Abstract of "Discriminative Methods for Label Sequence Learning" by Yasemin Altun, Ph.D., Brown University, May 2005.

Discriminative learning framework is one of the very successful fields of machine learning. The methods of this paradigm, such as Boosting and Support Vector Machines, have significantly advanced the state-of-the-art for classification by improving the accuracy and by increasing the applicability of machine learning methods. One of the key benefits of these methods is their ability to learn efficiently in high dimensional feature spaces, either by the use of implicit data representations via kernels or by explicit feature induction. However, traditionally these methods do not exploit dependencies between class labels where more than one label is predicted. Many real-world classification problems involve sequential, temporal or structural dependencies between multiple labels. The goal of this research is to generalize discriminative learning methods for such scenarios. In particular, we focus on label sequence learning.

Label sequence learning is the problem of inferring a state sequence from an observation sequence, where the state sequence may encode a labeling, an annotation or a segmentation of the sequence. Prominent examples include part-of-speech tagging, named entity classification, information extraction, continuous speech recognition, and secondary protein structure prediction.

In this thesis, we present three novel discriminative methods that are generalizations of AdaBoost and multiclass Support Vector Machines (SVM) and a Gaussian Process formulation for label sequence learning. These techniques combine the efficiency of dynamic programming methods with the advantages of the state-of-the-art learning methods. We present theoretical analysis and experimental evaluations on pitch accent prediction, named entity recognition and part-of-speech tagging which demonstrate the advantages over classical approaches like Hidden Markov Models as well as the state-of-the-art methods like Conditional Random Fields.

Discriminative Methods for Label Sequence Learning

by

Yasemin Altun

B.S., Middle East Technical University, 1997

M.Sc., Middle East Technical University, 1999

M.S., Brown University, 2003

Thesis

Submitted in partial fulfillment of the requirements for the
degree of Doctor of Philosophy in the Department of Computer Science
at Brown University.

May 2005

Date _____ _____

Thomas Hofmann, Advisor

Recommended to the Graduate Council

Date _____ _____

Mark Johnson, Reader

Date _____ _____

Eugene Charniak,Michael Collins, Reader

Approved by the Graduate Council

Date _____ _____

Acknowledgements

I would like to thank Thomas Hofmann for his invaluable guidance and support over the course of the time I was at Brown. He has been a great role model on every aspect of academic life. I am also profoundly grateful to Mark Johnson, for his priceless advice, very fruitful discussions and his support on the research direction I investigated. I also would like to thank Michael Collins and Alex Smola for their excellent suggestions and thought-provoking questions.

I want to express my appreciation to my collegues Massimiliano Ciaramita and Ioannis Tsochantaridis for the valuable research discussions, but most of all for being such great friends; to Aparna Nadig for always being there and taking care of me, for all the times we had together, for all the blues, reds and yellows, for all the dances; to Umut Ones, iyi ki yapmisiz.

My biggest thanks go to my family: to my sisters for their love and constant support, to my mom for all the sacrifices she has made, for her unconditional love and support. This thesis is dedicated to my grandmother: Doktor oldum anneannecigim...

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Supervised learning is one of the most important areas of machine learning. Binary classification (where the label set is $\{0, 1\}$), multi-class classification (where the label set is $\{0, 1, \ldots, m\}$) and regression (where the label set is $\Re$) are instances of this problem.

In supervised learning, we are given a labeled training sample of observation-response variable pairs (x, y) where $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. It is assumed that (x, y) pairs are drawn from an unknown but fixed distribution $p(x, y)$ defined over the joint space $\mathcal{X} \times \mathcal{Y}$. The goal is to learn a function f that predicts the best value for the response variable y given a new observation x. Generative learning methods aim to model the joint distribution $p(x, y)$ of observations and response variables and predict a label for the response variable of a new observation using the conditional probability $p(y|x)$ obtained by the Bayes rule. However, it has been commonly argued by many researchers that "*one should solve the [classification/regression] problem directly and never solve a more general problem as an intermediate step*" [Vapnik, 1998]. The *intermediate step* mentioned here is the modeling of the joint distribution as performed by the methods of the generative framework. Discriminative learning methods, on the other hand, solve the problem directly by learning a mapping from the observation space to label space, as suggested by Vapnik.

Discriminative learning methods, such as Boosting and Support Vector Machines are the state-of-the-art methods for supervised classification. They have improved the classification accuracy significantly and increased the applicability of machine learning methods. The main advantage of these methods is their ability to learn efficiently in

high dimensional feature spaces. This is achieved either by the use of implicit data representations via kernels or by explicit feature induction. These methods were originally proposed for binary classification, but have been generalized to multiclass classification and regression frameworks successfully. The problems that we are interested in this thesis are multi-class classification of multiple observations whose response variables are inter-dependent.

1.1 Structured Learning

In many real world problems, the response variables that we would like to predict are not isolated, but live within a temporal or structural dependency structure. These structures range from sequences to trees, lattices to more general graphs. Below are a few examples of these structures.

Sequences: An application where the response variables form a sequence dependency structure is protein secondary structure prediction. This is an important intermediate task for solving the protein folding problem, one of the most challenging tasks of molecular biology. The protein folding problem deals with predicting the three-dimensional structure (Figure 1.1) of a protein from a sequence of amino acids. Given the difficulty of predicting the tertiary structure, molecular biologists designed the simpler task of predicting the secondary structure, i.e. predicting shorter segments such as alpha helices, beta strands, which are packed into the tertiary structure later on. Figure 1.2 is an example of an amino acid sequence and its secondary structure. The dependence between the variables is due to the fact that a number of variables must be assigned the same label in order to construct a secondary structure.

Tree dependency: Syntactic parsing is a popular tree dependency example. Parsing is useful for almost every task in Natural Language Processing (NLP), e.g. grammar checking, machine translation, question answering. The task is to identify the phrase structure of a sentence generated by the grammatical derivations in order to construct the sentence. Figure 1.3 example of a sentence and its parse tree. The parse tree is formed by a particular configuration of labels of the nodes of the tree.

Lattices: In speech recognition, the goal is to recognize a sentence given its acoustic representation. Due to the very large number of similar possible realizations of the decoding of an acoustic signal, it is common to use lattices for efficient data representation and computation. Figure 1.4 presents an example of such a mapping taken from

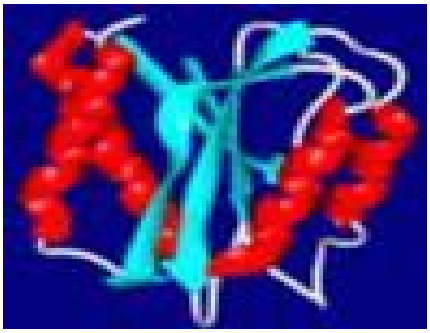

Figure 1.1: Three-dimensional structure of a protein

```
CCPTMAARIQYNACRALGTPRPVCAALSGCKILDVTKCPPDYRY
-EE---HHHHHHHHHHH-----HHHHHHHH--EE-------HHH
```

Figure 1.2: Example amino acid sequence and its secondary structure

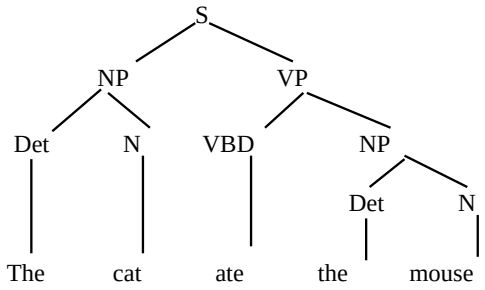

Figure 1.3: The parse tree of the sentence "The cat ate the mouse"

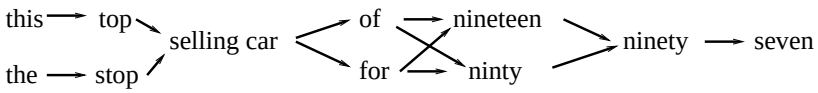

Figure 1.4: Example word lattice of an acoustic signal

[Hall and Johnson, 2004]. A word hypothesis (such as *top*) may be consistent with only a subset of the word hypotheses (such as *this*) and will be inconsistent with others (such as *the*).

There are other dependency structures observed in real world problems. In this thesis, we focus on the sequence dependency structures. The problem of labeling, annotating or segmenting observation sequences arises in many applications across a variety of scientific disciplines, mostly in natural language processing (NLP), information retrieval (IR), speech recognition, and computational biology. Prominent application examples include part-of-speech tagging (POS), named entity classification (NER), shallow parsing and chunking within the NLP domain [Manning and Schütze, 1999], information extraction and text segmentation within the IR domain, continuous speech recognition, pitch accent prediction and phoneme classification within the speech processing domain [Jurafsky and Martin, 2000], as well as secondary protein structure prediction, gene classification and protein homology detection within the computational biology domain [Durbin et al., 1998]. Problems of this type are commonly called **label sequence learning** or *sequential learning* [Dietterich, 2002].

1.2 Discriminative Label Sequence Learning

One drawback of discriminative methods is that traditionally they do not exploit dependencies between class labels where more than one label is predicted. For this reason, till recently, the predominant formalism for modeling label sequences has been based on Hidden Markov Models (HMMs) and variations thereof. Yet, despite their success HMMs have two major shortcomings. First, HMMs are typically trained in a non-discriminative manner using maximum likelihood estimation for a joint sampling model of observation and label sequences [1]. Second, efficient inference and learning in

[1] One can also train HMMs discriminatively. This method is closely related with McCallum et al. [2000]'s the Maximum Entropy Markov Models [Smola, 2004].

this setting often requires making questionable conditional independence assumptions. More precisely, in the case of HMMs, it is assumed that the Markov blanket of the hidden label variable at time step t consists of the previous and next labels as well as the t-th observation. This implies that all dependencies on past and future observations are mediated through neighboring labels which is clearly violated in many applications (especially where long distance dependencies are important).

The first problem poses the challenge of finding more appropriate *objective functions*, i.e. alternatives to the log-likelihood that are more closely related to application-relevant performance measures. The second problem is one of developing more powerful *architectures*, for example, by allowing direct dependencies between a label and past or future observations (overlapping features) or by efficiently handling higher-order combinations of input features. At the same time, one would like to address these shortcomings without sacrificing some of the benefits that HMMs offer, namely a dynamic programming formulation for efficient inference and learning.

To overcome the limitations of HMMs, different approaches for learning conditional models, such as Maximum Entropy Markov Models (MEMMs) [McCallum et al., 2000, Punyakanok and Roth, 2000], Conditional Random Fields (CRFs) [Lafferty et al., 2001, Johnson et al., 1999], perceptron re-ranking [Collins, 2000, 2002a, Collins and Duffy, 2001] have been proposed. These methods basically have two main advantages compared to HMMs: First, they are trained discriminatively by maximizing a conditional (or pseudo-) likelihood criterion. Second, they are more flexible in modeling additional dependencies such as direct dependencies of the t-th label on past or future observations. Among these models, CRFs are the state-of-the-art in terms of accuracy. In most general terms, CRFs define a conditional model over label sequences given an observation sequence in terms of an exponential family. Thus, they are a natural generalization of logistic regression to the problem of sequence learning.

1.3 Contributions

Label sequence learning In this thesis, we continue previous approaches of learning conditional models and investigate the use of other discriminative learning methods for label sequence learning. We focus on the supervised learning setting, thus we assume the availability of a set of labeled training sequences from which a mapping from observation sequences to correct label sequences is learned. Label sequence learning

can then be considered as a natural extension of supervised classification where the label space grows exponentially with the length of sequences. We propose efficient learning and inference algorithms that overcome the tractability issues caused by the extremely large class space by exploiting the sequence structure of the observation and label spaces. In particular, we present generalizations of the two most competitive discriminative methods for classification, Boosting and Support Vector Machines (SVMs) the problem of label sequence learning, as well as a Gaussian Process (GP) formulation for label sequence learning.

Sequence Boosting We first investigate different loss functions, an exponential loss function such as the one used in boosting algorithms [Schapire and Singer, 1999, Friedman et al., 2000] based on ranking the entire sequence and a loss function that explicitly minimize the zero/one loss on labels, i.e. the Hamming loss, as opposed to minimizing the loss on the entire sequence, as alternatives to the loss function of CRFs. We also present a boosting algorithm, Sequence AdaBoost [Altun et al., 2003a] that optimizes the exponential loss function for label sequences. It has the advantage of performing implicit feature selection, typically resulting in very sparse models. Feature sparseness is important for model regularization as well as for efficiency in high dimensional feature spaces.

Hidden Markov Support Vector Machines Both Sequence AdaBoost and the previous work mentioned above lack the power of kernel-based methods due to their explicit feature representations. We developed an architecture for learning label sequences which combines HMMs with SVMs in an innovative way (joint work with Ioannis Tschochantaridis). This novel architecture is called Hidden Markov SVM (HM-SVM) [Altun et al., 2003c]. HM-SVMs address all the shortcomings of HMMs, while retaining some of the key advantages of HMMs, namely the Markov chain dependency structure between labels and an efficient dynamic programming formulation. The two crucial properties of HM-SVMs inherited from SVMs are the maximum margin principle and a kernel-centric approach to learning non-linear discriminant functions.

Gaussian Process Sequence Classification We also investigate the use of a Gaussian Process (GP) formulation of label sequence learning, leading to Gaussian Prior Sequence (GPS) Classification [Altun et al., 2004b,a]. The main motivation for pursuing

this direction is to combine the best of both worlds from CRFs and SVMs. More specifically, one would like to preserve the main strength of CRFs, its rigorous probabilistic semantics. There are two important advantages of a probabilistic model. First, it is very intuitive to incorporate prior knowledge within a probabilistic framework. Second, in addition to predicting the best labels, one can compute posterior label probabilities and thus derive confidence scores for predictions. This property is valuable in particular for applications that require a cascaded architecture of classifiers. Confidence scores can be propagated to subsequent processing stages or used to abstain on certain predictions. The other design goal is the ability to use kernel functions in order to construct and learn over Reproducing Kernel Hilbert Spaces (RKHS), thereby to overcome the limitations of (finite-dimensional) parametric statistical models. A second, independent objective of GPS is to gain clarity with respect to two aspects on which CRFs and the SVM-based methods differ, the first aspect being the loss function (logistic loss vs. hinge loss) and the second aspect being the mechanism used for constructing the hypothesis space (parametric vs. RKHS).

It is important to note that even though the techniques proposed in this study are presented for sequences, they are immediately applicable to more general structures for which efficient dynamic programming techniques are available, since the building blocks of the computations in these methods are simply the sufficient statistics of the data.

The outline of the thesis is as follows: We first present the formal setting of the label sequence learning problem in Chapter 2. In this chapter, we also describe the details on CRFs and a perceptron algorithm for label sequences as well as the investigation of different loss function. Then the generalization of Boosting, Sequence AdaBoost, the generalization of SVM, HM-SVM and a GP sequence formulation, GPSC, are proposed in Chapters 3, 4 and 5 respectively. These chapters provide details on the objective functions, optimization methods, theoretical analysis and experimental evaluations of these techniques. Also, we provide an incremental comparison of these techniques at the end of each chapter. Finally, we conclude and suggest some future work in Chapter 6.

Chapter 2

Label Sequence Learning

In this chapter, we present the formal framework of the label sequence learning problem.

Label sequence learning can often be cast as a problem of inferring a Markov chain of label (or state) sequence from an observation sequence, where there is a one-to-one mapping between observations and labels. Hence, it is a generalization of the standard supervised classification problem where values of the random variables are predicted not only with respect to the observations but also with respect to values of the neighboring random variables. The goal is to learn a discriminant function for sequences, i.e. a mapping from observation sequences $\mathbf{x} = (x_1, x_2, \dots, x_t, \dots)$ to label sequences $\mathbf{y} = (y_1, y_2, \dots, y_t, \dots)$ with $x_t \in X, y_t \in \Sigma$. We call Σ, the label set of observations or the *micro-label set*, and $\mathcal{Y} = \Sigma^l$, the label set of observation sequences or the *macro-label set*, where l denotes the length of label sequence $\mathbf{y}$ [1]. Let $S = |\Sigma|$ be the size of the label set of each variable. Then, the size of the macro-label set scales exponentially with the length of observation sequences ($|\mathcal{Y}| = S^l$). We assume that a training set of labeled sequences $D \equiv \{(\mathbf{x}^i, \mathbf{y}^i) : i = 1, \dots, n\}$ where $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ is available to learn the mapping from $\mathbf{x}$ to $\mathbf{y}$. The training set, as well as the test set are assumed to be drawn i.i.d from an unknown, but fixed distribution $P(\mathbf{x}, \mathbf{y})$.

In a discriminative framework, the goal is to learn a mapping from $\mathcal{X}$ to $\mathcal{Y}$. In order to achieve this, one needs to describe three aspects of learning : a learning architecture, namely a parameterized family of discriminant functions, an objective function and finally an optimization method [2]. In Section 2.1, we describe the learning architecture

[1] For sequences of different length l, $\mathcal{Y}$ is going to be different. When clear from the context, we abbreviate $\mathcal{Y}^l$ as $\mathcal{Y}$.

[2] A related aspect is the regularization which is exploited in the following chapters.

that has been proposed by previous work and is adapted in this study. In Section 2.2, we present two loss function for label sequences, as well as some objective functions that motivate the objective functions used for label sequence learning problem. We then describe some optimization methods for these loss functions in Section 2.3.

2.1 Learning Architectures

A *learning architecture* specifies a family of Λ-parameterized discriminant functions $F(\mathbf{x}, \mathbf{y}; \Lambda)$ that assign a numerical score to pairs of observation/label sequences where:

$$F : \mathcal{X} \times \mathcal{Y} \rightarrow \Re. \tag{2.1}$$

One can think of $F(\mathbf{x}, \mathbf{y}; \Lambda)$ as measuring the *compatibility* between the observation sequence $\mathbf{x}$ and the label sequence $\mathbf{y}$. Each discriminant function F induces a mapping f,

$$f(\mathbf{x}; \Lambda) = \arg\max_{\mathbf{y}} F(\mathbf{x}, \mathbf{y}; \Lambda)\,, \tag{2.2}$$

where ties are arbitrarily broken. We initially restrict our attention to discriminant functions that are *linear* in some feature representation of $(\mathbf{x}, \mathbf{y})$, $\Psi(\mathbf{x}, \mathbf{y})$ [3].

$$\Psi : \mathcal{X} \times \mathcal{Y} \rightarrow \Re^d \tag{2.3}$$

is a feature map defined on the joint observation-label sequence space. Hence, F has the following general functional form:

$$F(\mathbf{x}, \mathbf{y}; \Lambda) = \sum_k \lambda_k \psi_k(\mathbf{x}, \mathbf{y}) = \langle \Lambda, \Psi(\mathbf{x}, \mathbf{y}) \rangle\,. \tag{2.4}$$

The model is assumed to be stationary. Thus, Λ parameters are shared by every position t in the sequence, resulting in the following formulation of the discriminative function:

$$F(\mathbf{x}, \mathbf{y}; \Lambda) = \sum_t \sum_k \lambda_k \psi_k(\mathbf{x}, \mathbf{y}; t) = \sum_t \langle \Lambda, \Psi(\mathbf{x}, \mathbf{y}; t) \rangle\,. \tag{2.5}$$

where $\Psi(\mathbf{x}, \mathbf{y}; t)$ is the same feature mapping, now restricted to positions in the sequence rather than the whole sequence. Then, one needs to define the features ψ_k to complete the design of the architecture. Our main design goal in defining Ψ is to make sure that f can be computed from F efficiently, i.e. using a Viterbi-like decoding algorithm.

[3] Features in maximum entropy modeling terminology are often referred as *weak learners* in the boosting community.

2.1.1 Feature Representation

Following previous work, we focus on extracting two types of features from a sequence pair $(\mathbf{x}, \mathbf{y})$. The first type of features capture inter-label dependencies between neighboring labels:

$$\psi_{s\sigma\bar{\sigma}}(\mathbf{x}, \mathbf{y}; t) = [\![y_s = \sigma]\!][\![y_t = \bar{\sigma}]\!], \tag{2.6}$$

where $\sigma, \bar{\sigma} \in \Sigma$ denote micro-labels and $[\![\cdot]\!]$ denotes the indicator function of the enclosed predicate. Here, $\psi_{s\sigma\bar{\sigma}}(\mathbf{x}, \mathbf{y}; t)$ corresponds to a feature $\psi_k(\mathbf{x}, \mathbf{y}; t)$ where k is indexed by $(s, \sigma, \bar{\sigma})$, the label σ at position s and the label $\bar{\sigma}$. These features simply indicate the statistics of how often a particular combination of labels occur at neighboring sites. We restrict the inter-label dependencies to neighboring labels (e.g. $s \in \{t+1, t+2\}$ for 3^{rd} order Markov features) to achieve our design goal, i.e. to ensure the availability of efficient dynamic programming algorithms.

The second type of features captures the dependency between a micro label and the observation sequence. First some observation attributes Φ that are relevant for the particular application are defined and then are combined with micro-labels. These features indicate the statistics of how often an attribute occurs with a particular label conjunctively

$$\psi_{sr\sigma}(\mathbf{x}, \mathbf{y}; t) = [\![y_t = \sigma]\!]\phi_r(x_s)\,. \tag{2.7}$$

Again $\psi_{sr\sigma}(\mathbf{x}, \mathbf{y}; t)$ corresponds to a feature $\psi_k(\mathbf{x}, \mathbf{y}; t)$ where k is indexed by (s, r, σ), the attribute r of the observation at position s and the label σ. For example, in Named-Entity Recognition, if $\phi_r(x)$ denotes whether the word x is 'Mr.', σ is the micro-label denoting the continuation of a person name, and s is the position before t, $s = t - 1$, then the feature $\psi_{sr\sigma}(\mathbf{x}, \mathbf{y}; t)$ denotes whether the t-th micro label in the label sequence $\mathbf{y}$ is a continuation of a person name and the previous word (the word at position $t-1$) is 'Mr.'

In the discriminative framework, we **condition on** the observation sequence (as opposed to *generating* it as in the generative framework). Hence, extracting features from arbitrarily past or future observations does not increase the complexity of the model. For this reason, there are no restrictions on the relationship of s and t. The features for which $s \neq t$ are usually called *overlapping* features (or *sliding window* features), since the same input attribute $\phi_a(x_s)$ is encoded in the model multiple times

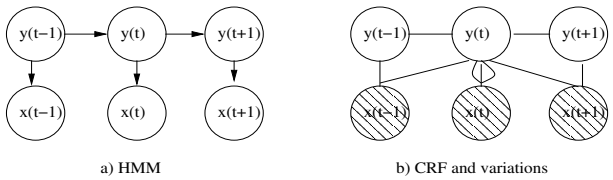

Figure 2.1: Graphical representation of HMMs and CRFs.

within different contexts. These features are used widely in discriminative sequence learning [Lafferty et al., 2001, McCallum et al., 2000, Collins, 2002a] and have been shown to improve classification accuracy in different applications.

The common use of overlapping features is by selecting an appropriate window size δ and defining a window centered at the current position, i.e. $s \in \{t-\delta, \ldots, t+\delta\}$. If each input is described by A attributes ϕ_a and if there are $S = |\Sigma|$ possible states, then one may extract a total of $(2\delta+1) \cdot S \cdot A$ features of this type by combining every input attribute with every micro-label for each position in the sliding window.

The features extracted at a position t are simply stacked together to form $\Psi(\mathbf{x}, \mathbf{y}; t)$. Many generalizations of this feature representation are possible, such as extracting different input features dependent of the relative distance $|t-s|$ in the chain. We use this representation as a proto-typical example.

Figure 2.1 illustrates the difference between the architecture of HMMs and the architecture described in this section, which is used in CRFs and the methods proposed in this study. Shaded areas indicate variables that the model conditions on. As the values of the variables are given, it is computationally inexpensive to use overlapping features in Figure 2.1b, which is not the case in HMMs.

2.1.2 Kernels for Label Sequences

To overcome the limitations of a linear discriminative function F, one may use kernel functions in order to construct and learn over Reproducing Kernel Hilbert Spaces (RKHS).

