathbf{y}$. It is convenient to use direct tensor products to combine feature maps over $\mathcal{X}$ and $\mathcal{Y}$. We define a joint feature map for string to string mapping by

$$\Psi(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \otimes \Lambda(\mathbf{y}) \tag{4.34}$$

We now notice that for any joint feature map Ψ constructed via the direct tensor product $\otimes$ as $\Psi = \Phi \otimes \Lambda$ the inner product can be written (cf. Proposition 21)

$$\langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}', \mathbf{y}') \rangle = \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle \langle \Lambda(\mathbf{y}), \Lambda(\mathbf{y}') \rangle \tag{4.35}$$

This implies that for feature maps Φ, Λ, that are implicitly defined via kernel functions K, $K(\mathbf{x}, \mathbf{x}') \equiv \langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \rangle$ and L, $L(\mathbf{y}, \mathbf{y}') \equiv \langle \Lambda(\mathbf{y}), \Lambda(\mathbf{y}') \rangle$, one can define a joint kernel function as follows,

$$J((\mathbf{x}, \mathbf{y}), (\mathbf{x}', \mathbf{y}')) = \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}', \mathbf{y}') \rangle = K(\mathbf{x}, \mathbf{x}') L(\mathbf{y}, \mathbf{y}') \tag{4.36}$$

and perform the computation independently for each kernel. For K and L we can employ one of the proposed string kernels, based on substrings [18], gapped substrings [29], k-length substrings [26], or mismatch penalties [27].

Method	Test Err
Baseline	0.903±0.026
k-NN*	0.963±0.037
KDE*	0.813±0.078
SVM_2	0.723±0.089
$\text{SVM}_2^{\triangle s}$	0.605±0.034
$\text{SVM}_2^{\triangle m}$	0.600±0.041

Table 4.5: Results on toy string to string mapping example.

4.6.2 Algorithms

We propose to use some external process to enumerate a small number of candidate outputs for a given $\mathbf{x}$. This has been pursued in [9, 7] for problems like parsing. In general, all that is required for learning is a set $\text{Gen}(\mathbf{x}_i) \subseteq \mathcal{Y}$ for every training instance. This can be understood as a process of boosting the performance of the original algorithm which was used to generate the candidate set Gen. For example, the latter may use a non-discriminative approach or a simple discriminative method like nearest neighbor, which may be inexpensive to perform, even for large configuration spaces $\mathcal{Y}$.

4.6.3 Sparseness

In order to bound the norm of the difference feature vectors, we prove the following simple result.

Proposition 25 *Define* $R_i \equiv \|\Phi(\mathbf{x}_i)\|$ *and* $S \equiv \max_{\mathbf{y}\in\mathcal{Y}} \|\Lambda(\mathbf{y})\|$. *Then*

$$\|\Psi(\mathbf{x}_i, \mathbf{y}) - \Psi(\mathbf{x}_i, \mathbf{y}')\|^2 \leq 2R_i^2 S^2 \tag{4.37}$$

for all $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$.

Proof $\langle \Psi(\mathbf{x}_i, \mathbf{y}), \Psi(\mathbf{x}_i, \mathbf{y}) \rangle = \|\Phi(\mathbf{x}_i)\|^2 \|\Lambda(\mathbf{y})\|^2 \leq R_i^2 S^2$. *In the first step, we have used Proposition 21.* ■

4.6.4 Example: Synthetic Data

As a toy example we generated synthetic data as described in [49]. In summary, there are three classes of strings of letters from the same alphabet of four letters (a, b, c, d). Input strings from all classes are generated by a Markov model with a random length between 10 and 15. In the first class transition from any letter to any other are equally likely, in the second class transition from any letter to itself has probability 0.7 and all other transitions 0.1, and finally in the third class only letters c and d are used with transition from any letter to itself 0.7. The output strings for each class are *abad*, *abbd* and *aabc* respectively, corrupted by the following noise. There is a probability of 0.3 of a random insertion of a random letter and 0.15 of two random insertions. After the potential insertions there is a probability of 0.3 of a random deletion and a probability of 0.15 of two random deletions.

We generated 200 such strings and performed 4-fold cross validation. For the joint feature map framework we have used combined string kernels. For KDE separate string kernels for the input and output were used. The parameters of the string kernels were $\lambda = 0.01$ and n-grams $= 3$. KDE also used an extra Gaussian kernel for the input strings whose parameters were chosen by 5-fold cross validation. SVM parameter C was set to 1. For all methods 1 minus the string kernel value between any incorrect output string and the correct one defined the loss of incorrect strings.

For k-NN ($k > 1$) as well as KDE a pre-image is found by the closest training example output to the given solution. Similarly, in order to apply our algorithm to this problem, the set of output strings was reduced to all the observed output string in the training data. It is thus as though function Gen($\mathbf{x}$) generated, out of all possible noisy outputs, only the ones observed in the training data.

Table 4.5 shows that SVM and especially the loss re-scaling formulations outperform k-NN and KDE. Note that the results reported for k-NN and KDE are the ones reported in [49], so the dataset is not exactly the same as the one we used, but they were both randomly generated according to the same procedure.

4.6.5 Discussion

On the surface our approach is related to the *kernel dependency estimation* approach [49]. There, however, separate kernel functions are defined for the input and output space, with the idea to encode a priori knowledge about the similarity or loss function in output space. In particular, this assumes that the loss is input dependent and known beforehand. More specifically, in [49] a kernel PCA is used in the feature space defined over $\mathcal{Y}$ to reduce the problem to a (small) number of independent regression problems. The latter corresponds to an unsupervised embedding (followed by dimension reduction) performed in the output space and no information about the inputs $\mathbf{x}$ is utilized in defining this low-dimensional representation. In contrast, the key idea in our approach is not primarily to define more complex functions, but to deal with more complex output spaces by extracting combined features over inputs and outputs. A discrete output variable $\mathbf{y}$ may not be simply a simple output, but may have an internal structure that can itself be described by certain features. These features may, in turn, interact in non-trivial ways with certain properties of the input patterns.

Special Case	Loss Type	$\triangle(\mathbf{y}, \bar{\mathbf{y}})$	Range
Multiclass classification	Zero-one	$\delta(y, \bar{y})]$	$\{0, 1\}$
Classification with taxonomies	Tree distance	$\sum_{y \prec z} \delta(z, z) + \sum_{\bar{y} \prec z} \delta(z, z) - \sum_{y \prec z, \bar{y} \prec z} \delta(z, z)$	$\{1, 2, \ldots, k-1\}$
Label sequence learning	Hamming	$\sum_{t=1}^{l} \delta(y^t, \bar{y}^t)$	$\{1, 2, \ldots, l\}$
Weighted context-free grammars	$1-\mathrm{F}_1$-score	$\left[\sum_{1 \leq r < s \leq l} \delta(y^{rs}, \bar{y}^{rs})\right] / (l-1)$	$[0, 1]$
String-to-string mapping	1−SK-score	$1 - L(\mathbf{y}, \bar{\mathbf{y}})$	$[0, 1]$

Table 4.6: Loss functions for various special cases.

Special Case	$\lvert\mathcal{Y}\rvert$	$\Psi(\mathbf{x},\mathbf{y})$	$J((\mathbf{x},\mathbf{y}),(\bar{\mathbf{x}},\bar{\mathbf{y}}))$
Multiclass classification	k	$\Phi(\mathbf{x}) \otimes \Lambda^c(y)$	$K(\mathbf{x},\bar{\mathbf{x}})\delta(y,\bar{y})$
Multiclass classification with output features	k	$\Phi(\mathbf{x}) \otimes \Lambda(y)$, $\Lambda(y) = (\tilde{\Lambda}(y)', \Lambda^c(y)')'$	$K(\mathbf{x},\bar{\mathbf{x}})L(y,\bar{y})$
Classification with taxonomies	k	$\Phi(\mathbf{x}) \otimes \Lambda(y)$, $\Lambda(y) = \sum_{y \prec y'} \Lambda^c(y')$	$K(\mathbf{x},\bar{\mathbf{x}})L(y,\bar{y})$
Label sequence learning	$\lvert\mathcal{X}\rvert\lvert\Sigma\rvert$ + $\lvert\Sigma^3\rvert$	$\sum_{t=1}^{l} \Phi(\mathbf{x}^t) \otimes \Lambda^c(y^t)$ $\oplus$ $\sum_{t=1}^{l-1} \Lambda(y^t) \otimes \Lambda^c(y^{t+1})$	$\sum_{t=1}^{l}\sum_{\bar{t}=1}^{\bar{l}} \delta(y^t,\bar{y}^{\bar{t}})K(\mathbf{x}^t,\bar{\mathbf{x}}^{\bar{t}})$ + $\eta^2 \sum_{t=1}^{l-1}\sum_{\bar{t}=1}^{\bar{l}-1} \delta(y^t,\bar{y}^{\bar{t}})\delta(y^{t+1},\bar{y}^{\bar{t}+1})$
Weighted context-free grammars	$\lvert X\rvert\lvert\Sigma\rvert$ + $\lvert\Sigma\rvert^3$	$\sum_{t=1}^{l} \Phi^c(x^t) \otimes \Lambda^c(y^t)$ $\oplus$ $\sum_{t=1}^{l-1} \Lambda^c(y^{rs}) \otimes \Lambda^c(y^{rt}) \otimes \Lambda^c(y^{(t+1)s})$	$\sum_{t=1}^{l}\sum_{\bar{t}=1}^{\bar{l}} \delta(y^t,\bar{y}^{\bar{t}})\delta(x^t,\bar{x}^s)$ + $\sum_{t=1}^{l-1}\sum_{\bar{t}=1}^{\bar{l}-1} \delta(y^{rs},\bar{y}^{\bar{r}\bar{s}})\delta(y^{rt},\bar{y}^{\bar{r}\bar{t}})\delta(y^{(t+1)s},\bar{y}^{(\bar{t}+1)\bar{s}})$
String to string mapping	$\lvert\text{Gen}(\mathbf{x})\rvert$	$\Phi(\mathbf{x}) \otimes \Lambda(\mathbf{y})$	$K(\mathbf{x},\mathbf{x}')L(\mathbf{y},\mathbf{y}')$