In some applications, features that are combinations of some attributes might be highly relevant. For example, in name entity recognition task an attribute *"The current word is* **United** *and the next word is* **States***."* is a more reliable predictor for the beginning of a location name than having two attributes *"The current word is* **United***."* and *"The next word is* **States***."*. If we consider only word features, then the number of

such n-gram features is n times of the size of the corpus, W. If features that fire often (such as capitalization, ending with some letter, etc.) are also included in the model, then the number of features would be some scalar c times nW, where $c < W$. If n, c and/or W is large, the computation and/or the storage of these higher order features explicitly may be intractable [4].

Kernel-based architectures provide an efficient alternative to parametric models with their ability to perform implicit non-linear mappings to a high dimensional space via kernel functions $K(x, \bar{x}) = \langle \tilde{\Phi}(x), \tilde{\Phi}(\bar{x}) \rangle$. In order to use kernels, we need methods that can work in a dual representation, where the data enters the computation only in the form of inner products of pairs of observations, which can in turn be replaced by the kernel function. SVMs and KLR are such methods and we investigate their generalizations to label sequence learning in Chapters 4 and 5 respectively.

The fundamental design decision for the kernel-based architecture is the engineering of the kernel function k that determines the kernel matrix $\mathbf{K}$ with entries $K_{(i,y),(j,y')} = k((\mathbf{x}^i, \mathbf{y}), (\mathbf{x}^j, \mathbf{y}'))$. As in the case of the primal form of features, our choice of kernel functions is restricted by computational constraints. We consider the kernel functions using which the discriminative function F can be computed efficiently.

Consider the kernel functions of the form

$$k((\mathbf{x}, \mathbf{y}), (\bar{\mathbf{x}}, \bar{\mathbf{y}})) = \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\bar{\mathbf{x}}, \bar{\mathbf{y}}) \rangle \tag{2.8}$$

The inner product of the 2^{nd} order Markov feature vectors $\Psi(\mathbf{x}, \mathbf{y})$ and $\Psi(\bar{\mathbf{x}}, \bar{\mathbf{y}})$ defined in Section 2.1.1 is given by $\sum_{s,t} [\![y_s = \bar{y}_t]\!] \langle \Phi(\mathbf{x}, s), \Phi(\bar{\mathbf{x}}, t) \rangle + \sum_{s,t} [\![y_s = \bar{y}_t \wedge y_{s+1} = \bar{y}_{t+1}]\!]$ where $\Phi(\mathbf{x}, s)$ denotes a vector of attributes of $\mathbf{x}$ that are extracted from a window of observations centered at position s. Kernelizing the inner product of the observation attributes, we propose to use the kernel function $k = k^1 + k^2$ for sequences, where

$$k^1((\mathbf{x}, \mathbf{y}), (\bar{\mathbf{x}}, \bar{\mathbf{y}})) \equiv \sum_{s,t} [\![y_s = \bar{y}_t]\!] g(\Phi(\mathbf{x}, s), \Phi(\bar{\mathbf{x}}, t)) \,, \tag{2.9a}$$

$$k^2((\mathbf{x}, \mathbf{y}), (\bar{\mathbf{x}}, \bar{\mathbf{y}})) \equiv \eta \sum_{s,t} [\![y_s = \bar{y}_t \wedge y_{s+1} = \bar{y}_{t+1}]\!], \tag{2.9b}$$

and g is a standard kernel function defined over observation patterns and $\eta > 0$. For example, when g is a polynomial kernel of degree 2, $g(x, \bar{x}) = (\langle x, \bar{x} \rangle + 1)^2$, the set of observation-label features induced by k^1 consists of all attributes $\Phi(\mathbf{x}, s)$ and their

[4] The situation is more severe when the features are real values and a Gaussian kernel is used, since this corresponds to having infinitely many features.

pairwise combinations conjoined with a particular label. Notice that k^1 couples observations in both sequences that share the same micro-labels at the respective positions, whereas k^2 counts the number of consecutive label pairs that are common in both label sequences (irrespective of the inputs). The scaling factor η in k^2 ensures that the inter-label dependency contributions are not overwritten by the observation-label dependency features, as it is common to have many observation attributes and g may increase their contribution, as in the polynomial kernel.

The following lemma provides some principles to construct complicated kernels from simpler ones:

Lemma 1 ([Cristianini and Shawe-Taylor, 2000] Proposition 3.12)**.** *Let K_1 and K_2 be kernels over $X \times X, X \subset \Re^n, a \in \Re^+, \Phi : X \to \Re^m$ with K_3 a kernel over $\Re^m \times \Re^m$. Then the following functions are kernels:*

1. $K(x, z) = K_1(x, z) + K_2(x, z)$
2. $K(x, z) = aK_1(x, z)$
3. $K(x, z) = K_3(\Phi(x), \Phi(z))$

Theorem 1. *The function k given in Equation 2.9 is symmetric positive semi-definite.*

Proof. k^2 is a kernel by the definition of a kernel, the inner product of a feature representation, and by Lemma 1.2. k^1 is also a kernel by the kernel definition and by Lemma 1.3 (by generating possibly infinite attributes of $(\mathbf{x}, s)$, joining it with the label value to form a new feature representation and taking the inner product of the new feature representation). Then, k is a kernel by Lemma 1.1. □

The rationale of performing kernelization only on the observations is the so-called *pre-image problem*: One can perform a non-linear feature mapping over observations using kernel functions during both learning and inference, since they are observed in both cases. However, nonlinear kernelization of labels results in the pre-image problem [Weston et al., 2002], namely the problem of projecting label sequences from high dimensional space (induced by the kernel function over labels) to the low dimensional (original) space during inference. This operation disturbs the linear nature of the search problem which enables the availability of the efficient Viterbi algorithm Thus kernelizing over labels leads to non-linear decoding schemes, which is problematic for sequences.

In order to avoid this problem, we restrict ourselves to linear feature maps over label sequences, but impose no restrictions on mappings over observation sequences.

One can generalize Equation 2.9 in various ways, for example by using higher order terms between micro-labels in both contributions, without posing major conceptual challenges. For the sake of presentation, we stick to Equation 2.9 and use it as a prototypical example of the more general setting.

The Representer Theorem [Kimeldorf and Wahba, 1971] describes the characteristics of optimal solutions for kernel-based methods:

Theorem 2. *Let $k : X * X \rightarrow Y$ be the kernel of the corresponding Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$. Given a training set $\{(x^1, y^1), \ldots, (x^n, y^n)\}$ where $x^i \in X$ and $y^i \in Y$, the solution $F \in \mathcal{H}$ of*

$$\min_{F \in \mathcal{H}} \sum_{i}^{n} \mathcal{L}(x^i, y^i, F) + \gamma \|F\|_{\mathcal{H}}^2 \tag{2.10}$$

where $\mathcal{L} : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \rightarrow \Re$ is a loss function measuring the discrepancy of $F(x^i)$ and y^i, is given by

$$F(x) = \sum_{i}^{n} \alpha_i k(x^i, x) \tag{2.11}$$

The most important consequence of the Representer Theorem is that one can find a finite set of parameters α_i to obtain the optimal solution of Equation 2.10, instead of dealing with the possibly infinite dimensional RKHS. This theorem holds for a variety of cost functions such as hinge loss and logarithmic loss, hence it applies to SVMs and kernel logistic regression. The Representer Theorem guarantees that the optimal solution of methods in which we use the kernelized architecture, namely HM-SVMs (Chapter 4) and GPS classification (Chapter 5), is of the following form:

$$F(\mathbf{x}^j, \mathbf{y}) = \sum_{i} \sum_{\bar{\mathbf{y}}} \alpha_{(i,\bar{\mathbf{y}})} k((\mathbf{x}^i, \bar{\mathbf{y}}), (\mathbf{x}^j, \mathbf{y})) = \alpha^T \mathbf{K} e_{(j,\mathbf{y})} \,. \tag{2.12}$$

where $e_{(j,\mathbf{y})}$ is the $(j, \mathbf{y})$-th unit vector. Equation 2.12 holds for suitably chosen α which depends on the objective function to be optimized. This Representer Theorem may not seem to be useful, since the number of α parameters scale exponentially with the length of sequence. However, in the following chapters, we make use of the dependency structure of $\alpha_{(i,.)}$ in order to reduce the number of these parameters.

2.2 Objective Functions

There is no single objective function for label sequences that would give the best performance in all situations. This choice rather depends on the specific application. In this thesis, a number of alternatives are investigated. We first point out some loss functions for sequence learning that are generalizations of the standard loss functions for variable classification. Unfortunately, it is NP-complete to optimize these functions. However, they are still worth investigating since they motivate the choice of the loss functions used in both previous work and this study.

2.2.1 Loss Functions for Sequences

Motivated from the standard zero-one classification loss, we define the empirical risk of label sequences for n training instances

$$\mathcal{R}^{\text{zo}}(\Lambda) = \frac{1}{n}\sum_{i=1}^{n}[\![f(\mathbf{x}^i;\Lambda) \neq \mathbf{y}^i]\!]\,. \tag{2.13}$$

A second risk function we consider is based on the ranking loss [Freund et al., 1998, Schapire and Singer, 1999] which measures the fraction of incorrect label sequences that are ranked higher than the correct one. Although ranking loss is more appropriate for ranking problems, it might also be advantageous over zero-one loss in classification. When perfect classification on the training data is not achievable, the ranking loss provides more information than the zero-one loss.

$$\mathcal{R}^{\text{rk}}(\Lambda, w) = \sum_{i=1}^{n}\sum_{\mathbf{y}\neq\mathbf{y}^i}[\![F(\mathbf{x}^i,\mathbf{y};\Lambda) \geq F(\mathbf{x}^i,\mathbf{y}^i;\Lambda)]\!]\,. \tag{2.14}$$

As the length of a sequence increases, the probability of achieving zero-loss on a sequence decreases exponentially (if the micro-label accuracy is not perfect). This motivates measuring the performance of a sequence classifier in terms of zero-one loss for individual labels. One might hope to design better classifiers by optimizing a loss function that is similar to the evaluation metric. Hence, we propose the Hamming risk [Schapire and Singer, 1999] which measures the zero-one loss for individual labels.

$$\mathcal{R}^{\text{hm}}(\Lambda) = \sum_{i=1}^{n}\frac{1}{l^i}\sum_{t=1}^{l^i}[\![f_t(\mathbf{x}^i;\Lambda) \neq y_t^i]\!]\,. \tag{2.15}$$

where $f_t(\mathbf{x}^i;\Lambda)$ is the t-th micro label of the label sequence given by $f(\mathbf{x}^i;\Lambda)$ and l^i is the length of the i$^t h$ sequence. Normalizing by this term leads to a loss between 0 and 1. Notice that Equation 2.15 reduces to the standard empirical misclassification risk, if the sequential nature of the data is ignored.

The three risk functions presented are discontinuous and non-convex in Λ, and are NP-complete to optimize. Moreover, minimizing the empirical risk alone is not sufficient to ensure good generalization. The methods discussed in this study can be understood as minimizing an upper bound on one of these risk functions, possibly combined with a regularization term. Below, we discuss two loss functions: the logarithmic loss and the marginal loss as upper bounds of $\mathcal{R}^{\text{zo}}$ and $\mathcal{R}^{\text{hm}}$ respectively.

2.2.2 Logarithmic Loss and Conditional Random Fields

We now present the conditional likelihood function optimized by Conditional Random Fields (CRFs) [Lafferty et al., 2001]. CRFs have been the state-of-the-art in label sequence learning till recently. They are a natural generalization of logistic regression to label sequences. The probability of a label sequence conditioning on an observation sequence is given by

$$p(\mathbf{y}|\mathbf{x};\Lambda) = \frac{1}{Z(\mathbf{x},\Lambda)} \exp\left[F(\mathbf{x},\mathbf{y};\Lambda)\right] , \tag{2.16}$$

where $Z(\mathbf{x},\Lambda) = \sum_{\mathbf{y}} \exp\left[F(\mathbf{x},\mathbf{y};\Lambda)\right]$ is a normalization constant. This distribution is in exponential normal form and the parameters Λ are also called *natural* or *canonical* parameters. The corresponding sufficient statistics are given by performing the sum over the sequence index t: $\sum_t \psi_k(\mathbf{x},\mathbf{y};t)$. If the features are restricted to binary values, these sufficient statistics simply count the number of times a feature ψ_k has been "active" along the labeled sequence $(\mathbf{x},\mathbf{y})$.

The objective in CRFs is the maximization of the conditional likelihood, or equivalently the minimization of the negative logarithmic loss (conditional log-likelihood):

$$\mathcal{R}^{\log}(\Lambda) = -\frac{1}{n}\sum_{i=1}^{n} \log p(\mathbf{y}^i|\mathbf{x}^i;\Lambda)\,. \tag{2.17}$$

When it is not regularized, the logarithmic loss is prone to overfitting, especially with noisy data. For this reason, it is common to penalize this objective function by adding a Gaussian prior (a term proportional to the squared norm $||\Lambda||^2$) as suggested in [Johnson et al., 1999, Chen and Rosenfeld, 1999] to avoid overfitting the training data.

The negative log-likelihood provides an upper bound on the empirical zero-one risk:

Proposition 1. $\mathcal{R}^{zo} \log 2 \leq \mathcal{R}^{log}$.

Proof. Distinguish two cases:
(i) $F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) > \max_{\mathbf{y} \neq \mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}; \Lambda)$ in which case $\log 2 [\![f(\mathbf{x}^i; \Lambda) \neq \mathbf{y}^i]\!] = 0 = -\log 1 \leq -\log p(\mathbf{y}^i | \mathbf{x}^i; \Lambda)$.
(ii) $F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) \leq \max_{\mathbf{y} \neq \mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}; \Lambda)$ in which case, since $p(\mathbf{y}^i | \mathbf{x}^i; \Lambda) \leq \frac{1}{2}$,
$\log 2 [\![f(\mathbf{x}^i; \Lambda) \neq \mathbf{y}^i]\!] = -\log \frac{1}{2} \leq -\log p(\mathbf{y}^i | \mathbf{x}^i; \Lambda)$.
Summing over all i completes the proof. □

2.2.3 Marginal Loss

In many applications, one is primarily interested in the per-label zero-one loss or *Hamming loss* [Schapire and Singer, 1999]. Here we propose a logarithmic Hamming loss function for label sequences. Following [Kakade et al., 2002, Altun et al., 2003a], one can define a logarithmic risk based on the marginal probabilities of the individual label variables y_t^i:

$$\mathcal{R}^{\text{mg}}(\Lambda) \quad = \quad -\sum_{i=1}^{n} \frac{1}{l^i} \sum_{t=1}^{l^i} \log p(y_t^i | \mathbf{x}; \Lambda) \tag{2.18}$$

where $p(y_t^i | \mathbf{x}; \Lambda) = \sum_{\mathbf{y}: y_t = y_t^i} p_\Lambda(\mathbf{y} | \mathbf{x}^i)$. This is a non-convex function and defines an upper bound on the Hamming risk $\mathcal{R}^{\text{hm}}$, if one uses a pointwise decoding function

Proposition 2. *With* $f_t(\mathbf{x}; \Lambda) = \arg\max_\sigma p(Y_t = \sigma | \mathbf{x}; \Lambda)$*, the following bound holds:* $\log 2 \cdot \mathcal{R}^{hm}(\Lambda) \leq \mathcal{R}^{mg}(\Lambda)$.

Proof. Distinguish two cases:
(i) $f_t(\mathbf{x}) = y_t$, then $[\![f_t(\mathbf{x}) \neq y_t]\!] = 0 \leq -\log 1 \leq -\log p(y_t | \mathbf{x}; \Lambda)$
(ii) $f_t(\mathbf{x}) \neq y_t$, then $\log 2 [\![f_t(\mathbf{x}) \neq y_t]\!] = \log 2 \leq -\log p(y_t | \mathbf{x}; \Lambda)$, since $Pr(Y_t = y_t | \mathbf{x}; \Lambda) \leq \frac{1}{2}$, because there exists $\sigma \neq y_t$ such that $p(Y_t = y_t | \mathbf{x}; \Lambda) \leq p(Y_t = \sigma | \mathbf{x}; \Lambda)$ and $p(Y_t = y_t | \mathbf{x}; \Lambda) + p(Y_t = \sigma | \mathbf{x}; \Lambda) \leq 1$. The claim follows by summing over all sequences and all positions in sequences and applying the bound to every term individually. □

2.3 Optimization Methods

The method of optimization might have a crucial role in the effectiveness of a classification method. In general, an optimization method which converges fast, results

in sparse solutions or that can handle a large number of features or large data might have advantages over other optimization methods. In some cases, an approximate optimization that is more efficient in one of these aspects might be preferred over an exact method, if they have similar accuracy. It might also be that an approximation method performs better than the exact method. For example, sparse greedy methods perform regularization inherently, which might lead to better generalization properties than performing regularization by adding a term to an objective function that is prone to overfitting.

Below, we present two optimization methods for the logarithmic loss of CRFs (Section 2.3.1), and one method for optimizing the marginal loss (Section 2.3.2).

2.3.1 Optimization of Conditional Random Fields

Exact Optimization of CRFs

A number of algorithms have been proposed to estimate Λ that minimize (2.17) exactly. These include iterative scaling [Lafferty et al., 2001] and various flavors of conjugate gradient descent and second order methods [Wallach, 2002, Sha and Pereira, 2003, Minka, 2001a]. Sha and Pereira [2003] show that Limited-Memory BFSG (L-BFSG), a quasi-Newton method, is more efficient for minimizing the convex loss function in (2.17) than the other methods in shallow parsing.

The gradient of the negative log-likelihood is given by:

$$\nabla_\theta \mathcal{R}^{\log} = \sum_i \mathbf{E}\left[\sum_t \psi(\mathbf{x},\mathbf{y};t)|\mathbf{x}=\mathbf{x}^i\right] - \sum_t \psi(\mathbf{x}^i,\mathbf{y}^i;t) \tag{2.19}$$

where the expectations are computed w.r.t. $p(\mathbf{y}|\mathbf{x};\Lambda)$. The stationary equations then simply state that the observed sufficient statistics should match their conditional expectations. Computationally, the evaluation of $\sum_t \psi(\mathbf{x}^i,\mathbf{y}^i;t)$ is straightforward counting, while summing over all sequences $\mathbf{y}$ to compute $\mathbf{E}\left[\sum_t \psi(\mathbf{x},\mathbf{y};t)|\mathbf{x}=\mathbf{x}^i\right]$ can be performed using dynamic programming techniques, namely forward-backward algorithm [Manning and Schütze, 1999], since the dependency structure between labels is a simple chain.

Approximate Optimization of CRFs: Hidden Markov Perceptron Learning

Computing the gradient of the logarithmic loss function, (2.19) might be computationally expensive if the corpus is very large, since one has to compute the expectations of

features for every training example. In some cases, a single example might be as informative as the entire corpus to update the parameters. Then, an online algorithm that updates the model using a single training example may converge substantially faster than a batch algorithm. If the distribution is peaked, the contribution of the most likely label dominates the expectation values. Assuming this, the so-called *Viterbi assumption*, one can compute a good approximation of the gradients by considering only the most likely label sequence according to the current model. The following online perceptron algorithm for label sequence learning (Algorithm 1), proposed by [Collins, 2002a], makes use of these two approximations:

Algorithm 1 Hidden Markov Perceptron algorithm.

1: Initialize $\Lambda_0 = \vec{0}$
2: **repeat**
3: **for** all training patterns $\mathbf{x}^i$ **do**
4: Compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} F(\mathbf{x}^i, \mathbf{y}; \Lambda_t)$
5: **if** $\mathbf{y}^i \neq \hat{\mathbf{y}}$ **then**
6: $\Lambda_{t+1} \leftarrow \Lambda_t + \psi(\mathbf{x}^i, \mathbf{y}^i) - \psi(\mathbf{x}^i, \hat{\mathbf{y}})$
7: **end if**
8: **end for**
9: **until** convergence or a maximum number of iterations reached.

At each iteration, Hidden Markov Perceptron (HM-Perceptron) computes an approximation of (2.19) using only the current training instance $\mathbf{x}^i$ (as opposed to the entire corpus) and the label sequence $\hat{\mathbf{y}}$ that achieves the highest score for $\mathbf{x}^i$ (as opposed to all possible label sequences). $\hat{\mathbf{y}}$ can be found by Viterbi decoding [Manning and Schütze, 1999]. In order to avoid fluctuations, one can average over the iterations of the perceptron algorithm as described in [Freund and Schapire, 1998]. Ciaramita and Johnson [2003] present a space-efficient implementation of the average perceptron algorithm.

Although we describe HM-Perceptron as an approximation algorithm of CRFs, it was originally proposed as a generalization of the perceptron algorithm, which allows the margin interpretation and leads to online mistake bounds due to Novikoff's Theorem [Novikoff, 1963], as discussed in Section 4.5.

Pitch Accent	$\mathcal{R}^{\log}$	$\mathcal{R}^{\mathrm{mg}}$	HM-Perceptron
Accuracy	76.62	76.40	76.55

Table 2.1: Per-label accuracy of Pitch Accent Prediction on CRFs, $\mathcal{R}^{\mathrm{mg}}$ and HM-Perceptron with window size 5.

2.3.2 Optimization of Marginal Loss

We propose using standard numerical optimization procedures such as L-BFSG to optimize $\mathcal{R}^{\mathrm{mg}}$. The gradient is given by

$$\nabla_\theta \mathcal{R}^{\mathrm{mg}} = -\sum_i \sum_t \left(\mathbf{E}\left[\psi(\mathbf{x}, \mathbf{y}; t) | \mathbf{x} = \mathbf{x}^i, y_t = y_t^i\right] - \mathbf{E}\left[\psi(\mathbf{x}, \mathbf{y}; t) | \mathbf{x} = \mathbf{x}^i\right] \right) \quad (2.20)$$

Equation 2.20 indicates that the expected sufficient statistics are now compared not to their empirical values, but to their expected values, conditioned on a given label value y_t^i. In order to evaluate these expectations, one can perform dynamic programming using the algorithm described in [Kakade et al., 2002], which has (independently of our work) also focused on the use of Hamming loss functions for sequences. This algorithm has the complexity of the forward-backward algorithm scaled by a constant.

2.4 Experiments

We experimentally evaluated the objective functions described above on one speech application, pitch accent prediction, and on two NLP applications, Named Entity Recognition (NER) and part-of-speech tagging (POS). The description of these applications and the features used can be found in Appendix B.

We ran experiments comparing the two loss functions $\mathcal{R}^{\log}$ (Equation 2.17) and $\mathcal{R}^{\mathrm{mg}}$ (Equation 2.18) using both BFSG. We also evaluated the performance of average HM-Perceptron algorithm. The regularization constants of both $\mathcal{R}^{\log}$ and $\mathcal{R}^{\mathrm{mg}}$ were adjusted using development data. For $\mathcal{R}^{\mathrm{mg}}$, in order to account for the risk of local minima, we iteratively restarted the optimization from a point that is slightly (and randomly) perturbed from the optimal point of the last optimization step until the value of the objective function does not improve.

The results summarized in Tables 2.1, 2.2 and 2.3 demonstrate the competitiveness of the $\mathcal{R}^{\mathrm{mg}}$ with respect to $\mathcal{R}^{\log}$. We observe that in different applications the

NER	$\mathcal{R}^{\log}$	$\mathcal{R}^{\mathrm{mg}}$	HM-Perceptron
F1	74.83	74.17	73.41

Table 2.2: F1 measure of NER on Spanish newswire corpus on CRFs, $\mathcal{R}^{\mathrm{mg}}$ and HM-Perceptron with window size is 3.

POS	$\mathcal{R}^{\log}$	$\mathcal{R}^{\mathrm{mg}}$	HM-Perceptron
Accuracy	95.68	95.71	95.91

Table 2.3: Per-label accuracy of POS tagging on PennTreeBank on CRFs, $\mathcal{R}^{\mathrm{mg}}$ and HM-Perceptron.

ordering of the performance of the two loss functions changes. The performance of average perceptron varies as well. In pitch accent predict and NER, average perceptron performs slightly worse than CRF. One might consider this as not surprising, since HM-Perceptron is an approximation of CRF. However, it performs better than CRFs in POS tagging. This result is consistent with the results in [Collins, 2002a] and it can be explained by the margin interpretation of HM-Perceptron and the generalization properties of the perceptron algorithm.