Table 4.7: Joint feature maps for various special cases.

Chapter 5

Conclusions

5.1 Summary

We presented a maximum-margin approach to learning functional dependencies for discrete interdependent and structured output spaces. In particular, we considered cases where the predicted outputs can themselves be characterized by output-specific attributes or are structured. The key idea is to model the problem as a (kernelized) linear discriminant function over a joint feature space of inputs and outputs. We demonstrated that our approach is very general, covering problems from natural language parsing and label sequence learning to multilabel classification and classification with output features. While the resulting training problem can be exponential in size, we presented an algorithm for which we prove polynomial convergence for a large class of problems. We also evaluated the algorithm empirically on a broad range of applications. The experiments show that the algorithm is feasible in practice and that it produces promising results in comparison to conventional generative models. A key advantage of the algorithm is the flexibility to include different loss functions, making it possible to optimize an upper bound on the desired performance criterion. Furthermore, the ability to include kernels opens the opportunity to learn more complex dependencies compared to conventional, mostly linear models.

5.2 Future Work

Various improvements with respect to the optimization method seem possible that would lead to more scalable algorithms. The naive sequential heuristic, for example, for selecting the next training example could be replaced by a parallel step that would evaluate the violation of the KKT conditions for each constraint and select the training example that incurs the largest violation.

A number of experimental evaluations could also be interesting. For example, an experimental evaluation of different loss functions and different upper bounds across a number of datasets, as well as an experimental verification of the sparseness properties.

This thesis has focused on learning general dependencies over *discrete* output spaces. Likewise, in many applications of regression methods, one has to predict multiple outputs simultaneously. Special cases include prediction of sequence values, e.g., time series prediction, or prediction of a spatial field of values, e.g. image modeling. An open research problem is to develop a generalization of SVM regression that can take into account dependencies between multiple outputs. This framework should unite SVM and the idea of ϵ-sensitive loss functions with Markov networks to model statistical dependencies. An important ensuing question is related to the computational aspects and sparseness issues: Is it possible to deal with an infinite number of constraints? This will lead to a semi-definite quadratic program that needs to be solved. Other related questions are to investigate how multivalue SVM regression compares to multivalue Gaussian process regression (cokriging) and other techniques, such as kernel dependency estimation.

Appendix A

Derivation of Dual Formulations

1-Norm Soft Margin–Slack Re-Scaling

The primal Lagrangian of the problem $\text{SVM}_1^{\triangle s}$ of Eq. (3.37) is

$$L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_i \xi_i \tag{A.1}$$

$$- \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \left[\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle - 1 + \frac{\xi_i}{\triangle(\mathbf{y}_i,\mathbf{y})} \right] - \sum_i \rho_i \xi_i \tag{A.2}$$

where $\boldsymbol{\alpha},\boldsymbol{\rho} \geq 0$ are the Lagrange mulipliers. The corresponding dual is found by differentiating with respect to $\mathbf{w}$ and $\boldsymbol{\xi}$, imposing stationarity

$$\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho})}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \delta\Psi_i(\mathbf{y}) = 0 \tag{A.3}$$

$$\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho})}{\partial \xi_i} = \frac{C}{n} - \sum_{\mathbf{y}\neq\mathbf{y}_i} \frac{\alpha_{i\mathbf{y}}}{\triangle(\mathbf{y}_i,\mathbf{y})} + \rho_i = 0 \tag{A.4}$$

and resubstituting the relations obtained into the primal we obtain the dual objective

$$L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho}) = \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} - \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \sum_j \sum_{\bar{\mathbf{y}}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \alpha_{j\bar{\mathbf{y}}} \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \rangle \tag{A.5}$$

which is identical to that for SVM_1. The difference is that the box constraints

$$\sum_{\mathbf{y}\neq\mathbf{y}_i} \frac{\alpha_{i\mathbf{y}}}{\triangle(\mathbf{y}_i,\mathbf{y})} \leq \frac{C}{n} \tag{A.6}$$

are now scaled by the loss function. The Karush-Kahn-Tucker complementarity conditions are given by the following equations

$$\alpha_{i\mathbf{y}}\left[\langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle - 1 + \frac{\xi_i}{\triangle(\mathbf{y}_i,\mathbf{y})}\right] = 0 \quad \forall i, \forall \mathbf{y} \neq \mathbf{y}_i \tag{A.7}$$

1-Norm Soft Margin–Margin Re-scaling

The primal Lagrangian of the problem $\text{SVM}_1^{\triangle m}$ is

$$L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{n}\sum_i \xi_i \tag{A.8}$$

$$- \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\left[\langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle - \triangle(\mathbf{y}_i,\mathbf{y}) + \xi_i\right] - \sum_i \rho_i\xi_i \tag{A.9}$$

where $\boldsymbol{\alpha}, \boldsymbol{\rho}, \geq 0$ are the Lagrange mulipliers. The corresponding dual is found by differentiating with respect to $\mathbf{w}$ and $\boldsymbol{\xi}$, imposing stationarity

$$\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho})}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\delta\Psi_i(\mathbf{y}) = 0 \tag{A.10}$$

$$\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho})}{\partial \xi_i} = \frac{C}{n} - \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} - \rho_i = 0 \tag{A.11}$$

and resubstituting the relations obtained into the primal we obtain the dual objective

$$L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho}) = \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\triangle(\mathbf{y}_i,\mathbf{y}) - \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \sum_j \sum_{\bar{\mathbf{y}}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\alpha_{j\bar{\mathbf{y}}} \langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}})\rangle \tag{A.12}$$

which differs from that for SVM_1 in the linear term that is now scaled by the loss function. The constraints are exactly the same as for SVM_1. The Karush-Kahn-Tucker complementarity conditions are given by the following equations

$$\alpha_{i\mathbf{y}}\left[\langle \mathbf{w}, \delta\Psi_i(\mathbf{y})\rangle - \triangle(\mathbf{y}_i,\mathbf{y}) + \xi_i\right] = 0 \quad \forall i, \forall \mathbf{y} \neq \mathbf{y}_i \tag{A.13}$$

2-Norm Soft Margin–Slack Re-scaling

The primal Lagrangian of the problem $\mathrm{SVM}_2^{\triangle s}$ is

$$L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2n}\sum_i \xi_i^2 \tag{A.14}$$

$$- \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \left[\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle - 1 + \frac{\xi_i}{\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}} \right] \tag{A.15}$$

where $\boldsymbol{\alpha},\boldsymbol{\rho} \geq 0$ are the Lagrange mulipliers. The corresponding dual is found by differentiating with respect to $\mathbf{w}$ and $\boldsymbol{\xi}$, imposing stationarity

$$\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho})}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \delta\Psi_i(\mathbf{y}) = 0 \tag{A.16}$$

$$\frac{\partial L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho})}{\partial \xi_i} = \frac{C}{n}\xi_i - \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} = 0 \tag{A.17}$$

and resubstituting the relations obtained into the primal we obtain the dual objective

$$L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\rho}) = \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \tag{A.18}$$

$$-\frac{1}{2}\sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \sum_j \sum_{\bar{\mathbf{y}}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}}\alpha_{j\bar{\mathbf{y}}} \left[\langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\bar{\mathbf{y}}) \rangle + \frac{n\delta_{ij}}{C\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}\sqrt{\triangle(\mathbf{y}_j,\bar{\mathbf{y}})}} \right] \tag{A.19}$$