2.5 Discussion

The experimental results are consistent with our intuition that there is no single objective function resulting in the best performance in all situations.

We have observed that the marginal loss gives better performance for POS tagging whereas log-loss gives better performance for pitch accent prediction and NER. For the rest of this thesis, we focus on objective functions that are defined over the entire sequence rather than the ones based on the Hamming loss.

We also observed that an approximation algorithm, HM-Perceptron, can perform better than the exact algorithm for some applications and maybe preferable over exact optimization of CRF on other applications, since it is very easy to implement, computational much more inexpensive and results in similar performance.

In the next three chapters, we present generalizations of Boosting, SVMs and kernel logistic regression to label sequence learning problem based on the framework presented

in this chapter. The advantages of each of these methods over the ones described in this chapter are also discussed.

Chapter 3

Sequence Boosting

Boosting [Freund and Schapire, 1997] is one of the most powerful learning ideas introduced to the machine learning community in the last decade. It is a discriminative learning technique to find a highly accurate classifier, an *ensemble*, by combining many simple, low-accuracy classifiers or *weak learners*. AdaBoost is the best known and most popular boosting algorithm and enjoys the advantages of the sparse feature representation.

In this chapter, we present a generalization of AdaBoost to label sequence learning, Sequence AdaBoost [Altun et al., 2003a]. Sequence AdaBoost has all the advantages of discriminative methods for sequence learning as well as its implicit feature selection property. This is important for efficient inference in high dimensional feature spaces and model regularization. Sequence AdaBoost also makes use of the availability of a dynamic programming formulation for efficient learning and inference due to the Markov chain dependency structure between labels.

After a short overview of Boosting (Section 3.1), we define the objective function of Sequence Boosting in Section 3.2. Then two optimization methods of Sequence Boosting, gradient based optimization and Sequence Adaboost, are described in Section 3.3. We present generalization properties of Sequence AdaBoost in Section 3.4 and briefly mention related work in Section 3.5. We conclude this chapter with experimental results (Section 3.6) and discussions (Section 3.7).

3.1 Boosting

Since Boosting is a very well known method in the machine learning community, we only present a brief overview in this section.

Given a (possibly infinite) set of weak learners (features) $\psi_i \in \mathcal{H}$ and a training set of n labeled examples (x^i, y^i) with $y^i \in \{1, \ldots, S\}$ drawn i.i.d. from an unknown, but fixed distribution $P(x, y)$, the goal of Boosting is to find a weight vector Λ for a linear combination of weak learners:

$$F(x, y; \Lambda) = \sum_r \lambda_r \psi_r(x, y) \tag{3.1}$$

The label of a new observation is found by simply performing a max operation on F over all possible labels.

The best known algorithm of Boosting is AdaBoost (Algorithm 2). We describe AdaBoost.MR, a multiclass version of AdaBoost based on the ranking loss [Schapire and Singer, 1999].

Algorithm 2 AdaBoost.MR

1: Initialize $D_0(i, y) = 1/(n(S-1)), \forall i, y \neq y^i; D_0(i, y^i) = 0, \forall i$.
2: Initialize $\Lambda = 0$
3: **for** $r = 1, \ldots, R$ **do**
4: Pick a weak learner ψ_k
5: Compute optimal increment $\triangle\lambda_k$
6: Update weight $\lambda_k \leftarrow \lambda_k + \triangle\lambda_k$
7: Update D_{r+1}:

$$D_{r+1}(i, y) = \frac{D_r(i, y) \exp\left(\frac{1}{2}\lambda_k(\psi_k(x^i, y) - \psi_k(x^i, y^i))\right)}{Z_r} \tag{3.2}$$

8: **end for**

AdaBoost.MR is an iterative algorithm. At each round, the goal is to find the weak learner that minimizes the rank loss:

$$\mathcal{R}^{\text{rnk}}(\Lambda) \quad = \sum_{i=1}^{n} \sum_{y \neq y^i} D(i, y) [\![F(x^i, y; \Lambda) \geq F(x^i, y^i; \Lambda)]\!] \tag{3.3}$$

where D is a distribution over (x^i, y) pairs $\forall i \in \{1, \ldots, S\}, \forall y$. This is achieved by finding the weak learner ψ_k that minimizes Z_r at each round, since the rank loss is upper bounded by Z_r for all r. Then the optimal weight update of ψ_k is found by:

$$\triangle\lambda_k = \frac{1}{2} \ln\left(\frac{1 - \epsilon_k}{\epsilon_k}\right) \tag{3.4}$$

where $\epsilon_k = P_{i\sim D_r}[\text{argmax}_y\, \psi_k(x^i, y) \neq y^i]$ is the error of weak learner ψ_k at round r. This serial update technique usually leads to a solution with only few non-zeros parameters and therefore results in a sparse feature representation.

Maintaining a weight distribution over training examples is a key idea of Boosting. This distribution is updated at every round, so that more emphasis is placed on the misclassified examples, Equation Equation 3.2. Z_r is a normalization constant chosen so that D_{r+1} is a distribution.

The choice of the optimal weight update as well as the distribution over the training examples is based on the loss function optimized by AdaBoost, i.e. the exponential loss:

$$\mathcal{R}(\Lambda) = \sum_{i=1}^{n} \sum_{y \neq y^i} D(i, y) e^{F(x^i, y; \Lambda) - F(x^i, y^i; \Lambda)} \tag{3.5}$$

Early studies suggested that boosting is a very robust method. This has been theoretical explained by the observation that it maximizes the ensemble margin [Schapire et al., 1998]. However, more thorough experimental studies showed that boosting can, in fact, overfit very badly with noisy data. Different regularization methods have been proposed, such as bounding the Λ parameters from above and below [Mannor et al., 2003], or adding a smoothing constant to the numerator and denominator in Equation Equation 3.4 which leads to limiting the magnitude of the λ values as well [Schapire and Singer, 1999].

3.2 Objective Function

Following Schapire et al. [1998], we propose the following objective function for label sequences:

$$\mathcal{R}^{\exp}(\Lambda, w) \equiv \sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}^i} D(i, \mathbf{y}) e^{F(\mathbf{x}^i, \mathbf{y}; \Lambda) - F(\mathbf{x}^i, \mathbf{y}^i; \Lambda)} \tag{3.6}$$

Proposition 3. *The exponential risk is an upper bound on the weighted version of the rank risk, Equation 2.14:*

$$\sum_{i=1}^{n} \sum_{\mathbf{y} \neq \mathbf{y}^i} D(i, \mathbf{y}) [\![F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq F(\mathbf{x}^i, \mathbf{y}^i; \Lambda)]\!] \leq \mathcal{R}^{exp}$$

Proof. (i) If $F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) > F(\mathbf{x}^i, \mathbf{y}; \Lambda)$ then $[\![F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq F(\mathbf{x}^i, \mathbf{y}^i; \Lambda)]\!] = 0 \leq e^z$ for any z.
(ii) Otherwise, $[\![F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq F(\mathbf{x}^i, \mathbf{y}^i; \Lambda)]\!] = 1 = e^0 \leq e^{F(\mathbf{x}^i, \mathbf{y}; \Lambda) - F(\mathbf{x}^i, \mathbf{y}^i; \Lambda)}$.
Performing a weighted sum over all instances and label sequences $\mathbf{y}$ completes the proof. □

Interestingly, when D is uniform for all $\mathbf{y}$ of each training instance $\mathbf{x}^i$, this loss function is simply the weighted inverse conditional probability of the correct label sequence Equation 2.16, as pointed out by Lafferty et al. [2001].

$$\mathcal{R}^{\exp}(\Lambda, w) = \sum_i D(i) \left[\frac{1}{p(\mathbf{y}^i | \mathbf{x}^i; \Lambda)} - 1 \right]. \tag{3.7}$$

Another way of writing the exponential loss function is using the odds ratio which reveals the relationship of Sequence Boost and HM-SVM as discussed in Section 4.7. Let O be the odds ratio between two label sequences $\mathbf{y}$ and $\bar{\mathbf{y}}$. For a given observation sequence $\mathbf{x}$, the odds ratio of two label sequences w.r.t. Λ is given by:

$$O(\mathbf{x}, \mathbf{y}, \bar{\mathbf{y}}; \Lambda) = \frac{p(\mathbf{y}|\mathbf{x}; \Lambda)}{p(\bar{\mathbf{y}}|\mathbf{x}; \Lambda)} \tag{3.8}$$

$$= \exp[\langle \Lambda, \sum_t \Psi(\mathbf{x}, \mathbf{y}; t) - \Psi(\mathbf{x}, \bar{\mathbf{y}}; t) \rangle] \tag{3.9}$$

Then, the exponential loss is simply the sum of odds ratio of the correct label sequence and all the incorrect label sequences weighted w.r.t. their lengths and our goal is to find Λ^* which minimizes the exponential loss:

$$\mathcal{R}^{\exp}(\Lambda, w) \equiv \sum_i D(i) \sum_{\mathbf{y} \neq \mathbf{y}^i} O(\mathbf{x}^i, \mathbf{y}^i, \mathbf{y}; \Lambda) \tag{3.10}$$

$$\Lambda^* = \operatorname*{argmin}_{\Lambda} \mathcal{R}^{\exp}(\Lambda, w) \tag{3.11}$$

What should be the form of $D(i)$? We consider distributions where each training instance is weighted with respect to its length in order to account for varying sequence length:

$$D(i) = \frac{w(l^i)}{\sum_i w(l^i)} \tag{3.12}$$

where l^i is the length of the i-th sequence. In the case of noisy data, the exponentiated difference between the correct and incorrect sequence scores (Equation 3.11) might be extremely large, scaling exponentially with the length of the sequence. Then, long

sequences might dominate the loss function and leave shorter sequences ineffective. To overcome this problem, we consider weighting functions $\log w(l^i) = l^i \log \pi$ with $\pi \in (0; 1]$ as plausible candidates.

3.3 Optimization Methods

$\mathcal{R}^{\exp}$ is a convex function. We first discuss an exact optimization method using standard gradient based methods, then present the Sequence AdaBoost algorithm.

3.3.1 Exact Optimization

In order to derive the gradient equations for the exponential loss, we can simply make use of the following elementary facts:

$$\nabla_\Lambda(-\log p(\Lambda)) = -\frac{\nabla_\Lambda p(\Lambda)}{p(\Lambda)}, \text{ and } \nabla_\Lambda \frac{1}{p(\Lambda)} = -\frac{\nabla_\Lambda p(\Lambda)}{p(\Lambda)^2} = \frac{\nabla_\Lambda(-\log p(\Lambda))}{p(\Lambda)}\,. \quad (3.13)$$

Then it is easy to see that

$$\nabla_\Lambda \mathcal{R}^{\exp}(\Lambda, w) \;=\; \sum_i \frac{\mathbf{E}\left[\sum_t \Psi(\mathbf{x}, \mathbf{y}; t)|\mathbf{x} = \mathbf{x}^i\right] - \sum_t \Psi(\mathbf{x}^i, \mathbf{y}^i; t)}{p(\mathbf{y}^i|\mathbf{x}^i; \Lambda)} D(i)\,. \quad (3.14)$$

The only difference between the gradient of $\mathcal{R}^{\log}$, (2.19), and the gradient of $\mathcal{R}^{\exp}$, (3.14), is the non-uniform weighting of different sequences by their inverse probability, hence putting more emphasis on training label sequences that receive a small overall (conditional) probability. In order not to emphasize long sequences excessively, the contribution of each training example is also weighted w.r.t. its length. In the absence of $w(l)$ term, $\nabla_\Lambda \mathcal{R}^{\exp}$ can be dominated by very long sequences since $p(\mathbf{y}^i|\mathbf{x}^i; \Lambda)$ decreases exponentiall with the length of the sequence.

Given the gradient equations, we can solve the optimization problem using a 1^{st} order gradient method, such as a Quasi-Newton method. Since these optimization techniques update all the parameters in parallel, the resulting classifier lacks sparseness properties. We now propose Sequence AdaBoost which is a optimization method for Sequence Boosting with the advantages of feature sparseness.

3.3.2 Sequence Boosting

As an alternative to a simple gradient method, we derive a boosting algorithm that generalizes the AdaBoost.MR algorithm for multiclass classification [Schapire and Singer,

1999] to label sequence learning. This is an approximation algorithm (optimizing a bound on the exponential loss function) with sequential updates. Due to this serial update technique, the algorithm usually stops after updating only a small subset of all parameters, therefore achieves a sparse feature representation.

Sequence AdaBoost is very similar to AdaBoost.MR. It forms a linear combination of weak learners, an ensemble, by selecting a weak learner and its weight parameter at each round. It also maintains a weight distribution of observation-label sequence pairs and updates this distribution after every round of boosting.

Weight distribution of examples

We use $\mathcal{Y}^l$ to denote all possible label sequences (*macro-label set*) of length l and define a sequence of distributions $D_r(i, \mathbf{y})$ over $(\mathbf{x}^i, \mathbf{y})$ pairs recursively as follows:

$$D_{r+1}(i, \mathbf{y}) \equiv \frac{D_r(i, \mathbf{y})}{Z_r} e^{\triangle\lambda_k(\sum_t \psi_k(\mathbf{x}^i, \mathbf{y}; t) - \psi_k(\mathbf{x}^i, \mathbf{y}^i; t))}. \tag{3.15}$$

Here $k = k(r)$ denotes the feature selected in the r-th round, $\triangle\lambda_k$ is the optimal weight update ψ_k and Z_r is the normalization constant. We initialize $D_0(i, \mathbf{y}) = \frac{w(l^i)}{(|\mathcal{Y}^{l^i}|-1)\sum_j w(l^j)}$ and $D_0(i, \mathbf{y}^i) = 0$ for all i. Let $\triangle\Lambda_k$ be a vector of zeros and the value $\triangle\lambda_{k(r)}$ at the $k(r)^{th}$ position. After R rounds of boosting, the parameter vector is given by $\Lambda = \sum_{r=1}^{R} \triangle\Lambda_k$. Then, at any round, the distribution is given by:

$$\begin{aligned} D(i, \mathbf{y}) &\equiv w(l^i) \frac{\exp\left[F(\mathbf{x}^i, \mathbf{y}) - F(\mathbf{x}^i, \mathbf{y}^i)\right]}{\sum_j w(l^j) \sum_{\mathbf{y}' \neq \mathbf{y}^j} \exp\left[F(\mathbf{x}^j, \mathbf{y}') - F(\mathbf{x}^j, \mathbf{y}^j)\right]}, \quad \forall \mathbf{y} \neq \mathbf{y}^i & (3.16) \\ &= D(i)\pi_i(\mathbf{y}) & (3.17) \end{aligned}$$

where

$$\begin{aligned} D(i) &\equiv \frac{w(l^i)\left[p(\mathbf{y}^i|\mathbf{x}^i; \Lambda)^{-1} - 1\right]}{\sum_j w(l^j)\left[p(\mathbf{y}^j|\mathbf{x}^j; \Lambda)^{-1} - 1\right]} & (3.18) \\ \pi_i(\mathbf{y}) &\equiv \frac{p(\mathbf{y}|\mathbf{x}^i; \Lambda)}{1 - p(\mathbf{y}^i|\mathbf{x}^i; \Lambda)}, & (3.19) \end{aligned}$$

Equation 3.17 shows how we can split $D(i, \mathbf{y})$ into a relative weight for each training instance, given by $D(i)$, and a relative weight of each sequence, given by $\pi_i(\mathbf{y})$, the re-normalized conditional probability $p(\mathbf{y}|\mathbf{x}^i; \Lambda)$. Notice that $D(i) \to 0$ as we approach the perfect prediction case of $p(\mathbf{y}^i|\mathbf{x}^i; \Lambda) \to 1$.

Proposition 4. *For any number of rounds R, $\mathcal{R}^{rk} \leq \prod_{r=1}^{R} Z_r$.*

Proof. Schapire and Singer [1999, Theorem 6] □

Selecting optimal feature and its weight update

Since optimizing the rank loss is NP complete, our goal is to minimize its upper bound, the partition function or weight normalization constant Z_r as in standard AdaBoost. We define a boosting algorithm which aims at minimizing Z_r w.r.t. a weak learner $\psi_{r(k)}$ and its corresponding optimal weight update $\triangle\lambda_{r(k)}$ at each round. In order to simplify the notation, we drop the round index r. Let $\psi = \psi_{r(k)}$ and $\triangle\lambda = \triangle\lambda_{r(k)}$. Then,

$$\begin{aligned} Z(\triangle\lambda) &\equiv \sum_i D(i) \sum_{\mathbf{y}\neq\mathbf{y}^i} \pi_i(\mathbf{y}) \exp\left[\triangle\lambda \left(\sum_t \psi(\mathbf{x}^i, \mathbf{y}; t) - \psi(\mathbf{x}^i, \mathbf{y}^i; t)\right)\right] \\ &= \sum_b \left(\sum_i D(i) p(b|\mathbf{x}^i; \Lambda, \psi)\right) \exp\left[b\triangle\lambda\right], \end{aligned} \tag{3.20}$$

where $p(b|\mathbf{x}^i; \Lambda, \psi) = \sum_{\mathbf{y}\in\mathbf{y}(b;\mathbf{x}^i,\psi)} \pi_i(\mathbf{y})$ and $\mathbf{y}(b; \mathbf{x}^i, \psi) \equiv \{\mathbf{y} : \mathbf{y} \neq \mathbf{y}^i \wedge (\sum_t \psi(\mathbf{x}^i, \mathbf{y}; t) - \psi(\mathbf{x}^i, \mathbf{y}^i; t)) = b\}$. This minimization problem is only tractable if the number of features is small, since a dynamic programming run with accumulators [Lafferty et al., 2001] for every feature is required in order to compute the probabilities $p(b|\mathbf{x}^i; \Lambda, \psi)$, i.e. the probability for the feature ψ to be active exactly ξ times, conditioning on the observation sequence $\mathbf{x}^i$, where ξ is the sum of b and the number of times ψ is observed in $(\mathbf{x}^i, \mathbf{y}^i)$.

In cases, where this is intractable (and we assume this is the case in most applications), one can instead minimize an upper bound on Z. To find such a bound we exploit the convexity of the exponential function and use the following inequality valid for x, $x_0 \leq x \leq x_1$, that is linear in x,

$$e^x \leq e^{x_0} \frac{x_1 - x}{x_1 - x_0} + e^{x_1} \frac{x - x_0}{x_1 - x_0}, \tag{3.21}$$

Let $\mathbf{u}(\mathbf{x}^i, \mathbf{y}) \equiv \sum_t \Psi(\mathbf{x}^i, \mathbf{y}; t) - \Psi(\mathbf{x}^i, \mathbf{y}^i; t)$, where $u_k(\mathbf{x}^i, \mathbf{y})$ measures the difference between the sufficient statistics of feature k in the correct label-observation sequence pair $(\mathbf{x}^i, \mathbf{y}^i)$ and a label-observation sequence pair $(\mathbf{x}^i, \mathbf{y})$. Let

$$\mathbf{U} = \max_{i,\mathbf{y}} \mathbf{u}(\mathbf{x}^i, \mathbf{y}) \tag{3.22}$$

$$\mathbf{L} = \min_{i,\mathbf{y}} \mathbf{u}(\mathbf{x}^i, \mathbf{y}) \tag{3.23}$$

$$\text{where} \qquad L_k \leq u_k(\mathbf{x}^i, \mathbf{y}) \leq U_k, \quad \forall k, i, \mathbf{y} \tag{3.24}$$

As before, fixing r let $u(\mathbf{x}^i, \mathbf{y}) = u_{k(r)}(\mathbf{x}^i, \mathbf{y})$, $U = U_{k(r)}$ and $L = L_{k(r)}$. These definitions allow us to upper bound Z:

$$\begin{aligned} Z(\triangle\lambda) &= \sum_i D(i) \sum_{\mathbf{y}\neq\mathbf{y}^i} \pi_i(\mathbf{y}) \exp\left[\triangle\lambda u(\mathbf{x}^i, \mathbf{y})\right] && (3.25) \\ &\leq \sum_i D(i) \sum_{\mathbf{y}\neq\mathbf{y}^i} \pi_i(\mathbf{y}) \left[\frac{U - u(\mathbf{x}^i, \mathbf{y})}{U - L} e^{\triangle\lambda L} + \frac{u(\mathbf{x}^i, \mathbf{y}) - L}{U - L} e^{\triangle\lambda U}\right] && (3.26) \\ &= s e^{\triangle\lambda L} + (1 - s) e^{\triangle\lambda U}, \text{ where} && (3.27) \\ s &\equiv \sum_i D(i) \sum_{\mathbf{y}\neq\mathbf{y}^i} \pi_i(\mathbf{y}) \frac{U - u(\mathbf{x}^i, \mathbf{y})}{U - L} && (3.28) \end{aligned}$$

Here, $0 \leq s \leq 1$ can be interpreted as a measure of correlation of Ψ and the correct label sequences with respect to the distribution $D_r(i, \mathbf{y})$. We call s the *pseudo-accuracy* of the feature ψ.

Taking the derivative and enforcing stationarity, we get the analytical solution of $\triangle\lambda$ that optimizes the upper bound on Z:

$$\triangle\lambda = \frac{1}{U - L} \log\left(\frac{-Ls}{U(1 - s)}\right). \tag{3.29}$$

Plugging Eq.Equation 3.29 into the upper bound gives:

$$Z \leq \left(\frac{-Ls}{U(1 - s)}\right)^{\frac{L}{U-L}} s\frac{U - L}{U} \tag{3.30}$$

Notice that this formulation is a generalization of standard Boosting. In boosting, it is usually assumed that $\psi_k(\mathbf{x}, \mathbf{y}) \in \{-1, 1\}$ or $\psi_k(\mathbf{x}, \mathbf{y}) \in [-1, 1]$ as in [Schapire and Singer, 1999]. The bound on the exponential function for $x_0 = L = -1$ and $x_1 = U = 1$ is given by:

$$e^{\alpha x} \leq \frac{1}{2} e^{-\alpha}(1 - x) + \frac{1}{2} e^{\alpha}(1 + x) \tag{3.31}$$

leading to the optimal update $\triangle\lambda$:

$$\triangle\lambda = \frac{1}{2} \log\left(\frac{s}{1 - s}\right). \tag{3.32}$$

which is a special form of Equation 3.29. Here, s corresponds to W_+ in the notation of Schapire and Singer [1999]. In that case Equation 3.30 rewrites as $Z \leq 2\sqrt{s(1 - s)}$.

In sequence learning, U_k and $-L_k$ can be as large as the length of a sequence if the feature ψ_k occurs frequently. The features that capture the inter-dependency of

micro labels are such features. Then, the bound in Equation 3.26 can be very loose, especially when there exists a very long sequence in the training set. One can get a better approximation by defining a bound for each sequence using the fact that the contribution of each sentence to the loss function is combined linearly. Let

$$\mathbf{U}^i = \max_{\mathbf{y}} \mathbf{F}(\mathbf{x}^i, \mathbf{y}) \tag{3.33}$$

$$\mathbf{L}^i = \min_{\mathbf{y}} \mathbf{F}(\mathbf{x}^i, \mathbf{y}) \tag{3.34}$$

Thus, the union of the intervals $[L_k^i; U_k^i]$ for every training sequence i gives $[L_k; U_k]$ for all feature ψ_k. We can define a tighter bound on Z as:

$$\begin{aligned} Z(\triangle\lambda) &\leq \sum_i D(i) \sum_{\mathbf{y} \neq \mathbf{y}^i} \pi_i(\mathbf{y}) \left[\frac{U^i - u(\mathbf{x}^i, \mathbf{y})}{U - L} e^{\triangle\lambda L} + \frac{u(\mathbf{x}^i, \mathbf{y}) - L}{U - L} e^{\triangle\lambda U} \right] \\ &= \sum_i D(i) \left(s^i e^{\triangle\lambda L} + (1 - s^i) e^{\triangle\lambda U} \right), \text{ where} \qquad (3.35) \\ s^i &\equiv \sum_{\mathbf{y} \neq \mathbf{y}^i} \pi_i(\mathbf{y}) \frac{U^i - u(\mathbf{x}^i, \mathbf{y})}{U^i - L^i} \qquad (3.36) \end{aligned}$$

This corresponds to defining a piecewise linear upper bound on the exponential function, which might lead to a substantially tighter bound than Equation 3.26. Figure 3.1 illustrates the difference between the two bounds on an example of 4 observation-label sequence pairs. Let $w(l^i) = 1, \forall i$. The line labeled **LB** is the loose bound given by Equation 3.21 where $x_0 = -3$ and $x_1 = 4$. The gap between Z (Equation 3.25) and the loose bound (Equation 3.26) is four times the area between **LB** and the exponential function. In the example, $U^1 = 4$, $L^1 = -2$, $U^2 = 3$, $L^2 = -3$, $U^3 = 3$, $L^3 = 0$, $U^4 = 2$, $L^4 = -1$. The gap between Z and the tight bound (Equation 3.35), is the sum of the area between each line $S*$ and the exponential curve, which is much smaller than the gap of the loose bound.