The Karush-Kahn-Tucker complementarity conditions are given by the following equations

$$\alpha_{i\mathbf{y}} \left[\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle - 1 + \frac{\xi_i}{\sqrt{\triangle(\mathbf{y}_i,\mathbf{y})}} \right] = 0 \quad \forall i, \forall \mathbf{y} \neq \mathbf{y}_i \tag{A.20}$$

2-Norm Soft Margin–Margin Re-scaling

The primal Lagrangian of the problem $\mathrm{SVM}_1^{\triangle m}$ is

$$L(\mathbf{w},\boldsymbol{\xi},\boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{2n}\sum_i \xi_i^2 - \sum_i \sum_{\mathbf{y}\neq\mathbf{y}_i} \alpha_{i\mathbf{y}} \left[\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle - \sqrt{\triangle(\mathbf{y}_i,\mathbf{y})} + \xi_i \right] \tag{A.21}$$

where $\boldsymbol{\alpha} \geq 0$ are the Lagrange mulipliers. The corresponding dual is found by differentiating with respect to $\mathbf{w}$ and $\boldsymbol{\xi}$, imposing stationarity

$$\frac{\partial L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \delta\Psi_i(\mathbf{y}) = 0 \tag{A.22}$$

$$\frac{\partial L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha})}{\partial \xi_i} = \frac{C}{n}\xi_i - \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} = 0 \tag{A.23}$$

and resubstituting the relations obtained into the primal we obtain the dual objective

$$L(\mathbf{w}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \sum_i \sum_{\mathbf{y} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})} \tag{A.24}$$

$$-\frac{1}{2} \sum_i \sum_{\mathbf{y} \neq \mathbf{y}_i} \sum_j \sum_{\tilde{\mathbf{y}} \neq \mathbf{y}_i} \alpha_{i\mathbf{y}} \alpha_{j\tilde{\mathbf{y}}} \left[\langle \delta\Psi_i(\mathbf{y}), \delta\Psi_j(\tilde{\mathbf{y}}) \rangle + \frac{n\delta_{ij}}{C} \right] \tag{A.25}$$

The Karush-Kahn-Tucker complementarity conditions are given by the following equations

$$\alpha_{i\mathbf{y}} \left[\langle \mathbf{w}, \delta\Psi_i(\mathbf{y}) \rangle - \sqrt{\triangle(\mathbf{y}_i, \mathbf{y})} + \xi_i \right] = 0 \quad \forall i, \forall \mathbf{y} \neq \mathbf{y}_i \tag{A.26}$$

Appendix B

Notation

Table B.1: Notation conventions used in this thesis.

Symbol	Interpretation
$\mathcal{X}$	input space
$\mathcal{Y}$	ouput space
$\|\mathcal{Y}\|$	size of output space
$\mathcal{S}$	sample
$(\mathbf{x}, \mathbf{y})$	training example
i, j	example identifier
n	number of examples
F	discriminant function
f	classification function
$(i\mathbf{y}_i), (j\mathbf{y}_j)$	training example identifiers
$(i\mathbf{y}), (j\bar{\mathbf{y}})$	pseudo example identifiers
$\mathbf{y}_i$	correct output of a given training example
$\hat{\mathbf{y}}$	highest scoring incorrect label
k	number of classes
r	class identifier
l, m	output length

continued on next page

Table B.1: *continued*

Symbol	Interpretation
r, s, t	indices of output components
Σ, T	output component space
σ, τ	output components
Φ	input feature map
Φ^c	canonical input feature map
Λ	output feature map
Λ^c	canonical output feature map
Ψ	joint input-output feature map
$\delta\Psi_i(\mathbf{y})$	joint feature map difference
K	input kernel function
L	output kernel function
J	joint kernel function
$\mathbb{R}$	real numbers
$\mathbb{R}_+$	space of non-negative real numbers
d, p	dimensions
$\mathbf{w}$	normal vector of a hyperplane
$\mathbf{u}, \mathbf{v}$	weight vector components
γ	margin
ξ	slack variable
L_1	l_1-norm
L_2	l_2-norm
C	complexity vs classification error trade-off constant
$\triangle$	loss function
α	Lagrange multiplier / expansion coefficient of weight vector
ρ	Lagrange multiplier
ϵ	precision
S_i	working set for a given example

continued on next page

Table B.1: *continued*

Symbol	Interpretation
S	working set over all examples
Θ	objective function
η	optimization direction
R_i	maximum joint feature map difference for a given example
$\bar{R}$	maximum joint feature map difference over all examples
$\triangle_i$	maximum loss for a given example
$\bar{\triangle}$	maximum loss over all examples

Bibliography

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

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

[3] Dimitri P. Bertsekas. *Nonlinear Programming: Second Edition.* Athena Scientific, 2004.