Unfortunately, there is no analytical solution of $\triangle\lambda$ that minimizes Equation 3.35. By taking the second derivative w.r.t. $\triangle\lambda$, it is easy to verify that this is a convex function in $\triangle\lambda$ which can be minimized with a simple line search. Since Equation 3.26 gives a looser bound, its update rule (Equation 3.29) leads to more conservative step sizes.

It is important to see that the quantities involved in the minimization of both bounds, s and s^i, are simple expectations of sufficient statistics. These values can be computed for all features simultaneously using a single dynamic programming run per

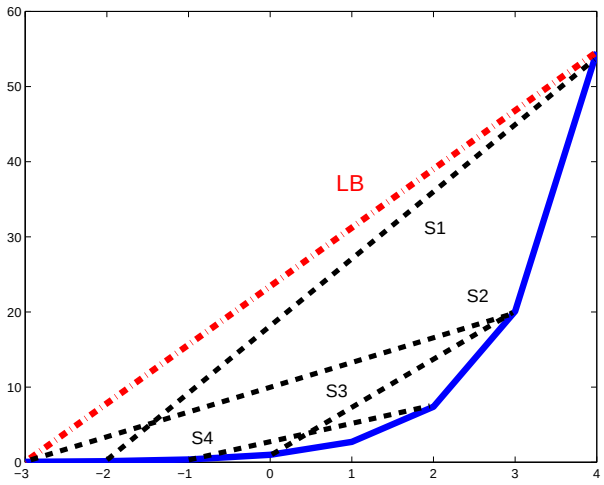

Figure 3.1: An example of the tight and loose bounds on the normalization constant Z.

sequence. Thus, at each round of boosting, the complexity of selecting the feature to include in the ensemble is $\mathbf{O}(nl)$ where nl denotes the total length of the training set and is independent of the number of features. The complexity of finding the optimal weight update is $\mathbf{O}(1)$ for the loose bound. Finding the optimal weight update of the tight bound involves one dimensional function minimization. If some minor book-keeping prior to optimization is performed (as described below in Section 3.3.2), each iteration requires only $\mathbf{O}(l_m)$ complexity where l_m denotes the maximum length of the sequences in training data. Thus, the complexity of this step is dominated by the feature selection step. In our experiments, the optimum was usually found in three or four optimization steps.

Both of the upper bounds may lead to conservative estimates of the optimal step sizes. One might use more elaborate techniques to find the optimal $\triangle\lambda_k$, once ψ_k has been selected. For example, $Z(\triangle\lambda_k)$ can be optimized exactly for the feature ψ_k that minimizes the upper bound on Z. This involves performing a dynamic programming with accumulators on the sequences in which ψ_k can be observed. The complexity of this process is $\mathbf{O}(nl)$. On average, the complexity is much less if the feature representation is sparse, as the features are active in only a few sequences.

Algorithm

We now describe Sequence AdaBoost with respect to the loose bound presented above. Sequence AdaBoost initializes the distribution over observation label sequence pairs and iteratively computes s_k and Z_k for every feature (by performing a single dynamic programming run), picks the feature ψ_k that minimizes Z, updates the feature weights Λ by adding the optimal feature weight update $\triangle\lambda_k$ and computes the new data distribution. The boosting algorithm that optimizes the tighter bound is given by replacing lines 4, 5 and 6 in Algorithm 3 accordingly. To optimize Z_k exactly, one simply changes Line 6.

Algorithm 3 Sequence AdaBoost.

1: Initialize $D_0(i, \mathbf{y}) = \frac{w(l^i)}{(|\mathcal{Y}^{l_i}|-1)\sum_j w(l^j)}$, $D_0(i, \mathbf{y}_i) = 0$.
2: Initialize $\Lambda = 0$.
3: **for** $r = 1, \ldots, R$ **do**
4: Perform a dynamic programming over the training data to compute $s_k, \forall k$
5: Select ψ_k that minimizes the upper bound in Eq.Equation 3.26.
6: Compute optimal increment $\triangle\lambda_k$ using Equation 3.29.
7: Update weight $\lambda_k \leftarrow \lambda_k + \triangle\lambda_k$.
8: Update D_{r+1} using Eq.Equation 3.15.
9: **end for**

For regularization purposes, we add a smoothing term to Equation 3.29 in order to limit the magnitude of $\triangle\lambda$:

$$\triangle\lambda = \frac{1}{U-L}\log\left(\frac{-Ls+\epsilon}{U(1-s)+\epsilon}\right). \tag{3.37}$$

where ϵ is a small positive constant.

If the feature representation is sparse (which is the case for many applications), some features may be active in only a small fraction of the training instances. One can restrict the dynamic programming only to the training instances that contain the feature that is added to the ensemble in the last round and reduce the training time substantially. Also, the pseudo-accuracy s_k of a feature ψ_k needs to be updated only if the feature selected in the previous round may co-occur with ψ_k in a sequence. In order to apply these computational reductions, we store the list of sequences that a feature might be observed, $SList_k$ for all features ψ_k and the list of features which may be observed in a sequence, $FList_i$ for all training sequences $(\mathbf{x}^i, \mathbf{y})$. These structures are extremely sparse. We keep a list of sequences SL_r for which a dynamic programming

pass is required in the current boosting round r. After each round of boosting, this list is updated to include all the sequences in which the last feature picked ψ_k may be observed, $SList_k$. We also keep a list of features FL_r for which s_k requires updating. This is especially useful if we optimize Z_k exactly, as the complexity of this minimization is large. A similar technique has been used in [Collins, 2000].

Algorithm 4 A more efficient version of Sequence AdaBoost.

1: Initialize $D_0(i, \mathbf{y}) = \frac{w(l^i)}{(|\mathcal{Y}^{l_i}|-1)\sum_j w(l^j)}$, $D_0(i, \mathbf{y}_i) = 0$.
2: Initialize $\Lambda = 0$.
3: Store $SList_k$, $\forall k$ and $FList_i$, $\forall i$.
4: Initialize $FL_1 = \Psi$, $SL_1 = \{1, \ldots, n\}$.
5: **for** $r = 1, \ldots, R$ **do**
6: Perform a dynamic programming over SL_r
7: Compute and store $s_k, \forall k \in FL_r$
8: Select $\psi_k \in \Psi$ that minimizes the upper bound in Eq.Equation 3.26.
9: Compute optimal increment $\triangle\lambda_k$ using Equation 3.29.
10: Update weight $\lambda_k \leftarrow \lambda_k + \triangle\lambda_k$.
11: Update D_{r+1} using Eq.Equation 3.15.
12: Update $SL_{r+1} = SList_k$, $FL_{r+1} = FList_i$ $\forall i \in SL$.
13: **end for**

When optimizing the tight bound, in order to perform efficient optimization for the optimal weight update $\triangle\lambda$, along with $Slist_k$, we also store the maximum and minimum values of each feature ψ_k. Once ψ_k is selected, one can go through $Slist_k$ to collect the terms in the optimization function with the same exponent, leading to an optimization step of $\mathbf{O}(l_m)$ complexity.

3.4 Analysis

As standard Boosting, Sequence AdaBoost maximizes the ensemble margin. We first present the generalization properties of Sequence AdaBoost based on margin bounds in Theorem 3 (Section 3.4.1). We then prove that Sequence AdaBoost maximizes the margin in Section 3.4.2.

3.4.1 Margin Based Generalization Bounds

Theorem 3 is proved by [Collins, 2002b] and is an extension of Theorem 6 by Schapire et al. [1998]. It is valid for any voting method of label sequences, including Sequence

AdaBoost. It is based on the assumption that feature weights $\lambda_k \in [0,1]$ and $\sum_k \lambda_k = 1$. Since each feature $\psi_k \in \mathcal{H}$ is bounded and $f(x)$ performs a max operation, scaling Λ leaves $f(x)$ unchanged. As for the non-negativity constraints, we define a new feature representation $\bar{\Psi} \in \Re^{2d}$, where d is the size of Ψ. For each feature $\psi_k \in \Psi$, we define two features in $\bar{\Psi}$, $\bar{\psi}_k = \psi_k$ and $\bar{\psi}_{d+k} = -\psi_k$. When $\bar{\lambda}$ is restricted to $\Re^+$, the class of functions $\bar{F} = \langle \bar{\Lambda}, \bar{\Psi} \rangle$ and $F = \langle \Lambda, \Psi \rangle$ are equivalent. Thus, without loss of generality, we can assume $\lambda_k \in [0,1]$ and $\sum_k \lambda_k = 1$.

Let the convex hull $\mathcal{C}$ of $\mathcal{H}$ be the set of mappings that can be generated by taking a weighted average of classifiers from $\mathcal{H}$.

$$\mathcal{C} \equiv \left\{ F : (\mathbf{x}, \mathbf{y}) \rightarrow \sum_{\psi \in \mathcal{H}} \sum_{t=1}^{l} \lambda_\psi \psi(\mathbf{x}, \mathbf{y}; t) | \lambda_\psi \geq 0; \sum_{\psi} \lambda_\psi = 1 \right\} \tag{3.38}$$

where l is the maximum length of the sequences in the data set. In order to deal with sequences of varying lengths, we assign $\Psi(\mathbf{x}^i, \mathbf{y}; t) = \mathbf{0}, \forall i, l_i < t \leq l$. Let $S = |\Sigma|$ be the size of the micro-label set and γ be the margin:

$$\gamma(F, \mathbf{x}, \mathbf{y}) = F(\mathbf{x}, \mathbf{y}) - \max_{\mathbf{y}' \neq \mathbf{y}} F(\mathbf{x}, \mathbf{y}') \tag{3.39}$$

Theorem 3 (Theorem 8 [Collins, 2002b])**.** *Let $\mathcal{D}$ be a distribution over $\mathcal{X} * \mathcal{Y}$ and let D be a sample of n examples chosen independently at random according to $\mathcal{D}$. Assume that the feature (base-classifier) space $\mathcal{H}$ is finite, and let $\delta > 0$. Then with probability at least $1 - \delta$ over the random choice of the training set D, every function $F \in \mathcal{C}$ satisfies the following bound for all $\theta > 0$:*

$$P_{\mathcal{D}}[\gamma(F, \mathbf{x}, \mathbf{y}) \leq 0] \leq P_D[\gamma(F, \mathbf{x}, \mathbf{y}) \leq \theta] + O\left(\sqrt{\frac{1}{n} \left(\frac{R^2 \log(nN)}{\theta^2} + \log \frac{1}{\delta} \right)} \right) \tag{3.40}$$

where $\|\Psi(x, \mathbf{y}) - \Psi(x, \tilde{\mathbf{y}})\| \leq R, \forall x, \forall \mathbf{y}, N = |S|^l$.

3.4.2 Margin Maximization

We now prove that Sequence Adaboost maximizes the number of training examples with large margin, by a simple extension of Theorem 5 by Schapire et al. [1998]. In Theorem 3, the margin is defined such that Λ is normalized. To achieve this, we redefine $F(\mathbf{x}, \mathbf{y}) = \sum_r \lambda_r \sum_t \psi_r(\mathbf{x}, \mathbf{y}; t) / \sum_r \lambda_r$.

Theorem 4. *If Sequence Adaboost (Algorithm 3) generates weighted training pseudo-accuracy $s_1, \ldots, s_t$, then for all θ,*

$$P_D[\gamma(F, \mathbf{x}, \mathbf{y}) \leq \theta] \leq \prod_{r=1}^{R} \frac{U_r - L_r}{U_r} \sqrt[U_r - L_r]{s_r^{U_r+\theta}(1-s_r)^{-(L_r+\theta)} \left(-\frac{L_r}{U_r}\right)^{L_r+\theta}} \tag{3.41}$$

Proof. The proof is very similar to the proof given in [Schapire et al., 1998]. We repeat it for completeness.

When $\gamma(F, \mathbf{x}, \mathbf{y}) = \min_{\mathbf{y}' \neq \mathbf{y}} F(\mathbf{x}, \mathbf{y}) - F(\mathbf{x}, \mathbf{y}') \leq \theta$,

$$\min_{\mathbf{y}' \neq \mathbf{y}} \sum_t \sum_r \lambda_r(\psi_r(\mathbf{x}, \mathbf{y}; t) - \psi_r(\mathbf{x}, \mathbf{y}'; t)) \leq \theta \sum_{r=1}^{R} \lambda_r \tag{3.42}$$

$$1 \leq \exp(\theta \sum_{r=1}^{R} \lambda_r - \min_{\mathbf{y} \neq \mathbf{y}'} \sum_t \sum_r \lambda_r(\psi_r(\mathbf{x}, \mathbf{y}; t) - \psi_r(\mathbf{x}, \mathbf{y}'; t)) \tag{3.43}$$

Therefore,

$$P_D[\gamma(F, \mathbf{x}, \mathbf{y}) \leq \theta] \leq E_D[\exp(\theta \sum_{r=1}^{R} \lambda_r - \min_{\mathbf{y}' \neq \mathbf{y}} \sum_{t,r} \lambda_r(\psi_r(\mathbf{x}, \mathbf{y}; t) - \psi_r(\mathbf{x}, \mathbf{y}'; t))] \tag{3.44}$$

$$= \exp(\theta \sum_{r=1}^{R} \lambda_r) \sum_{i=1}^{n} \exp(-\min_{\mathbf{y}' \neq \mathbf{y}^i} \sum_{t,r} \lambda_r(\psi_r(\mathbf{x}^i, \mathbf{y}^i; t) - \psi_r(\mathbf{x}^i, \mathbf{y}'; t)))$$

$$\leq \exp(\theta \sum_{r=1}^{R} \lambda_r) \sum_{i=1}^{n} \sum_{\mathbf{y}^i \neq \mathbf{y}'} \exp(F(\mathbf{x}^i, \mathbf{y}') - F(\mathbf{x}^i, \mathbf{y}^i)) \tag{3.45}$$

$$= \exp(\theta \sum_{r=1}^{R} \lambda_r) \left(\prod_{r=1}^{R} Z_r\right) \sum_{i=1}^{n} \sum_{\mathbf{y}^i \neq \mathbf{y}'} D_{R+1}(i, \mathbf{y}') \tag{3.46}$$

Using Equation 3.29, Equation 3.30, the definition of ϵ as given in the theorem and the fact that $\sum_{i=1}^{m} \sum_{\mathbf{y}' \neq \mathbf{y}^i} D_{R+1}(i, \mathbf{y}') = 1$ proves the theorem. □

By setting $U = 1$ and $L = -1$, we see that Theorem 5 in [Schapire et al., 1998] is a special case of our theorem.

3.5 Related Work

Boosting has been applied to a variety of problems, e.g. text categorization [Schapire and Singer, 2000], document routing [Iyer et al., 2000], ranking [Freund et al., 1998], as well as part-of-speech (POS) tagging [Abney et al., 1999], prepositional phrase (PP)

attachment [Abney et al., 1999], named entity recognition [Collins, 2002c, Wu et al., 2002], parsing [Collins, 2000]. Most of these studies has neglected the structure of the observation-label space (in cases where such a structure exists) with a few exceptions.

Abney et al. [1999] attempt to capture the sequence structure in the POS tagging and PP attachment by using the correct neighboring micro-labels during the learning phase of AdaBoost and use a Viterbi-style optimization during the inference phase. We call this method *Viterbi-AdaBoost* below. Their results demonstrate that this model performs significantly worse than independent classification of each observation in the sequence. This observation can be explained by the mis-match between the learning and inference phases. During learning, since the correct neighboring labels are used, the corresponding features provide noiseless information (assuming the label noise is negligible). Hence, if the inter-dependence between labels are important, the classifier might adjust the weights such that it relies heavily on the noiseless label inter-dependence features. During inference, since the neighboring labels are predicted themselves, if the classifier is not perfect, the high confidence on the neighboring labels might lead to the propagation of an error along a sequence resulting in accuracy worse than independent classification of each observation in the sequence. In our method, however, learning and inference phrases are matched, i.e. the surrounding labels are determined via a dynamic programming technique during both training and testing phases.

Collins [2000] presents a modified version of AdaBoost applied to the parsing problem. This study, which is the motivation for our work, proposes an optimization function similar to (3.6):

$$\mathcal{R}^{\exp}(\Lambda, w) \equiv \sum_{i=1}^{n} \sum_{\mathbf{y} \in \mathbf{GEN}(\mathbf{x}^i), \mathbf{y} \neq \mathbf{y}^i} e^{F(\mathbf{x}^i, \mathbf{y}; \Lambda) - F(\mathbf{x}^i, \mathbf{y}^i; \Lambda)} \tag{3.47}$$

Based on this loss function, Collins [2000, 2002c] extends boosting to the structured observation-label classification problem. If the **GEN** set is small enough to be explicitly enumerated, the problem reduces to a multiclass classification where the size of the label set is $|\mathbf{GEN}(\mathbf{x}^i)|$ and can be solved efficiently with some book-keeping as described in [Collins, 2000]. If that is not the case (which is true for most structured problems), an external classifier is used to reduce **GEN** by selecting the N-best (structured) labels.

The difference between the two objective functions is the summation over labels: The summation in Equation 3.47 ranges over a top N list generated by an external mechanism classifier, whereas in our function the sum includes all possible sequences.

As demonstrated in Section 3.3, an explicit summation in Equation 3.6 is possible because of the availability of a dynamic programming formulation to compute sums over all sequences efficiently. One of the disadvantages of Equation 3.47 is the necessity of an external mechanism to provide a list of likely labels. Also studying some generalization properties of the algorithm, such as its effect on L_1 norm margin, is problematic [Collins, 2002b]. The advantage of this approach is the computational efficiency as one does not need to perform a dynamic programming at every round of AdaBoost.

3.6 Experiments

We experimentally evaluated Sequence AdaBoost on pitch accent prediction, Named Entity Recognition (NER) and part-of-speech tagging (POS).

We ran experiments comparing the loss function of CRFs (Equation 2.17) and the exponential loss (Equation 3.6) (Exp) optimized with the BFSG method [Benson et al., 2004] and with different formulations of Sequence AdaBoost (SBoost). We also evaluated the performance of standard AdaBoost (Boost), where the dependency between the micro-labels are ignored. For this classification problem, the feature space consists of all the features except the features represent the inter-label dependencies. Another model we tested was *Viterbi-AdaBoost* (VBoost), which uses the correct neighboring labels during training and performs a Viterbi decoding during testing [Abney et al., 1999].

The constant π of the weight function $w(l)$ in the exponential loss function is adjusted using the development data. Boosting was stopped when the accuracy of the development data started decreasing. We used the tight bound (Equation 3.35) to select the features and optimized Z exactly to find the optimal weight update for the selected feature, unless stated otherwise.

Figure 3.2 summarizes the results for pitch accent prediction of window size 5, i.e. when observation features are extracted from the current word, the two previous and next words. The first observation we make is that *appropriate* sequence models (CRF, Exp, SBoost) perform significantly better than standard AdaBoost and Viterbi-Boosting. As reported by Abney et al. [1999], Viterbi-Boosting performs worse than standard AdaBoost, which ignores the sequence structure of the problem. This result can be explained by the mismatch between training and testing phases as argued in Section 3.5. The difference between CRFs and the exponential risk optimization is not

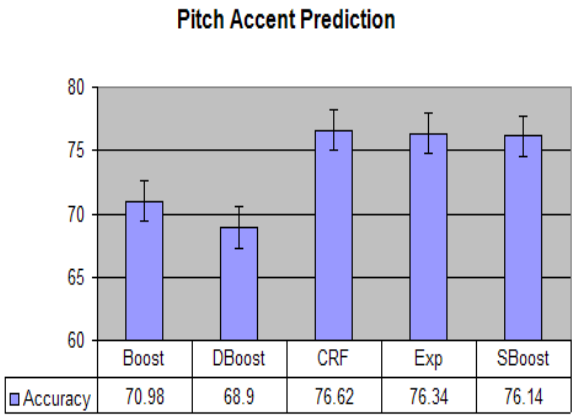

Figure 3.2: Accuracy of pitch accent prediction on Boosting formulations, $\mathcal{R}^{\exp}$ and CRFs over a window of size 5.

NER	$\mathcal{R}^{\log}$	$\mathcal{R}^{\exp}$	Boost
$S1$	59.92	59.68	48.23
$S2$	70.26	69.11	69.00
$S3$	74.83	74.22	73.88

Table 3.1: F1 measure of NER on Spanish newswire corpus on CRFs and Sequence Boosting. The window size is 3 for $S3$.

statistically significant. Sequence AdaBoost has the advantage of using only 11.56% of the parameters, whereas the other models use all of the parameters.

In NER and POS experiments, we experimented the feature sparseness properties of Sequence AdaBoost, by defining incremental sets of features. These results are summarized in Table 3.1 and in Table 3.2. The optimal weight update for POS was computed using the tight bound. The first set $S1$ includes only HMM features (See Appendix B.2 for details). $S2$ also includes spelling features of the current word and $S3$ includes features of the surrounding observations. Sequence AdaBoost performs substantially worse than the BFSG optimization of $\mathcal{R}^{\exp}$ when only HMM features are used, since there is not much information in the features other than the identity of the word to be labeled. Consequently, the boosting algorithm needs to include almost all weak learners in the ensemble and cannot exploit feature sparseness. When

POS	$\mathcal{R}^{\log}$	$\mathcal{R}^{\exp}$	Boost
$S1$	94.91	94.57	89.42
$S2$	95.68	95.25	94.91

Table 3.2: Accuracy of POS tagging on PennTreeBank.

there are more detailed features such as the spelling features, the boosting algorithm is competitive with the BFSG method, but has the advantage of generating sparser models. For example, in POS tagging, the feature *"The current word ends with -ing and the current tag is VBG."* replaces many word features. The BFSG method uses all of the available features, whereas boosting uses only about 10% of the features. Thus, the sparse nature of the application is necessary for the success of Sequence AdaBoost.

We now investigate different methods of optimizing Z in Sequence AdaBoost, namely using the loose and tight bounds (Equations 3.28 and 3.35 respectively) and optimizing Z exactly once the optimal feature is chosen with respect to the tighter bound. Even with the tighter bound in the boosting formulation, the same features are selected many times, because of the conservative estimate of the step size for parameter updates. We observe a speed up on the convergence of the boosting algorithm by optimizing Z exactly. In order to evaluate this, we present the features selected in the first 100 rounds along with the number of times each feature is selected in Table 3.3.

Table 3.3 demonstrates that the tight bound substantially improves the loose bound. For example, a feature selected 30 times in the first 100 rounds of boosting using the loose bound optimization, is selected only 5 times by the tight bound optimization. The features that are selected most frequently by the loose bound are the features that are likely to be observed at any position in the sentence, which leads to a very loose bound ($-x_0 = x_1 = l$ in Equation 3.21 where l is the maximum length of the sentences in the training set). Also, the step sizes achieved by the exact optimization is more accurate than the ones of the loose bound. The same feature is selected only once with this optimization (as opposed to 5 times with the tight bound optimization).

However, we do not observe this phenomena in pitch accent prediction. In the first 100 rounds, the highest frequency of selecting the same feature using the loose bound is 7. This can be explained by the difference of the average length of sequences in the two tasks (7 vs 36). When the length of a sequence is around 7, the loose bound is reasonably tight.

Features	L	T	TExact
y_t:I-ORG, y_{t-1}:I-ORG	30	5	1
y_t:I-PER, y_{t-1}:B-PER	30	1	1
y_t:I-MISC, first letter of x_t:2	5	5	2
y_t:I-LOC, x_{t-1} contains digit and hypen:4	3	1	1
y_t:O, x_{t+1} contains hypen and digit:4	2	2	2
y_t:I-MISC, y_{t-1}:I-MISC	1	8	2
y_t:B-PER, first letter of x_t:2,	1	7	4
y_t:B-PER, x_{t-1}'s initial capitalized and ends with dot:0	1	2	2
y_t:I-ORG, x_{t-1} is sentence initial and its initial is capitalized:2	1	2	2
y_t:I-ORG, x_{t-1}=de	1	2	1
y_t:I-MISC, x_{t+1}="	1	2	1
y_t:I-MISC, x_{t-1} ends with dot	1	2	0
y_t:I-PER, x_t ends with letter:4	0	0	2

Table 3.3: Features that are selected more than once in the first 100 rounds of boosting with the loose bound(L), the tight bound (T) or exact optimization of Z with the tight bound ($TExact$).

3.7 Discussion

We observed that the performance of Sequence AdaBoost is comparable or slightly worse than CRFs. CRFs and Sequence AdaBoost differs in terms of the loss functions (log vs. exp-loss) as well as the optimization method. Optimization of $\mathcal{R}^{\exp}$ show that some of this difference can be accounted to the loss function. There are boosting algorithms that optimize different loss functions other than the exp-loss, such as LogitBoost [Friedman et al., 2000], which optimizes the log-loss. We are going to investigate designing Sequence LogitBoost in our future work.

The advantage of the Sequence AdaBoost over CRFs is its implicit feature selection property, which results in very sparse models, using about 10% of the feature space. However, it is important that the application naturally has a sparse representation. We should also point out that an alternative method to Boosting in order to have sparse feature representations is regularizing the loss function with L_1 norm. L_1 norm, due to its hinge at 0, generally leads to sparse feature representations.

We observed that if the possible frequency of a feature does not vary much across sequences and the average length of sequences is small, the approximation given by the loose bound is tight enough to achieve adequate step sizes. Then, due to its analytical

solution, it is preferable over the other methods. Otherwise, one should use the tighter bound in order to select the features to include in the ensemble. The selection between the exact optimization versus optimizing the tight bound is a tradeoff between the number of boosting iterations and the size of the training data. When the training data is not very large, one may choose to optimize Z directly in order to reduce the boosting iterations.

Chapter 4

Hidden Markov Support Vector Machines

Support Vector Machines (SVMs), originally developed by Vapnik [1998], are a principled and powerful method rooted in the statistical learning theory and are one of the state-of-the-art classification methods. They are a discriminative learning method for inducing linear classifiers in a kernel-based feature space. The hyperplanes induced by SVMs enjoy the advantages of sparseness in the dual representation.

In this chapter, we present a generalization of SVMs to label sequence learning [Altun et al., 2003c]. This method is named Hidden Markov Support Vector Machines (HM-SVMs), since it combines the advantages of HMMs and SVMs. It is a discriminative sequence learning method with the power of maximum (soft) margin criteria and the strength of learning with non-linear feature mappings defined jointly on the observation-label space by using kernel functions, two genuine properties of SVMs. It also inherits dynamic programming techniques from HMMs to exploit the sequence structure of labels.

We employ the kernel for sequences presented in Section 2.1.2. We first overview SVMs in the standard setting briefly in Section 4.1. The following sections describe the objection function and the optimization methods (Sections 4.2 and 4.3). We present a convergence analysis of our optimization method in Section 4.4 and mention related work in Section 4.5. We conclude this chapter with experimental results in Section 4.6.

4.1 Support Vector Machines

SVMs have been studied in great depth for binary classification [Vapnik, 1998, 1999, Cortes and Vapnik, 1995]. The goal is to find a linear hyperplane that separates two classes with maximal margin γ. If a margin of $\gamma > 0$ is achieved, it can be made arbitrary large by scaling hyperplane parameters Λ due to the linearity of the hyperplane. One can overcome this multiple solution problem by either fixing the norm and maximizing the margin subject to the norm constraints or by fixing the margin to be equal to 1 (resulting in *a canonical hyperplane* [Cristianini and Shawe-Taylor, 2000]) and minimizing the squared norm of Λ, $\|\Lambda\|^2$, subject to the margin constraints.

As the data is not separable in many real-world problems, slack variables are introduced to allow for the violation of margin constraints. These variables are penalized in the objective function with either L_1 or L_2 norm. Hence, the objective function to minimize is given by:

$$\begin{aligned} R &= \frac{1}{2}\|\Lambda\|^2 + C\sum_{i=1}^{n}\xi_i \\ &\quad \text{s.t. } y^i\left[\langle\Lambda, \mathbf{x}^i\rangle + b\right] \geq 1 - \xi_i, \quad \xi_i \geq 0; \qquad \forall i \end{aligned} \tag{4.1}$$

where we use the more common L_1 norm penalty of the slack variables ξ_is. The dual of this optimization problem is a quadratic program:

$$\begin{aligned} W(\alpha) &= \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{l} y^i y^j \alpha_i \alpha_j \langle\mathbf{x}^i, \mathbf{x}^j\rangle \\ &\quad \text{s.t. } 0 \leq \alpha_i \leq C, \quad \sum_i \alpha_i y^i = 0 \quad \forall i \end{aligned} \tag{4.2}$$

The dual parameters that are larger than 0, $\alpha_i > 0$, correspond to the data points that satisfy the (soft) margin constraints with equality and the hyperplane can be stated in terms of these data points, *support vectors*:

$$F(\mathbf{x}) = \sum_i \alpha_i \langle\mathbf{x}, \mathbf{x}^i\rangle \tag{4.3}$$

where the inner product can be replaced with a kernel function to allow for classification using nonlinear feature mappings in the Reproducing Kernel Hilbert Space (RKHS).

Different formulations have been proposed to extend this framework to multiclass classification. The approach of building a classifier for each class and using some

measure such as a majority vote for inference is commonly called *one-against-all* and has been used widely in the machine learning community [Bottou et al., 1994, Schölkopf et al., 1995]. Alternatively, one can train $S(S-1)/2$ classifiers, where S is the size of the class label set, separating each class label from another. This method is called *one-against-one* [Kressel, 1999]. Weston and Watkins [1999] proposes to solve all pairwise comparisons of the correct classifier jointly (training classifiers by considering all classes at once):

$$\begin{aligned} R &= \frac{1}{2}\sum_{y=1}^{S}\|\Lambda_y\|^2 \\ &\quad \text{s.t. } \langle \Lambda_{y^i}.\mathbf{x}^i\rangle - \langle \Lambda_y.\mathbf{x}^i\rangle \geq 2, \quad \forall y \neq y^i, \forall i \end{aligned} \tag{4.4}$$

The multiclass approach we pursue here, proposed by [Crammer and Singer, 2001], is very similar to [Weston and Watkins, 1999]. Instead of discriminating the correct class for all incorrect classes, the goal is to discriminate the correct label from the most competitive incorrect label for each instance:

$$\begin{aligned} R &= \frac{1}{2}\|\mathbf{\Lambda}\|^2 \\ &\quad \text{s.t. } \langle \Lambda_{y^i}.\mathbf{x}^i\rangle - \max_{y\neq\mathbf{y}^i}\langle \Lambda_y.\mathbf{x}^i\rangle \geq 2, \quad \forall i \end{aligned} \tag{4.5}$$

The advantage of this approach over [Weston and Watkins, 1999] is the reduced number of slack variables in the soft margin formulation: Instead of having S parameters, there is only one slack variable for each training instance.

The classification rule is given by:

$$f(x) = \operatorname*{argmax}_y \langle \Lambda_y, x\rangle \tag{4.6}$$

This approach is also advantageous over one-against-all and one-against-one methods, as it does not reduce the problem into many smaller problems which leads to a substantial reduction of information available in the training data. We now consider a generalization of this formulation to label sequence learning.

4.2 Objective Function

As in standard SVM classification, we start by defining the margin. We propose the margin defined in Equation 3.39 for the analysis of Sequence AdaBoost:

$$\gamma(\mathbf{x},\mathbf{y};\Lambda) \equiv F(\mathbf{x},\mathbf{y};\Lambda) - \max_{\mathbf{y}'\neq\mathbf{y}} F(\mathbf{x},\mathbf{y}';\Lambda) \tag{4.7}$$

This is simply the generalization of the multiclass separation margin in [Crammer and Singer, 2001] to label sequences. Notice that $\gamma(\mathbf{x}^i, \mathbf{y}^i) > 0$ implies that the observation sequence $\mathbf{x}^i$ is correctly classified, since the correct label sequence $\mathbf{y}^i$ receives the highest score.

In the maximum margin setting, the goal is to find a hyperplane that not only separates the correct label sequence $\mathbf{y}^i$ from the incorrect macro-labels $\mathbf{y} \neq \mathbf{y}^i$ for each observation sequence, but also separates them with maximal margin. This can be achieved by measuring the difference of the scores of the correct macro-label and the maximally scored incorrect macro-label, $\gamma(\mathbf{x}^i, \mathbf{y}^i)$ for all training sequences and by setting Λ to maximize the minimum of the margins:

$$\Lambda^* = \operatorname*{argmax}_{\Lambda} \min_i \gamma(\mathbf{x}^i, \mathbf{y}^i; \Lambda) \tag{4.8}$$

Notice that the margin $\gamma(\mathbf{x}, \mathbf{y})$ is simply the minimum of the log-odds ratio of an observation-label sequence pair $(\mathbf{x}, \mathbf{y})$ as defined in Equation 3.9. Then, the optimization can be stated in terms of log-odds ratios to reveal the similarity between the objective functions of Sequence Boosting, Equation 3.11, and of HM-SVMs:

$$\Lambda^* = \operatorname*{argmax}_{\Lambda} \min_i \min_{\mathbf{y} \neq \mathbf{y}^i} \log O(\mathbf{x}^i, \mathbf{y}^i, \mathbf{y}; \Lambda) \tag{4.9}$$

4.2.1 Separable Case

As in standard SVMs, the margin can be made arbitrary large by scaling Λ, if a minimal margin of $\gamma > 0$ can be achieved. We overcome this multiple solution problem by fixing the margin to be equal to 1 and minimizing the squared norm of Λ, $\|\Lambda\|^2$ subject to the margin constraints:

$$\textbf{SVM}_0\textbf{:} \ \min_{\Lambda} \frac{1}{2}\|\Lambda\|^2 \tag{4.10}$$

$$\text{s.t. } F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) - F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq 1, \quad \forall i, \mathbf{y} \neq \mathbf{y}^i \tag{4.11}$$

Here each non-convex constraint has been expanded into linear inequalities, resulting in a total of $S^l - 1$ inequalities for a training sequence of length l where $S = |\Sigma|$. In the separable case, the solution Λ^* to SVM_0 achieves $\mathcal{R}^{\text{zo}}(\Lambda^*) = 0$.

As stated in Section 2.1.2 [1], the optimal solution of the above optimization is given

[1] We derive this solution for the non-separable case below.

by (2.12), namely:

$$F(\mathbf{x}^j, \mathbf{y}) = \sum_i \sum_{\bar{\mathbf{y}}} \alpha_{(i,\bar{\mathbf{y}})} K_{(i,\bar{\mathbf{y}}),(j,\mathbf{y})} = \alpha^T \mathbf{K} e_{(j,\mathbf{y})} \,. \tag{4.12}$$

4.2.2 Soft Margin Formulations

In general, the data will not be separable. One can generalize the above formulation to allow margin violations for non-separable cases as in standard SVMs. Here, we investigate three formulations. First, one may add one slack variable $\xi(i)$ for every training sequence, such that $\xi(i)$ is shared across the macro-label set of $\mathbf{x}^i$. A soft-margin SVM problem with L_1 penalty on slack variables can then be formulated as

$$\begin{aligned} \mathbf{SVM_1}\text{:} \min_{\Lambda,\xi} \frac{1}{2}\|\Lambda\|^2 + C\sum_{i=1}^{n} \xi(i), \quad \text{s.t.} \;\; \xi(i) \geq 0, \; \forall i \\ F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) - F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq 1 - \xi(i), \quad \forall i, \mathbf{y} \neq \mathbf{y}^i \end{aligned}$$

where the optimal solution of the slack variables can be found wrt the Λ parameters, $\xi(i;\Lambda) = \max\{0, 1 - \gamma(\mathbf{x}^i, \mathbf{y}^i; \Lambda)\}$.

Proposition 5. *The risk* $\mathcal{R}^{svm}(\Lambda) = \frac{1}{n}\sum_{i=1}^{n} \xi(i;\Lambda) \geq \mathcal{R}^{zo}(\Lambda)$

Proof. (i) If $\xi(i;\Lambda) < 1$, then one gets $F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) - \max_{\mathbf{y}\neq\mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}; \Lambda) = \gamma(\mathbf{x}^i, \mathbf{y}^i) > 0$ which means the data point is correctly classified and $[\![f(\mathbf{x}^i;\Lambda) \neq \mathbf{y}^i]\!] = 0 \leq \xi(i;\Lambda)$.
(ii) If $\xi(i;\Lambda) \geq 1$, then the bound holds automatically, since $[\![f(\mathbf{x}^i;\Lambda) \neq \mathbf{y}^i]\!] \leq 1 \leq \xi(i;\Lambda)$.
Summing over all i completes the proof. □

A second alternative is using the less common L_2 penalty with the same set of slack variables.

$$\begin{aligned} \mathbf{SVM_2}\text{:} \min_{\Lambda,\xi} \frac{1}{2}\|\Lambda\|^2 + C\sum_{i=1}^{n} \xi^2(i), \quad \text{s.t.} \;\; \xi(i) \geq 0, \; \forall i \\ F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) - F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq 1 - \xi(i), \quad \forall i, \mathbf{y} \neq \mathbf{y}_i \end{aligned}$$

Proposition 6. *The risk* $\mathcal{R}^{svm}(\Lambda) = \frac{1}{n}\sum_{i=1}^{n} \xi^2(i;\Lambda) \geq \mathcal{R}^{zo}(\Lambda)$

Proof. Analogous to L_1 penalty. □

As a third alternative, one can introduce one slack variable for every training instance and every sequence $\mathbf{y}$:

$$\mathbf{SVM}_3\text{:} \min_{\Lambda,\xi} \frac{1}{2}\|\Lambda\|^2 + C\sum_{i=1}^{n}\sum_{\mathbf{y}}\xi(i,\mathbf{y}), \quad \text{s.t.} \;\; \xi(i,\mathbf{y}) \geq 0, \; \forall i, \mathbf{y} \neq \mathbf{y}^i$$
$$F(\mathbf{x}^i,\mathbf{y}^i;\Lambda) - F(\mathbf{x}^i,\mathbf{y};\Lambda) \geq 1 - \xi(i,\mathbf{y}), \quad \forall i, \mathbf{y} \neq \mathbf{y}^i$$

with the optimal solution of the slack variables is given by $\xi(i,\mathbf{y};\Lambda) = \max\{0, 1 - F(\mathbf{x}^i,\mathbf{y}^i) + F(\mathbf{x}^i,\mathbf{y})\}$. This loss function provides an upper bound on the rank loss, Equation 2.14:

Proposition 7. $\frac{1}{n}\sum_{i=1}^{n}\sum_{\mathbf{y}\neq\mathbf{y}^i}\xi(i,\mathbf{y};\Lambda) \geq \mathcal{R}^{rk}(\Lambda,\mathbf{w})$.

Proof. (i) If $\xi(i,\mathbf{y};\Lambda) < 1$, then $F(\mathbf{x}^i,\mathbf{y}^i;\Lambda) > F(\mathbf{x}^i,\mathbf{y};\Lambda)$ which implies that $\mathbf{y}$ is ranked lower than $\mathbf{y}^i$, in which case $\xi(i,\mathbf{y};\Lambda) \geq 0$ establishes the bound.
(ii) If $\xi(i,\mathbf{y};\Lambda) \geq 1$, then the bound holds trivially, since the contribution of every pair $(\mathbf{x}^i,\mathbf{y})$ to $\mathcal{R}^{\text{rk}}$ can be at most 1. □

In this chapter, we focus on $\mathbf{SVM}_1$, as we expect the number of active inequalities in $\mathbf{SVM}_1$ to be much smaller compared to $\mathbf{SVM}_3$, since $\mathbf{SVM}_1$ only penalizes the *largest* margin violation for each example. We also prefer $\mathbf{SVM}_1$ over $\mathbf{SVM}_2$ due to the more common penalty used in $\mathbf{SVM}_1$.

The corresponding Lagrangian of $\mathbf{SVM}_1$ is given by:

$$\begin{aligned} L(\Lambda,\xi,\alpha,r) &= \frac{1}{2}\|\Lambda\|^2 + C\sum_{i=1}^{n}\xi(i) \\ &- \sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_{(i,\mathbf{y})}\left(F(\mathbf{x}^i,\mathbf{y}^i;\Lambda) - F(\mathbf{x}^i,\mathbf{y};\Lambda) - 1 + \xi(i)\right) - \sum_{i=1}^{n} r_i\xi(i) \end{aligned} \tag{4.13}$$

for $\alpha_{(i,\mathbf{y})} \geq 0$, $r_i \geq 0$ where we introduced Lagrange multipliers $\alpha_{(i,\mathbf{y})}$ associated with $(\mathbf{x}^i,\mathbf{y})$ pair for every margin inequality. To find the corresponding dual, we differentiate L with respect to Λ and ξ and equate to 0:

$$\nabla_\Lambda L = \Lambda - \sum_{i,\mathbf{y}\neq\mathbf{y}^i}\alpha_{(i,\mathbf{y})}(\Psi(\mathbf{x}^i,\mathbf{y}^i) - \Psi(\mathbf{x}^i,\mathbf{y})) = 0 \tag{4.14a}$$

$$\nabla_{\xi_i} L = C - \sum_{\mathbf{y}\neq\mathbf{y}^i}\alpha_{(i,\mathbf{y})} - r_i = 0 \tag{4.14b}$$

Let $\triangle\Psi(i,\mathbf{y}) = \Psi(\mathbf{x}^i,\mathbf{y}^i) - \Psi(\mathbf{x}^i,\mathbf{y})$. Substituting Equation 4.14 into the primal, we obtain the dual quadratic program (QP) of $\mathbf{SVM}_1$:

$$\max_{\alpha} \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \alpha_{(i,\mathbf{y})} - \frac{1}{2} \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \sum_{j,\bar{\mathbf{y}}} \alpha_{(i,\mathbf{y})}\alpha_{(j,\bar{\mathbf{y}}\neq\mathbf{y}^j)} \triangle\Psi(i,\mathbf{y})\triangle\Psi(j,\bar{\mathbf{y}}) \tag{4.15}$$

We define $z_{(i,\mathbf{y})} \in \{-1,1\}$ as a function indicating whether $\mathbf{y}$ is the correct macro-label for the observation sequence $\mathbf{x}^i$ or not:

$$z_{(i,\mathbf{y})} = \left\{ \begin{array}{cc} 1 & \mathbf{y} = \mathbf{y}^i \\ -1 & \text{o.w.} \end{array} \right\}$$

We also introduce the dummy Lagrange parameter

$$\alpha_{(i,\mathbf{y}^i)} = \sum_{\mathbf{y}\neq\mathbf{y}^i} \alpha_{(i,\mathbf{y})} \tag{4.16}$$

Using these definitions in order to replace the sum over $\mathbf{y} \neq \mathbf{y}^i$ with the sum over $\mathbf{y} \in \mathcal{Y}$, we can rewrite Equation 4.15 as

$$\textbf{DSVM}_1\textbf{:} \quad \max_\alpha \tfrac{1}{2}\textstyle\sum_{i,\mathbf{y}} \alpha_{(i,\mathbf{y})} - \tfrac{1}{2}\sum_{i,\mathbf{y}}\sum_{j,\bar{\mathbf{y}}} \alpha_{(i,\mathbf{y})}\alpha_{(j,\bar{\mathbf{y}})} z_{(i,\mathbf{y})} z_{(j,\bar{\mathbf{y}})} \langle \Psi(\mathbf{x}^i,\mathbf{y}), \Psi(\mathbf{x}^j,\bar{\mathbf{y}})\rangle \tag{4.17a}$$

$$\text{s.t.} \qquad \alpha_{(i,\mathbf{y})} \geq 0, \tag{4.17b}$$

$$\textstyle\sum_{\mathbf{y}} \alpha_{(i,\mathbf{y})} \leq 2C, \tag{4.17c}$$

$$\textstyle\sum_{\mathbf{y}} z_{(i,\mathbf{y})}\alpha_{(i,\mathbf{y})} = 0\,; \quad \forall i,\mathbf{y} \tag{4.17d}$$

The introduction of the z function allows us to convert the optimization to a binary classification problem where the incorrect macro-labels of each observation sequence is separated from the correct one. We call the incorrect observation-label sequence pairs *negative pseudo-examples* as they are generated only implicitly.

Since $r_i \geq 0$, Equation 4.14b enforces the upper bound on the Lagrange parameters, Equation 4.17c. The difference between $\mathbf{DSVM}_1$ and the standard binary classification problem is the coupling of Lagrange multipliers: the additional interaction among the observation-label sequence pairs for every observation sequence.

Karush-Kuhn-Tucker (KKT) conditions are given by:

$$\alpha_{(i,\mathbf{y})}\left[F(\mathbf{x}^i,\mathbf{y}^i;\Lambda) - F(\mathbf{x}^i,\mathbf{y};\Lambda) - 1 + \xi_i\right] = 0, \quad \forall i, \forall \mathbf{y} \neq \mathbf{y}^i \tag{4.18a}$$

$$\xi_i(\sum_{\mathbf{y}} \alpha_{(i,\mathbf{y})} - 2C) = 0, \quad \forall i \tag{4.18b}$$

Notice that Equation 4.18b implies that the non-zero slack variables $\xi_i > 0$ can only occur when $\sum_{\mathbf{y}} \alpha_{(i,\mathbf{y})} = 2C$. Then due to the equality constraints in Equation 4.17d,

$$\alpha_{(i,\mathbf{y}^i)} = \sum_{\mathbf{y} \neq \mathbf{y}^i} \alpha_{(i,\mathbf{y})} = C \tag{4.19}$$

when $\xi(i) > 0$. Also, Equation 4.17d generalizes the standard constraints of binary classification SVMs and implies that $\alpha_{(i,\mathbf{y})} = 0$, if $\alpha_{(i,\mathbf{y}^i)} = 0$. Hence, negative pseudo-examples can be support vectors, only if the the positive example $(\mathbf{x}^i, \mathbf{y}^i)$ is a support vector.

From Equation 4.14a and the definition of $\alpha_{(i,\mathbf{y}^i)}$, the optimal solution of $\mathbf{DSVM}_1$ is given by:

$$F(\mathbf{x}, \mathbf{y}) = \sum_i \sum_{\mathbf{y}} \alpha_{(i,\mathbf{y})} \langle \Psi(\mathbf{x}, \mathbf{y}), \Psi(\mathbf{x}^i, \mathbf{y}) \rangle \tag{4.20}$$

4.3 Optimization Method

$\mathbf{DSVM}_1$ is a quadratic program parameterized with Lagrange parameters α, whose number scales exponentially with the length of the sequences. However, we expect that only a very small fraction of these parameters (corresponding to *support sequences*) will be active at the solution because of two reasons. First, the hinge loss leads sparse solutions as in standard SVMs. Second, and more importantly, many of the parameters are closely related because of the sequence structure, i.e. large amount of overlap of the information represented by each parameter.

The interactions between these parameters are limited to parameters of the same training instances, thus the parameters of the observation sequence $\mathbf{x}^i$, $\alpha_{(i,.)}$, are independent of the parameters of other observation sequences, $\alpha_{(j,.)}$. Our optimization method exploits this dependency structure of the parameters and the anticipated sparseness of the solution to achieve computational efficiency.