[4] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[5] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proceedings of the ACM 13th Conference on Information and Knowledge Management*, 2004.

[6] M. Collins. Discriminative reranking for natural language parsing. In *International Conference on Machine Learning*, 2000.

[7] 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.

[8] M. Collins. *Parameter Estimation for Statistical Parsing Models: Theory and Practice of Distribution-Free Methods.* Kluwer, 2004.

[9] 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.

[10] M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Conference of the Association for Computational Linguistics*, 2002.

[11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

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

[13] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines (and other kernel-based learning methods).* Cambridge University Press, 2000.

[14] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Artificial Intelligence Research*, 2:263–286, 1995.

[15] G. D. Forney Jr. The viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, 1973.

[16] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[17] D Haussler. Convolutional kernels on discrete structures. Technical Report UCSC-CRL-99-10, Computer Science Department, University of California at Santa Cruz, 1999.

[18] R. Herbrich. *Learning kernel classifiers: theory and algorithms.* MIT Press, 2002.

[19] T. Hofmann, I. Tsochantaridis, and Y. Altun. Learning over structured output spaces via kernel functions. In *Proceedings of the Sixth Kernel Workshop*, 2002.

[20] T. Joachims. Making large-scale svm learning practical. In Bernhard Schölkopf, Chris Burges, and Alex Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 169–185. MIT Press, 1998.

[21] Thorsten Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the Tenth European Conference on Machine Learning*, pages 137–142, London, UK, 1998. Springer-Verlag.

[22] M. Johnson. PCFG models of linguistic tree representations. *Computational Linguistics*, 1999.

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

[24] 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.

[25] J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, AB, 2004.

[26] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: a string kernel for protein classification. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, 2002.

[27] C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for svm protein classification. In *Advances in Neural Information Processing Systems*, 2002.

[28] C. J. Lin. On the convergence of the decomposition method for support vector machines. *Transactions on Neural Networks*, 12:1288–1298, 2001.

[29] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, Feb 2002.

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

[31] A. McCallum, D. Freitag, and F. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 591–598, San Francisco, 2000. Morgan Kaufmann.

[32] M. Minsky and S.A.Papert. *Perceptrons: An introduction to computational geometry.* MIT Press, 1969.

[33] T. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[34] A. Novikoff. On convergence proofs for perceptrons. In *Proceeding of the Symposium on the Mathematical Theory of Automata*, volume 12, pages 615–622, 1963.

[35] E. Osuna, R. Freund, and F. Girosi. An improved training algorithm for suppport vector machines. In *Proceeding of the IEEE Workshop on Neural Networks for Signal Processing VII*, pages 276–285. IEEE, 1997.

[36] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: An application to face detection. In *Proceeding of the Conference on Computer Vision and Pattern Recognition*, pages 130–136, 1997.

[37] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In Bernhard Schölkopf, Chris Burges, and Alex Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, pages 185–208. MIT Press, 1998.

[38] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *Advances in Neural Information Processing Systems 13*, pages 995–1001, Cambridge, MA, 2001. MIT Press.

[39] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.

[40] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill, New York, NY, 1986.

[41] B. Schölkopf and A. Smola. *Learning with kernels.* MIT Press, 2002.

[42] B. Schölkopf, A. J. Smola, and K. R. Müller. Kernel principal component analysis. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 327–352. MIT Press, 1999.

[43] 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.

[44] A. Smola and T. Hofmann. Exponential families for generative and discriminative estimation. Technical report, 2003.

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

[46] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machines for inter-dependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, Banff, AB, 2004.

[47] V.N. Vapnik. *Statistical Learning Theory.* Wiley, 1998.

[48] C. Watkins. Dynamic alignment kernels. In A.J. Smola, P.L. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 39–50. MIT Press, Cambridge, MA, 2000.

[49] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik. Kernel dependency estimation. In *Advances in Neural Information Processing Systems 15*, Cambridge, MA, 2003. MIT Press.

[50] J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD-TR-98-04, Department of Computer Science, Royal Holloway, University of London, 1998.