4.3.1 Algorithm

We propose to use a row selection or working set procedure to incrementally add inequalities to the problem, similar to the one proposed in [Crammer and Singer, 2001]. We maintain an active set of macro labels, S^i, for every instance which are initially $\{\mathbf{y}^i\}$, the correct label sequences. We call these sets *working sets* and define the optimization

problem only in terms of the Lagrange parameters corresponding to the working set of a particular observation sequence. We incrementally update the working sets by adding Lagrange parameter(s) (corresponding to observation-label sequence pair(s)) to the optimization problem. This is done by iterating over training sequences and finding the macro-label $\mathbf{y}$ that achieves a best score with respect to the current classifier F other than the correct one. Such a sequence is found by performing a two-best Viterbi decoding [Schwarz and Chow, 1990] as described in Section 4.3.2. The satisfaction of the margin constraint by this macro-label implies that all the other macro-labels satisfy their margin constraints as well. If $\mathbf{y}$ violates the margin constraint, we add it into the working set of $\mathbf{x}^i$ and optimize the quadratic program with respect to the Lagrange parameters in the working set of $\mathbf{x}^i$, while keeping the remaining variables fixed. Thus, we add at most one negative pseudo-example to the working set at each iteration. This procedure can be viewed as a version of a cyclic coordinate ascent. The algorithm is described in Algorithm 5.

Algorithm 5 Working set optimization for HM-SVM.

Initialize $S^i \leftarrow \{\mathbf{y}^i\}$, $\alpha_{(i,.)} = \mathbf{0}$, $\forall i$
repeat
  for $i = 1, \ldots, n$ **do**
    Compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \neq \mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}; \alpha)$
    if $F(\mathbf{x}^i, \mathbf{y}^i; \alpha) - F(\mathbf{x}^i, \hat{\mathbf{y}}; \alpha) < 1 - \xi_i$ **then**
      $S^i \leftarrow S^i \cup \{\hat{\mathbf{y}}\}$
      Optimize $\mathbf{DSVM}_1$ over $\alpha_{(i,\mathbf{y})}, \forall \mathbf{y} \in S^i$
    end if
    Remove $\mathbf{y} \in S^i$ with $\alpha_{(i,\mathbf{y})} < \epsilon$
  end for
until no margin constraint is violated

4.3.2 Viterbi Decoding in HM-SVM

One can replace the inner product in the optimal solution of $\mathbf{DSVM}_1$ (Equation 4.20) with a kernel function:

$$F(\mathbf{x}, \mathbf{y}) = \sum_i \sum_{\bar{\mathbf{y}}} \alpha_{(i,\bar{\mathbf{y}})} k((\mathbf{x}^i, \bar{\mathbf{y}}), (\mathbf{x}, \mathbf{y}))\,. \tag{4.21}$$

We employ the kernel for sequences presented in Section 2.1.2 which is repeated below:

$$k((\mathbf{x},\mathbf{y}),(\bar{\mathbf{x}},\bar{\mathbf{y}})) \equiv k^1((\mathbf{x},\mathbf{y}),(\bar{\mathbf{x}},\bar{\mathbf{y}})) + k^2((\mathbf{x},\mathbf{y}),(\bar{\mathbf{x}},\bar{\mathbf{y}})) \tag{4.22a}$$

$$k^1((\mathbf{x},\mathbf{y}),(\bar{\mathbf{x}},\bar{\mathbf{y}})) = \sum_{s,t} [\![y_s = \bar{y}_t]\!] g(\Phi(\mathbf{x},s),\Phi(\bar{\mathbf{x}},t)) \,, \tag{4.22b}$$

$$k^2((\mathbf{x},\mathbf{y}),(\bar{\mathbf{x}},\bar{\mathbf{y}})) = \eta \sum_{s,t} [\![y_s = \bar{y}_t \wedge y_{s+1} = \bar{y}_{t+1}]\!]. \tag{4.22c}$$

One can exploit the kernel structure and decompose F into two contributions, $F(\mathbf{x},\mathbf{y}) = F_1(\mathbf{x},\mathbf{y}) + F_2(\mathbf{x},\mathbf{y})$, where

$$F_1(\mathbf{x},\mathbf{y}) = \sum_{s,\sigma} [\![y_s = \sigma]\!] \sum_{i,t} \beta(i,t,\sigma) g(\Psi(\mathbf{x},s),\Psi(\mathbf{x}^i,t)), \tag{4.23a}$$

$$\beta(i,t,\sigma) = \sum_{\mathbf{y}} [\![y_t = \sigma]\!] \, \alpha_{(i,\mathbf{y})} \,. \tag{4.23b}$$

and where

$$F_2(\mathbf{x},\mathbf{y}) = \eta \sum_{\sigma,\tau} \delta(\sigma,\tau) \sum_{s} [\![y_{s-1} = \sigma \wedge y_s = \tau]\!] \,, \tag{4.24a}$$

$$\delta(\sigma,\tau) = \sum_{i,\bar{\mathbf{y}}} \alpha_{(i,\bar{\mathbf{y}})} \sum_{t} [\![\bar{y}_{t-1} = \sigma \wedge \bar{y}_t = \tau]\!] \tag{4.24b}$$

Thus, we only need to keep track of the number of times a micro-label pair (σ,τ) was predicted incorrectly and the number of times a particular observation x_s^i was incorrectly classified. This representation leads to an efficient computation as it is independent of the number of incorrect sequences $\hat{\mathbf{y}}$.

In order to perform the Viterbi decoding, we have to compute the transition cost matrix and the observation cost matrix H^i for the i-th sequence. The latter is given by

$$H^i_{s\sigma} = \sum_{j} \sum_{t} \beta(j,t,\sigma) g(\Psi(\mathbf{x}^i,s),\Psi(\mathbf{x}^j,t)) \tag{4.25}$$

The coefficients of the transition matrix are simply given by the values $\delta(\sigma,\tau)$. Once the computation of the observation cost matrix and the transition cost matrix is completed, Viterbi decoding amounts to finding the values that maximizes the potential function at each position in the sequence.

4.4 Analysis

In this section, we show that the coordinate descent method in Algorithm 5 strictly improves the objective function at each iteration. Since our objective function is bounded, the algorithm converges to the minima. We first present the following two Lemmata that allow us to prove Proposition 8.

Lemma 2. *Assume $\bar{\alpha}$ is a solution of* $\mathbf{DSVM}_1$*, then $\bar{\alpha}_{(i,\mathbf{y})} = 0$ for all pairs $(\mathbf{x}^i, \mathbf{y})$ for which $F(\mathbf{x}^i, \mathbf{y}; \bar{\alpha}) < \max_{\bar{\mathbf{y}} \neq \mathbf{y}^i} F(\mathbf{x}^i, \bar{\mathbf{y}}; \bar{\alpha})$.*

Proof. Proof by contradiction. Define $\tilde{F}(\mathbf{x}^i; \alpha) = \max_{\mathbf{y} \neq \mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}^i; \alpha)$. Hence if $\bar{\alpha}$ is a solution, then

$$F(\mathbf{x}^i, \mathbf{y}^i; \bar{\alpha}) - \tilde{F}(\mathbf{x}^i; \bar{\alpha}) \geq 1 - \xi(i) \tag{4.26}$$

Assume $\bar{\alpha}_{(i,\mathbf{y})} > 0$. If $\mathbf{y}$ is a label sequence such that $F(\mathbf{x}^i, \mathbf{y}; \bar{\alpha}) < \tilde{F}(\mathbf{x}^i; \bar{\alpha})$ then

$$F(\mathbf{x}^i, \mathbf{y}^i; \bar{\alpha}) - F(\mathbf{x}^i, \mathbf{y}; \bar{\alpha}) > F(\mathbf{x}^i, \mathbf{y}^i; \bar{\alpha}) - \tilde{F}(\mathbf{x}^i; \bar{\alpha}) \geq 1 - \xi_i \tag{4.27}$$

This contradicts the KKT complementary condition Equation 4.18a, given the assumption $\bar{\alpha}_{(i,\mathbf{y})} > 0$. □

Lemma 3. *Let $\mathbf{D} = \mathbf{z}^T \mathbf{K} \mathbf{z}$ for $\mathbf{z} = (z_{(i,\mathbf{y})})_{i,\mathbf{y}}$. Then $\alpha' \mathbf{D} e_{(i,\mathbf{y})} = z_{(i,\mathbf{y})} F(\mathbf{x}^i, \mathbf{y})$, where $e_{(i,\mathbf{y})}$ refers to the canonical basis vector corresponding to the dimension of $\alpha_{(i,\mathbf{y})}$.*

Proof. $\alpha' \mathbf{D} e_{(i,\mathbf{y})} = z_{(i,\mathbf{y})} \sum_{j,\mathbf{y}'} \alpha_{(j,\mathbf{y}')} z_{(j,\mathbf{y}')} k((\mathbf{x}^i, \mathbf{y}), (\mathbf{x}^j, \mathbf{y}')) = z_{(i,\mathbf{y})} F(\mathbf{x}^i, \mathbf{y})$. □

Algorithm 5 optimizes $\mathbf{DSVM}_1^i \equiv \mathbf{DSVM}_1(\alpha_{(i,.)}; \{\alpha_{(j,.)} : j \neq i\})$ over the arguments $\alpha_{(i,.)}$ while keeping all other $\alpha_{(j,.)}$'s fixed. Adopting the proof presented in Osuna et al. [1997], we prove that Algorithm 5 strictly improves the objective function every time it expands the working set.

Proposition 8. *Assume a working set $S^i \subseteq \mathcal{Y}$ with $\mathbf{y}^i \in S^i$ is given and that a solution for the working set has been obtained, i.e. $\alpha_{(i,\mathbf{y})}, \forall \mathbf{y} \in S^i$ maximize the objective $\mathbf{DSVM}_1^i$ subject to the constraints that $\alpha_{(i,\mathbf{y})} = 0$ for all $\mathbf{y} \notin S^i$. If there exists a negative pseudo-example $(\mathbf{x}^i, \hat{\mathbf{y}})$ with $\hat{\mathbf{y}} \notin S^i$ such that the margin constraints are not satisfied, then adding $\hat{\mathbf{y}}$ to the working set $S' \equiv S^i \cup \{\hat{\mathbf{y}}\}$ and optimizing over S' subject to $\alpha_{(i,\mathbf{y})} = 0$ for $\mathbf{y} \notin S'$ yields a strict improvement of the objective function.*

Proof. Case I: If the training example $(\mathbf{x}^i, \mathbf{y}^i)$ is not a support vector, then there are no support vector of $\mathbf{x}^i$, since $\alpha_{(i,\mathbf{y}_i)} = \sum_{\mathbf{y} \neq \mathbf{y}^i} \alpha_{(i,\mathbf{y})} = 0$. Consider $\bar{\alpha}^i = \delta e_{(i,\mathbf{y}^i)} + \delta e_{(i,\hat{\mathbf{y}})}$, for some $\delta > 0$. Then, the difference in cost function can be written as:

$$\begin{aligned}
&\mathbf{DSVM}_1^i(\bar{\alpha}^i; \{\alpha_{(j,.)} : j \neq i\}) - \mathbf{DSVM}_1^i(0; \{\alpha_{(j,.)} : j \neq i\}) \\
&= (\delta e_{(i,\mathbf{y}^i)} + \delta e_{(i,\hat{\mathbf{y}})})^T \mathbf{1} - \alpha^T \mathbf{D}(\delta e_{(i,\mathbf{y}^i)} + \delta e_{(i,\hat{\mathbf{y}}^i)}) \\
&- \frac{1}{2}(\delta e_{(i,\mathbf{y}^i)} + \delta e_{(i,\hat{\mathbf{y}}^i)})^T \mathbf{D}(\delta e_{(i,\mathbf{y}^i)} + \delta e_{(i,\hat{\mathbf{y}}^i)}) \qquad (4.28) \\
&= 2\delta - \delta \left(F(\mathbf{x}^i, \mathbf{y}^i) - F(\mathbf{x}^i, \hat{\mathbf{y}}^i)\right) - \mathbf{O}(\delta^2) \geq \delta - \mathbf{O}(\delta^2) \qquad (4.29)
\end{aligned}$$

since $F(\mathbf{x}^i, \mathbf{y}^i) - F(\mathbf{x}^i, \hat{\mathbf{y}}^i) < 1$. By choosing δ small enough we can make $\delta - \mathbf{O}(\delta^2) > 0$.

Case II: If the training example is a support vector, $\alpha_{(i,\mathbf{y}_i)} > 0$, then there has to be a negative pseudo-example $\bar{\mathbf{y}}$ with $\alpha_{(i,\bar{\mathbf{y}})} > 0$. Consider $\bar{\alpha}^i = \alpha_{(i,.)} + \delta e_{(i,\hat{\mathbf{y}}^i)} - \delta e_{(i,\bar{\mathbf{y}}^i)}$.

$$\begin{aligned}
&\mathbf{DSVM}_1^i(\bar{\alpha}^i; \{\alpha_{(j,.)} : j \neq i\}) - \mathbf{DSVM}_1^i(\alpha_{(i,.)}; \{\alpha_{(j,.)} : j \neq i\}) \\
&= (\delta e_{(i,\hat{\mathbf{y}})} - \delta e_{(i,\bar{\mathbf{y}})})^T \mathbf{1} - \alpha^T \mathbf{D}(\delta e_{(i,\hat{\mathbf{y}})} - \delta e_{(i,\bar{\mathbf{y}})}) - \mathbf{O}(\delta^2) \qquad (4.30) \\
&= \delta(F(\mathbf{x}^i, \hat{\mathbf{y}}) - F(\mathbf{x}^i, \bar{\mathbf{y}})) - \mathbf{O}(\delta^2) \qquad (4.31)
\end{aligned}$$

Hence we have to show that $F(\mathbf{x}^i, \hat{\mathbf{y}}) - F(\mathbf{x}^i, \bar{\mathbf{y}}) \geq \epsilon > 0$ independent of δ. From the KKT conditions we know that $F(\mathbf{x}^i, \mathbf{y}^i) - F(\mathbf{x}^i, \bar{\mathbf{y}}) = 1$. The margin constraint of $(\mathbf{x}^i, \hat{\mathbf{y}})$ is violated if $F(\mathbf{x}^i, \mathbf{y}^i) - F(\mathbf{x}^i, \hat{\mathbf{y}}) < 1$. Setting $\epsilon = 1 + F(\mathbf{x}^i, \hat{\mathbf{y}}) - F(\mathbf{x}^i, \mathbf{y}^i)$ concludes the proof. □

4.5 Related Work

SVMs have been applied to many problems that can be modelled as a label sequence learning problem without exploiting the dependency structure of the labels. NER [Takeuchi and Collier, 2002] and protein secondary structure [Hua and Sun, 2001] are such examples. There are few exceptions to this trend such as protein secondory structure prediction study by Kudo and Matsumoto [2001] where individual classifiers are trained using on the correct neighboring labels as well as the observation sequence and the sequence structure is exploited with dynamic programming techniques during inference. This is the SVM version of *Viterbi approach* described in Section 3.5 and it may fail if label dependencies play an crucial role and the per-label accuracy is less than perfect, since it results in over-confidence on the neighboring labels.

In the last two years, there have been many studies on maximum margin formulations in structured observation-label domains. Weston et. al. [Weston et al., 2002] proposed *Kernel Density Estimation* (KDE) one of the first studies that exploits the structure of the observation and label spaces within a kernel-based framework. They extract independent features for input and output spaces. The goal is to learn a mapping from the feature space of observations, $\Psi(\mathbf{x})$ to the feature space of labels $\bar{\Psi}(\mathbf{y})$. This is achieved by finding the principal components of $\bar{\Psi}(\mathbf{y})$ using Kernel Principal Component Analysis (KPCA) and learning a mapping from $\Psi(\mathbf{x})$ to each of these components independently. The prediction is then given by solving a pre-image problem, i.e. constructing the label $\mathbf{y}$ from the feature representation $\bar{\Psi}(\mathbf{y})$.

The key idea of our approach is to extract features jointly from observation-label pairs, not independently from the observation and label spaces. This is the main difference between our approach and KDE. The compatibility of an observation sequence $\mathbf{x}$ and a label sequence $\mathbf{y}$ may depend on a particular property of $\mathbf{x}$ in conjunction with a particular property of $\mathbf{y}$, since $\mathbf{y}$ has an internal structure that can be expressed with some features. These features in turn may interact in non-trivial ways with certain properties of the observation sequence patterns. Also, the pre-image phase of KDE involves a search over the possible label set which is problematic for domains like label sequence learning where the size of the possible label set grows exponentially with the length of sequences.

We presented the primal formulation of a perceptron algorithm for label sequence learning in Section 2.3.1. In order to avoid an explicit evaluation of the feature map as well as a direct (i.e. primal) representation of the discriminant function, one can derive an equivalent dual formulation of the perceptron algorithm. The dual perceptron for structured domains (Algorithm 6) was first proposed by Collins and Duffy [2001] in NLP context. The key difference between this method and HM-SVMs is that the perceptron terminates when the correct macro-label receives the maximum score, whereas HM-SVMs continue the optimization until the maximal margin is achieved. Each iteration of the dual HM-Perceptron is more efficient than HM-SVMs' since its update rule is a simple increment rather than a QP solution as in HM-SVMs.

The generalization properties of this algorithm can be defined by its online mistake bounds. The number of mistakes made by the perceptron algorithm is at most $(R/\gamma)^2$ where $R \geq \|\Psi(\mathbf{x}^i, \mathbf{y}^i) - \Psi(\mathbf{x}^i, \mathbf{y})\|, \forall i, \forall \mathbf{y} \in \mathbf{GEN}(\mathbf{x}^i)$ and γ is the maximum achievable margin on all sequences [Collins, 2002a, Theorem 1].

Algorithm 6 Dual form of Hidden Markov Perceptron algorithm.

Initialize all $\alpha_{(i,\mathbf{y})} = 0$
repeat
  for all training patterns $\mathbf{x}^i$ **do**
    Compute $\hat{\mathbf{y}}^i = \arg\max_{\mathbf{y}\in\mathcal{Y}} F(\mathbf{x}^i, \mathbf{y})$, where
    $F(\mathbf{x}^i, \mathbf{y}) = \sum_j \sum_{\bar{\mathbf{y}}} \alpha_{(j,\bar{\mathbf{y}})} \langle \Psi(\mathbf{x}^i, \mathbf{y}), \Psi(\mathbf{x}^j, \bar{\mathbf{y}}) \rangle$
    if $\mathbf{y}^i \neq \hat{\mathbf{y}}^i$ **then**
      $\alpha_{(i,\mathbf{y}_i)} \leftarrow \alpha_{(i,\mathbf{y}_i)} + 1$
      $\alpha_{(i,\hat{\mathbf{y}}^i)} \leftarrow \alpha_{(i,\hat{\mathbf{y}}^i)} - 1$
    end if
  end for
until no more errors

Collins [2002b] presents a generalization of SVMs for structured observation label spaces, applied to parsing. The goal is to find the maximal margin where margin is defined as

$$\gamma(\mathbf{x}^i; \Lambda) = \frac{1}{\|\Lambda\|} \left(F(\mathbf{x}^i, \mathbf{y}^i) - \max_{\mathbf{y}\in\mathbf{GEN}(\mathbf{x}^i), \mathbf{y}\neq\mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}) \right) \tag{4.32}$$

and $\mathbf{GEN}(\mathbf{x}^i)$ is the N-best macro-label set generated by an external mechanism. The problem is formulated as a quadratic optimization, reducing to a large multiclass SVM. The difference between this work and our approach is the use (and need) of an external classifier and the restriction of the possible label set to $\mathbf{GEN}(\mathbf{x})$. However, each iteration of this method is more efficient than HM-SVMs as it does not require a 2-best Viterbi decoding.

Taskar et al. [2004] proposed Max-Margin Markov (M^3) Networks, published after this work. This is also a maximum margin formulation for structured observation and label domains and is very similar to HM-SVMs. The main difference is in their definition of the margin constraints:

$$F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) - \max_{\mathbf{y}\neq\mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq \triangle(\mathbf{y}^i, \mathbf{y}), \quad \forall i \tag{4.33}$$

where $\triangle(a, b)$ denotes the Hamming distance between a and b. Our margin constraints are given in Equation 4.11:

$$F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) - \max_{\mathbf{y}\neq\mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq 1, \quad \forall i$$

They propose to reparameterize the quadratic program, leading to a substantial reduction on the number of constraints, scaling only linearly with the length of sequences.

In our formulation, on the other hand, the number of constraints scale exponentially. However, the margin constraints in the exponential parameter space translate into polynomial number of constraints that are coupled due to proper marginalization. Because of this coupling, the optimization is in fact performed in the exponential space, leading to an algorithm very similar to Algorithm 5.

In a recent study, Tsochantaridis et al. [2004] generalized the framework of HM-SVMs and MMMs by defining *re-scaled slack variable* and *re-scaled margin* formulation where the margin constraints are given below (respectively):

$$F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) - \max_{\mathbf{y} \neq \mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq 1 - \frac{\xi_i}{\triangle(\mathbf{y}^i, \mathbf{y})}, \quad \forall i \tag{4.34}$$

$$F(\mathbf{x}^i, \mathbf{y}^i; \Lambda) - \max_{\mathbf{y} \neq \mathbf{y}^i} F(\mathbf{x}^i, \mathbf{y}; \Lambda) \geq \triangle(\mathbf{y}^i, \mathbf{y}) - \xi_i, \quad \forall i \tag{4.35}$$

Tsochantaridis et al. [2004] propose a generalization of Algorithm 5 and show that it terminates in polynomial number of steps with respect to the length of sequences. Thus, even though there are exponential number of constraints, the optimal solution can be obtained by evaluating only a very small percent of the total constraints, only a polynomial number of constraints.

4.6 Experiments

We first compare the performance of HM-SVMs with CRFs and dual HM-Perceptron on pitch accent prediction. Features were extracted from on window of size 5 and development data was used to adjust the regularization constant in CRFs. We used polynomial kernel of degrees 1, 2 and 3 from HM-SVMs and degree 2 for dual HM-Perceptron. C of HM-SVMs was set to 1 and no optimization was performed for this parameter. SVM$*$ denotes HM-SVMs using polynomial kernel function of degree $*$.

The results in Figure 4.1 demonstrate that HM-SVMs perform better than the other sequence learning techniques. Trained with 1^{st} order features, CRFs achieve a much better score than the equivalently informed HM-SVM (SVM1). However, using second and third order features, HM-SVM3 improves over CRFs. It is important to note that extracting second order features explicitly corresponding to features in the order of 10 millions are quite slow. The difference between the performance of HM-Perceptron2 and HM-SVM2 demonstrates the validity of the maximum margin framework for sequence labelling. It is rather surprising that the accuracy of HM-Perceptron2 is about 0.8%

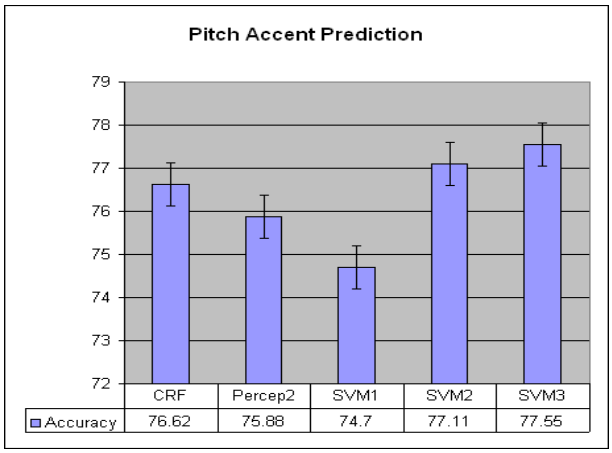

Figure 4.1: Per-label accuracy of pitch accent prediction on CRFs, HM-SVMs and dual HM-Perceptron over a window of size 5.

NER	HMM	CRF	Percep2	SVM2	SVM3
F1	69.11	72.62	72.15	72.77	73.15

Table 4.1: F1 measure of NER on 3000 Spanish newswire corpus on HMMs, CRFs, dual HM-Perceptron and HM-SVMs with window size is 3.

worse than CRFs even though HM-Perceptron2 uses 2^{nd} order features and that the primal HM-Perceptron achieves an accuracy of 76.55%, only using the 1^{st} order features. This can be explained by the averaging operation of the primal form of HM-Perceptron.

We evaluated sparseness properties of HM-SVMs. We report average values over 10 folds. There are 5716 support sequences, whereas there are $2.8166e + 14$ many margin constraints. Also, 41.73% of the training sequences satisfy the margin constraints, so the dual parameters of these examples are 0. Only 4.2435 support sequences per support examples are selected, which is extremely sparse considering the average length of the sequences is 7.

We also compared the performance of HM-SVMs with HMMs, CRFs and the dual HM-Perceptron on NER. The experimental setup was the same as pitch accent experiments, except the window size was 3. We randomly extracted 3000 sentences that are shorter than 60 and contained at least 3 name micro-labels and performed 5 fold cross-validation. Although in a generative model like an HMM, overlapping features

violate the model, our empirical evaluations showed that HMMs using the overlapping features outperformed the ordinary HMMs on NER. For this reason, we only report the results of HMMs with overlapping features.

The results summarized in Table 4.1 demonstrate the competitiveness of HM-SVMs. As expected, HMMs are outperformed by all the other methods. Again, CRFs perform better than the HM-Perceptron algorithm using polynomial kernel of degree 2. HM-SVMs achieve the best results, which validates our approach of explicitly maximizing a soft margin criterion.

Figure 4.2 illustrates the nature of the extracted support sequences [2]. The example sentence along with the correct macro-label is given on the top of the figure. **N** stands for non-name entities. The upper case letters stand for the beginning and the lower case letters stand for the continuation of the types of name entities (e.g. **M**: Miscellaneous beginning, **o**: Organization continuation). We also present a subset of the *support sequences* $\mathbf{y}$ with maximal $\alpha_{(i,\mathbf{y})}$. The first sequence is the correct macro-label. The other support sequences are depicted at the positions where they differ from the correct one. The example illustrates a regular pattern of support sequences such that most of the pseudo-negative support sequences differ from the correct label sequence only in a few positions, leading to sparse solutions. In this particular example, there are 29 support sequences, whereas the size of macro-label set for this sequence is 9^{16}.

4.7 Discussion

HM-SVMs differ from the previous sequence learning methods that we have investigated, namely CRFs and Sequence Boosting, in two main aspects: the loss function and the mechanism to construct the hyperspace. HM-SVMs optimize the hinge loss and have the ability to use kernel function to construct and learn over RKHS. On the other hand, CRFs and Sequence AdaBoost operate in the parameteric space (performing computations explicitly in the feature space) and optimize the log-loss and the hinge loss respectively. Given their power of representation (due to the kernel functions), HM-SVMs are expected to be more successful where (possibly infinitely many) higher order features are relevant for the application. However, this requires computing the kernel matrix iteratively, which may be computationally expensive when the data set is large.

[2] This example is from a smaller experiment with 300 sentences.

```
PP ESTUDIA YA PROYECTO LEY TV REGIONAL REMITIDO
O  N       N  N        M   m  m        N

POR LA JUNTA Merida ( EFE ) .
N   N  O     L      N  O  N N


ONNNMmmNNNOLNONN
--------M-------
------N---------
----------P-----
----N-----------
N---P-----------
-------m--------
-----------o----
```

Figure 4.2: Example sentence, the correct named entity labeling, and a subset of the corresponding support sequences.

The ability to use kernel functions is because of the L_2-norm optimization, which leads to an inner product in feature spaces. Thus, the norm of the margin does not only serve as a regularizer, but can also provide kernelization properties. In the next chapter, we investigate a GP formulation for sequence classification which can be considered as a method that combines the best of both worlds: HM-SVMs' RKHS property due to the L_2 norm optimization and CRFs' advantages of a probabilistic framework due to the log-loss function.

As stated earlier, the objective function of HM-SVMs and Sequence Boosting can be expressed in terms of the log-odds ratio or the margin. HM-SVMs aim to maximize the minimum margin, whereas the loss function of Sequence Boosting considers the odds ratio of all observation-label sequence pairs:

$$\Lambda^* = \arg\max_{\Lambda} \min_{i} \min_{\mathbf{y} \neq \mathbf{y}^i} O(\mathbf{x}^i, \mathbf{y}^i, \mathbf{y}; \Lambda)$$

$$\Lambda^* = \operatorname*{argmin}_{\Lambda} \sum_{i} D(i) \sum_{\mathbf{y} \neq \mathbf{y}^i} O(\mathbf{x}^i, \mathbf{y}, \mathbf{y}^i; \Lambda)$$

where $O(\mathbf{x}^i, \mathbf{y}, \bar{\mathbf{y}}) = \exp(F(\mathbf{x}, \mathbf{y}) - F(\mathbf{x}, \bar{\mathbf{y}}))$.

Boosting and SVMs are large-margin classifiers, which are known to have good generalization properties. In general, the margin of an example and a hyperplane can be defined wrt an arbitrary norm. For example, in binary classification, L_p norm margin of an observation x^i is given by

$$\gamma^p(x^i;\Lambda) = \frac{y^i\langle x^i,\Lambda\rangle}{\|\Lambda\|_p} \tag{4.36}$$

Breiman [1999] shows that Arc-GV boosting algorithm solves a linear program to maximize the L_1 margin. For margin $\gamma > 0$, this is also true for AdaBoost [Freund and Schapire, 1996, Ratsch et al., 2000] and consequently for Sequence AdaBoost. We have seen that, in SVMs, as well as HM-SVMs, a quadratic program is solved to maximize L_2 norm margin.

Mangasarian [1999, Theorem 2.2] gives an insight of the geometrical properties of margin maximization for arbitrary norm projections: Normalizing the hyperplane parameters Λ wrt an L_p norm corresponds to measuring a distance of a point x to the hyperplane with respect to L_q norm measure, where $1/p + 1/q = 1$. This means that Boosting requires a bound on the maximum absolute value (L_∞) of the data, whereas SVM requires a bound on the L_2-norm of the data. Thus, Boosting is expected to perform better when weak learners have a similar output range. In sequence labeling framework, this corresponds to problems where the length of the sequences are balanced across the data set and the variance of the expectations of features is small. This might constraint the possible application set of Sequence Boosting severely. To partially overcome this limitation, we introduce $w(l_i)$ to the optimization function of Sequence Boosting. This term scales exponentially with the length of sequences and therefore compensates for varying sequence lengths. One can also divide longer sequences into subsequences.

Sequence AdaBoost induces classifiers that have feature sparseness in the primal form, due to the greedy optimization method. HM-SVMs also return sparse classifiers, but w.r.t. the dual parameters. This sparseness is because of the hinge loss, $\max(0, 1 + F(\mathbf{x}^i, \mathbf{y}) - F(\mathbf{x}^i, \mathbf{y}^i))$, which is truncated to 0 when the corresponding margin constraint is satisfied. Thus, sparseness is inherent to the objective function of HM-SVMs, as long as some (soft) margin is achievable.

Chapter 5

Gaussian Process Sequence Classification

Gaussian Processes (GPs) are non-parametric tools making use of the *kernel trick* to work in high (possibly infinite) dimensional spaces like SVMs. As other discriminative methods, they predict single variables and traditionally do not take into account any dependency structure in case of multiple label predictions. Our goal in this chapter is to formulate the problem of label sequence learning as a Gaussian Process and develop an optimization method for this formulation, namely Gaussian Process Sequence classification (GPS). GPS can be thought of as the dual formulation of CRFs. It combines the advantages of CRFs, which we see in its rigorous probabilistic semantics and the advantages of kernel-based methods.

We first overview Gaussian Processes (GPs) in Section 5.1 and then present the objective function of GPS (Section 5.2). Exploiting the compositionality of the kernel function, we derive a 1^{st} order gradient-based optimization method for the GPS classification in Section 5.3 and provide theoretical analysis of the GPS classification in Section 5.4. Section 5.5 presents related work and we conclude this chapter with some experimental results in Section 5.6 and a comparison of GPS classification with other label sequence learning methods.

For notational convenience, we will assume that all training sequences are of the same length l in this chapter.

5.1 Gaussian Process Classification

In supervised classification, we are given a training set of n labeled instances or observations (x^i, y^i) with $y^i \in \{1, \ldots, m\}$, drawn i.i.d. from an unknown, but fixed, distribution $P(x, y)$. We denote the training observations and labels by $\mathbf{x} = (x^1, \ldots, x^n)$ and $\mathbf{y} = (y^1, \ldots, y^n)$, respectively. We introduce an intermediate, unobserved stochastic process $\mathbf{F} \equiv (F(x, y))$ [1]. Given an instantiation of the stochastic process, we assume that the conditional probability $p(y|x, \mathbf{F})$ depends only on the values of $\mathbf{F}$ at the observation x via a multinomial response model, i.e.

$$p(y|x, \mathbf{F}) = p(y|F(x, \cdot)) = \frac{\exp(F(x, y))}{\sum_{y'=1}^{m} \exp(F(x, y'))} \tag{5.1}$$

For notational convenience, we will identify $\mathbf{F}$ with the relevant restriction of $\mathbf{F}$ to the training patterns $\mathbf{x}$ and represent it as a $n \times m$ matrix. For simplicity we will (in slight abuse of notation) also think of $\mathbf{F}$ as a vector with multi-index (i, y).

In a Bayesian framework, the prediction of a label for a new observation x is obtained by computing the posterior probability distribution over labels and selecting the label that has the highest probability:

$$p(y|\mathbf{x}, \mathbf{y}, x) = \int p(y|\mathbf{F}(x, \cdot))\, p(\mathbf{F}|\mathbf{x}, \mathbf{y})\, d\mathbf{F} \tag{5.2}$$

Thus, one needs to integrate out all $n \cdot m$ latent variables of $\mathbf{F}$. Since this is in general intractable, it is common to perform a saddle-point approximation of the integral around the optimal point estimate, which is the maximum a posterior (MAP) estimate: $p(y|\mathbf{x}, \mathbf{y}, x) \approx p(y|\mathbf{F}^{\text{map}}(x, \cdot))$ where $\mathbf{F}^{\text{map}} = \text{argmax}_{\mathbf{F}} \log p(\mathbf{F}|\mathbf{x}, \mathbf{y})$. Exploiting the conditional independence assumptions, the posterior of $\mathbf{F}$ can – up to a multiplicative constant – be written as

$$p(\mathbf{F}|\mathbf{x}, \mathbf{y}) \propto p(\mathbf{F}) p(\mathbf{y}|\mathbf{F}, \mathbf{x}) = p(\mathbf{F}) \prod_{i=1}^{n} p(y^i|\mathbf{F}(x^i, \cdot)) \tag{5.3}$$

The key idea of Gaussian Processes is to define the prior $p(\mathbf{F}|\mathbf{x}, \mathbf{y})$ directly within the function space, without parameterizing $\mathbf{F}$. In particular, this distribution is assumed to be Gaussian, making $\mathbf{F}$ a Gaussian Process:

[1] The output of the function F at a data point (x, y) is a random variable. Then the set of output values of all data points $\mathbf{F} \equiv (F(x, y))$ are correlated random variables, so called a *stochastic process*.

Definition 1 ([Gibbs, 1997]). *A Gaussian Process is a collection of random variables* $\mathbf{F} = (F(x_1), F(x_2), \dots)$ *which have a joint Gaussian distribution*

$$p(\mathbf{F}|\{x_n\}, \mathbf{K}) = \frac{1}{Z} \exp\left(-\frac{1}{2}(\mathbf{F}-\mu)^T \mathbf{K}^{-1}(\mathbf{F}-\mu)\right) \tag{5.4}$$

for any collection of inputs $\{x_n\}$*, where* μ *is the mean and* $\mathbf{K}$ *is the covariance matrix of the data defined by the covariance function* C*.*

It is common to use zero mean Gaussian processes with a kernel function C as the covariance function. In multiclass classification, it is generally assumed that the processes $\mathbf{F}(\cdot, y)$ and $\mathbf{F}(\cdot, y')$ are uncorrelated for $y \neq y'$ [Williams and Barber, 1998]. We denote by $\mathbf{K}$ the kernel matrix with entries $K_{(i,y),(j,y')} = C((x^i, y), (x^j, y'))$. Notice that under the above assumptions $\mathbf{K}$ has a block diagonal structure with blocks $\mathbf{K}(y) = (K_{ij}(y))$, $K_{ij}(y) \equiv C_y(x^i, x^j)$, where C_y is a class-specific covariance function.

Combining the GP prior over $\mathbf{F}$ and the conditional model in Equation 5.1 yields the more specific expression:

$$\log p(\mathbf{F}|\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} \left[F(x^i, y^i) - \log \sum_{y=1}^{m} \exp(F(x^i, y)) \right] - \frac{1}{2}\mathbf{F}^T \mathbf{K}^{-1} \mathbf{F} + \text{const.} \tag{5.5}$$

As stated in Section 2.1.2, the Representer Theorem [Kimeldorf and Wahba, 1971] guarantees that the maximizer of Equation 5.5 is of the form:

$$\mathbf{F}^{\text{map}}(x^i, y) = \sum_{j=1}^{n} \sum_{y'=1}^{m} \alpha_{(j,y')} K_{(i,y),(j,y')} = \alpha^T \mathbf{K} e_{(i,y)} \tag{5.6}$$

with suitably chosen coefficients α where $\alpha_{(j,y)}$ is a coefficient of the training example x^i and the label y, and $e_{(i,y)}$ is the (i,y)-th unit vector. In the block diagonal case, $K_{(i,y),(j,y')} = 0$ for $y \neq y'$ and this reduces to the simpler form:

$$\mathbf{F}^{\text{map}}(x^i, y) = \sum_{j=1}^{n} \alpha_{(j,y)} C_y(x^i, x^j). \tag{5.7}$$

Using the representation in Equation 5.6, we can rewrite the optimization problem as an objective R which is the negative of Equation 5.5 parameterized by α:

$$\begin{aligned} R(\alpha|\mathbf{x}, \mathbf{y}) &= \alpha^T \mathbf{K} \alpha - \sum_{i=1}^{n} \log p(y^i|x^i, \alpha) \\ &= \alpha^T \mathbf{K} \alpha - \sum_{i=1}^{n} \alpha^T \mathbf{K} e_{(i,y^i)} + \sum_{i=1}^{n} \log \sum_{y} \exp(\alpha^T \mathbf{K} e_{(i,y)}) \end{aligned} \tag{5.8}$$

A comparison between Equation 5.8 and a similar multiclass SVM formulation Equation 4.5 [Crammer and Singer, 2001] clarifies the connection to SVMs. Their difference lies primarily in the utilized loss functions: logistic loss vs. hinge loss. Because the hinge loss truncates values smaller than ϵ to 0, it enforces sparseness in terms of the α parameters. This is not the case for logistic regression as well as other choices of loss functions.[2]

As explained in [Gibbs, 1997], Gaussian Processes have been studied since 1960's, starting with Matheron [1963]'s *kriging* which is identical to the Gaussian Process regression. In the last ten years, many researchers studied the Bayesian inference of Gaussian Process regression, e.g. [Neal, 1996, 1997, Williams and Rasmussen, 1996]. For Gaussian Process classification, on the other hand, there are well-established approximate Bayesian inference techniques [Mackay, 1992, Neal, 1996]. Other Bayesian approximation methods are Laplace approximation [Williams and Barber, 1998, Williams and Seeger, 2000], variational methods [Tommi and Jordan, 1996], mean field approximations [Opper and Winther, 2000], and expectation propagation [Minka, 2001b, Seeger et al., 2003].

Following the early literature on Gaussian Process regression, we presented a classification method for Gaussian Process using a MAP estimate, rather than a Bayesian inference method. This is because the generalization of Bayesian Gaussian Process classification leads to a computationally very expensive method, posing serious scalability problems. Then, our Gaussian Process formulation for label sequences is equivalent to the generalization of kernel logistic regression to sequences.

5.2 Objective Function

In sequence labeling problem, we simply replace the observation-label pairs (x, y) with observation-label *sequence* pairs $(\mathbf{x}, \mathbf{y})$. Then, the problem can be considered as a multiclass classification where the input is an observation sequence $\mathbf{x}$ and for an observation sequence of length l, the corresponding label set $\mathcal{Y}$ is of size $m = S^l$, where $S = |\Sigma|$ is the size of the micro-label set.

We use the sequence kernel described in Section 2.1.2 for the covariance function. Notice that the use of a block diagonal kernel matrix is not an option in the current

[2]Several studies focused on finding sparse solutions of Equation 5.8 or optimization problems similar to Equation 5.8 [Bennett et al., 2002, Girosi, 1998, Smola and Schölkopf, 2000, Zhu and Hastie, 2001].

setting, since it would prohibit generalizing across label sequences that differ in as little as a single micro-label. The kernel function Equation 2.9 indeed allows for generalization across label sequences as it simply compares the micro labels of the observation pairs in each sequence at any position due to the stationarity assumption.

Given this setup, one can naively follow the same line of argumentation as in the GP classification case of Section 5.1, evoke the Representer Theorem and ultimately arrive at the objective in Equation 5.8. Since we need it for subsequent derivations, we restate the objective here

$$R(\alpha|Z) = \alpha^T \mathbf{K}\alpha - \sum_{i=1}^{n} \alpha^T \mathbf{K} e_{(i,\mathbf{y}^i)} + \sum_{i=1}^{n} \log \sum_{\mathbf{y}\in\mathcal{Y}} \exp\left(\alpha^T \mathbf{K} e_{(i,\mathbf{y})}\right) \tag{5.9}$$

whose optimal solution is given by Equation 5.6 as restated below:

$$F(\mathbf{x}^i, \mathbf{y}; \alpha) = \alpha^T \mathbf{K} e_{(i,\mathbf{y})} = \sum_{j=1}^{n} \sum_{\bar{\mathbf{y}}\in\mathcal{Y}} \alpha_{(j,\bar{\mathbf{y}})} K_{(i,\mathbf{y}),(j,\bar{\mathbf{y}})} \tag{5.10}$$

Notice that the sum over the macro-label set, $\mathcal{Y}$, which grows exponentially in the sequence length and unfortunately, unlike hinge loss, log loss does not enforce sparseness in terms of the α parameters. Therefore, this view suffers from the large cardinality of $\mathcal{Y}$. In order to re-establish tractability of this formulation, we use a trick similar to the one deployed in [Taskar et al., 2004] and reparameterize the objective in terms of an equivalent lower dimensional set of parameters. The crucial observation is that the definition of k in Equation 2.9 is homogeneous (or stationary). Thus, the absolute positions of patterns and labels in the sequence are irrelevant. This observation can be exploited by re-arranging the sums inside the kernel function with the outer sums, i.e. the sums in the objective function.

5.2.1 Exploiting Kernel Structure

In order to carry out the reparameterization more formally, we proceed in two steps. The first step consists of finding an appropriate low-dimensional summary of α. In particular, we are looking for a parameterization that does not scale with $m = S^l$. The second step consists of re-writing the objective function in terms of these new parameters.

As we prove subsequently, the following linear map $\mathbf{\Theta}$ extracts the information in

α that is relevant for solving Equation 5.9:

$$\gamma \equiv \mathbf{\Theta}\alpha, \quad \mathbf{\Theta} \in \{0,1\}^{n \cdot l \cdot S^2 \times n \cdot m} \tag{5.11}$$

where

$$\theta_{(j,t,\sigma,\tau),(i,\mathbf{y})} \equiv \delta_{ij}[\![y_t = \sigma \wedge y_{t+1} = \tau]\!] \tag{5.12}$$

Notice that each variable $\theta_{(j,t,\sigma,\tau),(i,\mathbf{y})}$ encodes whether the input sequence is the j-th training sequence and whether the macro-label $\mathbf{y}$ contains micro-labels σ and τ at position t and $t+1$, respectively. Hence, $\gamma_{(j,t,\sigma,\tau)}$ is simply the sum of all $\alpha_{(j,\mathbf{y})}$ over macro-labels $\mathbf{y}$ that contain the $\sigma\tau$-motif at position t:

$$\gamma_{(j,t,\sigma,\tau)} = \sum_{\mathbf{y} \in \mathcal{Y}} \alpha_{(j,\mathbf{y})}[\![y_t = \sigma \wedge y_{t+1} = \tau]\!]. \tag{5.13}$$

We define two reductions derived from γ via further linear dimension reduction,

$$\gamma^{(1)} \equiv \mathbf{P}\gamma, \text{ with } P_{(i,s,\sigma),(j,t,\tau,\rho)} = \delta_{ij}\delta_{st}\delta_{\sigma\tau}\,, \tag{5.14a}$$

$$\gamma^{(2)} \equiv \mathbf{Q}\gamma, \text{ with } Q_{(i,\sigma,\zeta),(j,t,\tau,\rho)} = \delta_{ij}\delta_{\sigma\tau}\delta_{\zeta\rho}\,. \tag{5.14b}$$

Intuitively, $\gamma^{(1)}$ and $\gamma^{(2)}$ are the marginals: $\gamma^{(2)}_{(i,\sigma,\tau)}$ is the sum of all $\alpha_{(i,\mathbf{y})}$ over every position in the macro-label $\mathbf{y}$ that contains $\sigma\tau$-motif. $\gamma^{(1)}_{(i,s,\sigma)}$, on the other hand, is the sum of all $\alpha_{(i,\mathbf{y})}$ that has σ micro-label at position s in macro-label $\mathbf{y}$:

$$\gamma^{(1)}_{(i,s,\sigma)} = \sum_{\mathbf{y}} \alpha_{(i,\mathbf{y})}[\![y_s = \sigma]\!]\,, \tag{5.15a}$$

$$\gamma^{(2)}_{(i,\sigma,\tau)} = \sum_{\mathbf{y},t} \alpha_{(i,\mathbf{y})}[\![y_s = \sigma \wedge y_{s+1} = \tau]\!] \tag{5.15b}$$

We can now show how to represent the kernel matrix using the previously defined matrices $\mathbf{\Theta}$, $\mathbf{P}$, $\mathbf{Q}$ and the gram matrix $\mathbf{G}$ with $G_{(i,s),(j,t)} = g(x^i_s, x^j_t)$.

Proposition 9. *With the definitions from above:*

$$\mathbf{K} = \mathbf{\Theta}^T\mathbf{K}'\mathbf{\Theta}, \quad \mathbf{K}' \equiv \left(\mathbf{P}^T\mathbf{H}\mathbf{P} + \mathbf{Q}^T\mathbf{Q}\right)$$

where $\mathbf{H} = \mathrm{diag}(\mathbf{G}, \ldots, \mathbf{G})$.

Proof. By definition, the elements of the reparameterized kernel is given by:

$$K'_{(i,t,\sigma,\tau),(j,s,\bar{\sigma},\bar{\tau})} = [\![\sigma = \bar{\sigma} \wedge \tau = \bar{\tau}]\!] + [\![\sigma = \bar{\sigma}]\!] g(\Phi(\mathbf{x}^i, t), \Phi(\mathbf{x}^j, s)). \tag{5.16}$$

The proof follows through by elementary comparison of coefficients. □

We now have S^2 parameters for every observation $\mathbf{x}_s$ in the training data, leading to a total of nlS^2 parameters. We can rewrite the optimal solution in terms of these variables:

$$\begin{aligned} F(\mathbf{x}^i, \mathbf{y}) &= \alpha^T \mathbf{K} e_{(i,\mathbf{y})} \qquad (5.17) \\ &= \sum_{s,\sigma,\tau} [\![y_s = \sigma \wedge y_{s+1} = \tau]\!] \sum_{j,t,\bar{\mathbf{y}} \in \mathcal{Y}} [\![\bar{y}_t = \sigma \wedge \bar{y}_{t+1} = \tau]\!] \alpha_{(j,\bar{\mathbf{y}})} \\ &+ \sum_{s,\sigma} [\![y_s = \sigma]\!] \sum_{j,t,\bar{\mathbf{y}} \in \mathcal{Y}} [\![\bar{y}_t = \sigma]\!] \alpha_{(j,\bar{\mathbf{y}})} g(\Phi(\mathbf{x}^i, t), \Phi(\mathbf{x}^j, s)) \\ &= \sum_{s} \sum_{j,t,\sigma,\tau} \gamma_{(j,t,\sigma,\tau)} \Big([\![y_s = \sigma \wedge y_{s+1} = \tau]\!] + [\![y_s = \sigma]\!] k(x_s^i, x_t^j) \Big) \\ &= \sum_{s} \sum_{j,t,\sigma,\tau} \gamma_{(j,t,\sigma,\tau)} K'_{(i,s,y_s,y_{s+1}),(j,t,\sigma,\tau)} \\ &= \gamma^T \mathbf{K}' \mathbf{\Theta} e_{(i,\mathbf{y})} \qquad (5.18) \end{aligned}$$

as well as the objective function:

$$R(\gamma|Z) = \gamma^T \mathbf{K}' \gamma - \sum_{i=1}^{n} \gamma^T \mathbf{K}' \mathbf{\Theta} e_{(i,\mathbf{y}^i)} + \sum_{i=1}^{n} \log \sum_{\mathbf{y} \in \mathcal{Y}} \exp\left(\gamma^T \mathbf{K}' \mathbf{\Theta} e_{(i,\mathbf{y})}\right) \quad (5.19)$$

During the reparameterization process, we suppressed the dependency of γ parameters on α parameters. α's impose marginalization constraints on γ's (to ensure the consistency of the distribution over each label), leading to coupling of γ parameters. Optimization over this constraint space is extremely difficult. However, due to the strict convexity of the optimization function (with respect to both γ and α) and due to the Representer Theorem, the unique solution in the γ space is guaranteed to coincide with the unique solution in the α space. Therefore, we can safely ignore the constraints and perform optimization over the complete γ space.

5.3 Optimization Method

The optimization methods mentioned in Section 5.1 require the computation of the Hessian matrix. In sequence labeling, this corresponds to computing the expectations of micro-labels within different cliques (see Equation 5.31), which may not be tractable to compute exactly for large training sets. In order to minimize R with respect to γ, we propose 1^{st} order optimization methods. In Section 5.3.1, we present an exact optimization method, which we call Dense Gaussian Process Sequence Classification

(DGPS). To ensure scalability, we present two sparse methods in Sections 5.3.2 and 5.3.3. We give the formulation of how the Hessian of R can be computed in Section 5.3.4 in order to point out the complexity.

5.3.1 A Dense Algorithm

It is well-known that the derivative of the log partition function with respect to γ is simply the expectation of sufficient statistics:

$$\nabla_\gamma \left[\log \sum_{\mathbf{y} \in \mathcal{Y}} \exp \left(\gamma^T \mathbf{K}' \mathbf{\Theta} e_{(i,\mathbf{y})} \right) \right] = \mathbf{K}' \mathbf{E}_{\mathbf{y}} \left[\mathbf{\Theta} e_{(i,\mathbf{y})} \right] \tag{5.20}$$

where $\mathbf{E}_{\mathbf{y}}$ denotes an expectation with respect to the conditional distribution of the macro-label $\mathbf{y}$ given the observation sequence $\mathbf{x}^i$. Then, the gradients of R are trivially given by:

$$\nabla_\gamma R = 2\mathbf{K}'\gamma - \sum_{i=1}^{n} \mathbf{K}' \mathbf{\Theta} e_{(i,\mathbf{y}^i)} + \sum_{i=1}^{n} \mathbf{K}' \mathbf{E}_{\mathbf{y}} \left[\mathbf{\Theta} e_{(i,\mathbf{y})} \right] \tag{5.21}$$

As the first two terms involve simple matrix multiplications, the remaining challenge is to come-up with an efficient way to compute the expectations. First of all, let us more explicitly examine these quantities:

$$\mathbf{E}_{\mathbf{y}} [(\mathbf{\Theta} e_{(i,\mathbf{y})})_{(j,t,\sigma,\tau)}] = \delta_{ij} \mathbf{E}_{\mathbf{y}} [[\![y_t = \sigma \wedge y_{t+1} = \tau]\!]] \tag{5.22}$$

In order to compute the above expectations one can once again exploit the structure of the kernel and is left with the problem of computing probabilities for every neighboring micro-label pair (σ, τ) at positions $(t, t+1)$ for all training sequences $\mathbf{x}^i$. The latter can be accomplished by performing the forward-backward algorithm over the training data using the transition matrix $\mathbf{T}$ and the observation matrices $\mathbf{O}^{(i)}$, which are simply decompositions and reshapings of $\mathbf{K}'$:

$$\bar{\gamma}^{(2)} \equiv \mathbf{R} \gamma^{(2)}, \text{ with } R_{(\sigma,\zeta),(i,\tau,\rho)} = \delta_{\sigma\tau} \delta_{\zeta\rho} \tag{5.23a}$$

$$\mathbf{T} \equiv [\bar{\gamma}^{(2)}]_{S,S} \tag{5.23b}$$

$$\mathbf{O}^{(i)} = [\gamma^{(1)}]_{n \cdot l, S} \mathbf{G}_{(i,.),(.,.)} \tag{5.23c}$$

where $[x]_{m,n}$ denotes the reshaping operation of a vector x into an $m * n$ matrix, $\mathbf{A}_{I,J}$ denotes the $|I| * |J|$ sub-matrix of $\mathbf{A}$ and $(.)$ denotes the set of all possible indices.

Termwise equations for the transition and observation matrices are given as follows:

$$T_{(\sigma,\tau)} = \sum_{i,t} \gamma_{(i,t,\sigma,\tau)} \tag{5.24a}$$

$$O^{(i)}_{(t,\sigma)} = \sum_{j,s,\tau} \gamma_{(j,s,\sigma,\tau)} g(\Phi(\mathbf{x}^i, t), \Phi(\mathbf{x}^j, s)) \tag{5.24b}$$

Algorithm 7 One optimization step of Dense Gaussian Process Sequence Classification (DGPS)

Require: Training data $(\mathbf{x}^i, \mathbf{y}^i)_{i=1:n}$; Proposed parameter values γ_c

1: Initialize $\gamma_c^{(1)}, \gamma_c^{(2)}$ (Equation 5.14).
2: Compute $\mathbf{T}$ wrt $\gamma_c^{(2)}$ (Equation 5.23b).
3: **for** $i = 1, \ldots, n$ **do**
4: Compute $\mathbf{O}^{(i)}$ wrt $\gamma_c^{(1)}$ (Equation 5.23c).
5: Compute $p(\mathbf{y}^i|\mathbf{x}^i, \gamma_c)$ and $\mathbf{E}_{\mathbf{y}}[[\![y_t = \sigma \wedge y_{t+1} = \tau]\!]]$ for all t, σ, τ via forward-backward algorithm using $\mathbf{O}^{(i)}$ and $\mathbf{T}$
6: **end for**
7: Compute $\nabla_\gamma R$ (Equation 5.21).

A single optimization step of DGPS is described in Algorithm 7. Note that the only matrices computed are $O^{(i)}$ and T. The other matrices are introduced only for reparameterization and are never computed. The complexity of one optimization step is $O(t^2)$ dominated by the computation of $O^{(i)}$ for all i where $t = nlr^2$. We propose to use a quasi-Newton method for the optimization process. Then, the overall complexity is given by $O(\eta t^2)$ where $\eta < t^2$. We only need to store the γ parameters. Thus, the memory requirement is given by the size of γ, $O(t)$.

During inference, one can find the most likely label sequence for an observation sequence $\mathbf{x}$ by performing a Viterbi decoding using the transition and observation probability matrices.

5.3.2 A Sparse Algorithm of Observations

While the above method is attractive for small data sets, the computation or the storage of $\mathbf{K}'$ (thus the observation and transition matrices $O^{(i)}$ and T) poses a serious problem when the data set is large. Also, classification of a new observation involves evaluating the covariance function at nl observations, which is more than acceptable for many applications. Hence, one has to find a method for sparse solutions in terms of the γ parameters to speed up the training and prediction stages.

We propose a sparse greedy method, Sparse Gaussian Process Sequence Classification (SGPS), that is similar to the method presented by [Bennett et al., 2002]. In order to motivate this algorithm, we present the following lower bound on convex functions which is simply a tangent of the convex function.

Lemma 4 (Lower Bound on Convex Functions)**.** *Let $C : \Theta \to \Re$ be a convex function on a vector space, let $\theta_0 \in \Theta$ and denote by $g \in \partial_\theta C(\theta_0)$ a vector in the subdifferential of C at θ_0. Then*

$$\min_{\theta \in \Theta} C(\theta) + \|\theta\|^2 \geq C(\theta_0) + \|\theta_0\|^2 - \|g + \theta_0\|^2. \tag{5.25}$$

Proof. Since C is convex, it follows that for any subdifferential $g \in \partial_\theta C(\theta_0)$ we have $C(\theta) \geq C(\theta_0) + g^\top \delta\theta$. Consequently,

$$\min_{\theta \in \Theta} C(\theta) + \|\theta\|^2 \geq \min_{\delta\theta \in \Theta} C(\theta_0) + g^\top \delta\theta + \|\theta_0 + \delta\theta\|^2. \tag{5.26}$$

The minimum is obtained for $\delta\theta = -(2g + \theta_0)$, which proves the claim. □

This bound provides a valuable selection and stopping criterion for the inclusion of subspaces during the greedy optimization process. Note in particular that $g + \theta_0$ is the gradient of the optimization problem in Equation 5.25, hence we obtain a lower bound on the objective function in terms of the L_2 norm of the gradient. This means that optimization over a subspace spanned by a (set of) parameter(s) is only useful if the gradient in the corresponding direction is large enough.

SGPS starts with an empty matrix $\hat{\mathbf{K}}$. At each iteration, SGPS selects a training instance $\mathbf{x}^i$ and computes the gradients of the parameters associated with $\mathbf{x}^i$, $\gamma_{(i,.)}$, to select the d coordinates with the largest absolute value of the gradient vector of R over this subspace. Then $\hat{\mathbf{K}}$ is augmented with the columns associated with the selected parameters and SGPS performs optimization of the current problem using a Quasi-Newton method. This process is repeated until the gradients vanish (i.e. they are smaller than a threshold value η) or a maximum number p of coordinates are selected (i.e. some sparseness level is achieved). Since the bottleneck of this method is the computation of the expectations, $\mathbf{E}_\mathbf{y}[[\![y_t = \sigma \wedge y_{t+1} = \tau]\!]]$, once the expectations are computed, we can pick not only one, but d coordinates.

One has two options to compute the optimal γ at every iteration: by updating all of the γ parameters selected until now, or alternatively, by updating only the parameters selected in the last iteration. We prefer the latter because of its less expensive iterations.

This approach is in the spirit of a boosting algorithm or the cyclic coordinate descent optimization method.

We consider two alternatives for picking the training sequence at each iteration. One can select the sequence one of whose gradients has the highest magnitude or select a random sequence. The former criteria does not necessarily give the best improvement when the number of coordinates to be selected $d > 1$. Also, its complexity is significantly higher than selecting a training sequence at random.

Algorithm 8 Sparse Gaussian Process Sequence Classification (SGPS) algorithm.

Require: Training data $(\mathbf{x}^i, \mathbf{y}^i)_{i=1:n}$; Maximum number of coordinates to be selected, p, $p < nlr^2$; Threshold value η for gradients

1: $\mathbf{K} \leftarrow []$
2: **repeat**
3: Pick i:
4: (1) Pick $i = \arg\max_{i'} |\nabla_{\gamma_{(i',.)}} R|$, or
5: (2) Pick $i \in \{1, \ldots, n\}$ randomly
6: Compute $\nabla_{\gamma_{(i,.)}} R$ (Equation 5.21).
7: $s \leftarrow d$ coordinates of $\nabla_{\gamma_{(i,.)}} R$ with largest absolute value
8: $\hat{\mathbf{K}} \leftarrow [\hat{\mathbf{K}}; \mathbf{K}' e_s]$
9: Optimize R wrt s.
10: **until** $\nabla_\gamma < \eta$ or p coordinates selected.

SGPS is described in Algorithm 8. When the training sequences are selected randomly, the complexity is $O(p^2 t)$ where p is the maximum number of coordinates allowed. The complexity increases by a factor of nl, the number of training sequences, when the best training sequences is selected (Line 4). The storage complexity is of $O(pt)$, dominated by $\mathbf{K}'$.

5.3.3 A Sparse Algorithm of Observation Sequences

One can also design a similar algorithm that optimizes $R(\alpha|Z)$ in Equation 5.9 to obtain sparse solutions in terms of observation-label sequence pairs $(\mathbf{x}, \mathbf{y})$.

In order to find the steepest descent direction for this formulation, we need to compute the gradient:

$$\nabla_\alpha R = 2\mathbf{K}\alpha - \sum_{i=1}^{n} \mathbf{K} e_{(i,\mathbf{y}^i)} + \sum_{i=1}^{n} \mathbf{E}_{\bar{\mathbf{y}}}[\mathbf{K} e_{(i,\bar{\mathbf{y}})}] \,. \tag{5.27}$$

where the expectation is with respect to the conditional distribution of the label sequence $\mathbf{y}$ given the observation sequence $\mathbf{x}^i$. Not surprisingly, these expectations are the sum of the expectations with respect to the reduced parameters, in Equation 5.22:

$$\mathbf{E}_{\mathbf{y}}[(\mathbf{K}e_{(i,\mathbf{y})})_{(j,\bar{\mathbf{y}})}] = \sum_t \mathbf{K}'\mathbf{E}_{\mathbf{y}}[(\mathbf{\Theta}e_{(i,\mathbf{y})})_{(j,t,\bar{y}_t,\bar{y}_{t+1})}] \tag{5.28}$$

The Viterbi decoding as well as the expectations can be computed as described in Section 4.3.2.

5.3.4 GPSC 2^{nd} Order Optimization Methods

The MAP estimate which we performed in the previous section might lead to over-fitting the posterior $P(\mathbf{F}|Z)$, since the MAP estimate does not control the variance. This might lead to a non-high probability mass around the peak of the posterior distribution. To approximate the probability mass of the posterior, one can perform a Laplace approximation which fits a Gaussian centered at the MAP estimate and computes the volume under the Gaussian, based on the justification that under certain regularity conditions, the posterior distribution approaches to a Gaussian distribution as the number of samples grows [Williams and Barber, 1998]. Finding the maximum can be achieved by performing iteratively Newton-Ralphson updates.

It is well known that for twice differentiable convex functions $L(\theta)$ the Newton updates

$$\theta \leftarrow \theta - [\nabla^2_\theta L(\theta)]^{-1}\nabla_\theta L(\theta) \tag{5.29}$$

converges to the minimum quadratically. The first gradients can be computed readily by Equation 5.21. The Hessian is given by:

$$\nabla^2_\gamma R = 2\mathbf{K}' + \sum_{i=1}^{n} \mathbf{K}'\mathbf{Var}_{\mathbf{y}}\big[\mathbf{\Theta}e_{(i,\mathbf{y})}\big] \tag{5.30}$$

The complexity of this computation is more obvious when expressed termwise:

$$\begin{aligned}
\nabla^2_{\gamma_{(j,t,\sigma,\tau)},\gamma_{(k,r,\bar{\sigma},\bar{\tau})}} R \;&=\; \sum_i^N \sum_{s,p} \mathbf{E}_{\mathbf{y}}\left[K'_{(i,s,y_s,y_{s+1}),(j,t,\sigma,\tau)} K'_{(i,p,y_p,y_{p+1}),(k,r,\bar{\sigma},\bar{\tau})}\right] \\
&-\; \sum_i^N \sum_{s,p} \mathbf{E}_{\mathbf{y}}\left[K'_{(i,s,y_s,y_{s+1}),(j,t,\sigma,\tau)}\right] \mathbf{E}_{\mathbf{y}}\left[K'_{(i,p,y_p,y_{p+1}),(k,r,\bar{\sigma},\bar{\tau})}\right] \\
&+\; 2K'_{(j,t,\sigma,\tau),(k,r,\bar{\sigma},\bar{\tau})}
\end{aligned} \tag{5.31}$$

Computing the second and the third terms in Equation 5.31 poses no computational challenge. However, the computation of the first term involves finding the joint expectations $E_{\mathbf{y}}[y_s = \sigma \wedge y_t = \bar{\sigma}]$ and $E_{\mathbf{y}}[y_s = \sigma \wedge y_{s-1} = \tau \wedge y_t = \bar{\sigma} \wedge y_{t-1} = \bar{\tau}]$ over all possible macro-labels $\mathbf{y}$ of an observation sequence $\mathbf{x}$ for all possible s, t pairs. As this computation scales quadratically with the length of the sequence, repeated computation of the Hessian exactly is intractable in general. To alleviate this problem, as Altun et al. [2004b] propose to approximate the Hessian by ignoring the correlations which go beyond the clique boundaries, leading to approximating the Hessian by a block-diagonal matrix. In this case, Newton's method turns into a block-preconditioned gradient descent, also known as a block-Jacobi method. The price for this approximation is the slower convergence, which in the worst case is linear rather than quadratic. For coherence, we restrict our attention to 1^{st} order methods in the following sections.

5.4 Analysis

In this section, we provide approximation analysis of the sparse Gaussian Process classification. We apply the Gauss-Southwell approximation bound analysis to the SGPS algorithm that chooses the best sequence. We also point out the asymptotic convergence of the SGPS algorithm that chooses a random sequence at every iteration.

In the following analysis, we assume $\gamma_k > 0, \forall k$. Thus, the hypothesis space is the convex hull $\mathcal{C}$ of H, where H is the set defined by the column sum of $\mathbf{K'\Theta}$ matrix:

$$\mathcal{C} \equiv \left\{ F : (\mathbf{x}^i, \mathbf{y}) \rightarrow \sum_{k=1}^{j} \gamma_k u_k | \quad u_k = (\mathbf{K'\Theta} e_{(i,\mathbf{y})})_k \quad \gamma_k > 0, \forall k; \right\} \tag{5.32}$$

Optimization over $\mathcal{C}$ is obviously a special case of the GPS classification problem presented previously. However, as argued in Section 3.4, since we are interested in the macro-label that maximizes $F(\mathbf{x}, \mathbf{y})$, introducing $-u_k$ for each u_k enables us to generalize the non-negativity constraint. Therefore, we can assume the above definition of the hypothesis space without loss of generality.

The optimization method described in Algorithm 8 with the best training sequence selection criteria (Line 4) is closely related to a well-known iterative method for function minimization which is known as *coordinate descent method*, *coordinate relaxation method*, *Gauss-Seidel method*, or *SOR (Successive overrelaxation method)* [3] and can be

[3] In fact, Gauss-Seidel method is a special case of SOR method.

stated in its most general form as follows [Luo and Tseng, 2001, Ortega and Rheinboldt, 2000, Hackbusch, 1994, Murty, 1998]:

Iteration 0: Choose arbitrary $x^0 \in \Re^k$

Iteration r+1: Given an x^r, choose an $i \in \{1, \ldots, k\}$, and compute a new iterate x^{r+1} satisfying:

$$x_i^{r+1} = \arg\min_{x_i} f(x_1^r, \ldots, x_{i-1}^r, x_i, x_{i+1}^r, \ldots, x_k^r) \tag{5.33}$$

where the goal is to minimize $f(x)$. When f is convex, the sequence generated by this method is guaranteed to converge to the unique minimum irrespective of the initial vector x^0 [Murty, 1998].

A special case of this method is *Gauss-Southwell* method. Its selection criteria of x_i is given by $|x_i^{r+1} - x_i^r| \leq \beta \max_{i'} |x_{i'}^{r+1} - x_{i'}^r|$. It is proved by Luo and Tseng [2001] that for

$$f(x) = g(x) + b^T x \tag{5.34}$$

where g is twice continuously differentiable and strictly convex, thus the Hessian $\nabla^2 g(x)$ is positive definite, Gauss-Southwell method converges at least linearly:

$$f(x^{r+1}) - f(x^*) \leq \left(1 - \frac{1}{\eta}\right)(f(x^r) - f(x^*)) \tag{5.35}$$

where x^* denotes the optimal solution, and $1 < \eta < \infty$. Thus

$$f(x^r) - f(x^*) \leq \left(1 - \frac{1}{\eta}\right)^t (f(x^0) - f(x^*)) \tag{5.36}$$

and for a given precision ϵ, one needs to perform $O(\log(1/\epsilon))$ iterations. Ratsch et al. [2002] extends the proof for an approximate Gauss-Southwell method where at each iteration the coordinate that has the largest absolute value in the gradient vector is selected:

$$i = \arg \max_{i' \in \{1,\ldots,k\}} |(\nabla f(x))_{i'}| \tag{5.37}$$

and points out that

$$\eta < O\left(\frac{\rho\sigma k^4 n^2}{\beta^2}\right) \tag{5.38}$$

where ρ is the Lipschitz constant of ∇g, σ is a lower bound on the eigenvalues of $\nabla^2 g(x^r)$ for all iterates r, n is the size of the training set and β scales with the ratio of the upper and lower bounds on the Hessian.

In Equation 5.34, let $b = 0$ and $g = R$ in Equation 5.19. Since the Hessian of the negative log function is positive definite, the Hessian constraint of the theorem is satisfied due to additivity. Let d in Algorithm 8 be 1 and i be selected by Line 4. If the maximum number of coordinates, p in Algorithm 8, is selected large enough, the algorithm is equivalent to a Gauss-Southwell method and is proved to converge to the optimal solution linearly. Clearly, $p > 1$ only increases the rate of convergence. As for the bound on η, we note that ρ and n scale linearly with the length of the sequences, whereas β scales quadratically.

Unfortunately, Algorithm 8 with the best training sequence criteria may not be feasible for large data sets. Algorithm 8 with random sequence selection, on the other hand, is an instance of a Gauss-Seidel method. As discussed above, Gauss-Seidel methods are guaranteed to converge asymptotically. However, this yields significantly slower convergence of random selection than best selection algorithm.

5.5 Related Work

In a recent study [Lafferty et al., 2004], independent of our work, a formulation similar to GPSC has been presented from a different perspective: kernelized Conditional Random Fields (KCRFs). KCRFs are a generalization of CRFs by introducing a more sophisticated regularizer than the one used in CRFs: an L_2 norm regularizer associated with an RKHS. As state in Section 4.7, this form of regularizer results in kernelized methods, KCRFs in this case. As regular CRFs, KCRFs can be defined on general graph structures. The Representer Theorem assures the optimal solution of KCRFs is a linear combination of the cliques of the graph g:

$$F(.) = \sum_i \sum_{c \in \mathcal{C}(g^i)} \sum_{y_c \in \mathcal{Y}^{|c|}} \alpha_{(i,c,y_c)} K_c(\mathbf{x}^i, y_c; .) \tag{5.39}$$

where c is a clique among all the cliques of the graph g^i denoted by $\mathcal{C}(g^i)$. Lafferty et al. [2004] also extend this framework to semi-supervised learning. We believe this is a very exciting direction to pursue and still has lots of challenges to explore.

Altun et al. [2004b] approach the problem in general graph structure as well and propose a method to approximate the Hessian making the 2^{nd} order methods tractable

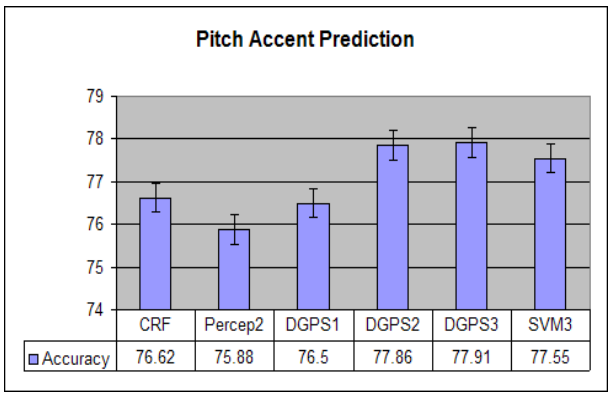

Figure 5.1: Test accuracy of Pitch Accent Prediction task over a window of size 5 on CRFs and GPS classification.

for Gaussian Process Sequence classification. This is achieved by ignoring correlations that go beyond the clique boundaries. Proofs of decomposition results of undirected graphical models and constructions for kernels are presented.

5.6 Experiments

We compared the performance of CRFs and HM-SVMs with the GPSC dense and sparse methods according to their test accuracy on pitch accent prediction. When performing experiments on DGPS, we used polynomial kernels with different degrees (denoted with DGPS$\mathbf{X}$ in Figure 5.1 where $\mathbf{X} \in \{1, 2, 3\}$ is the degree of the polynomial kernel). We used third order polynomial kernel in HM-SVMs (denoted with SVM3 in Figure 5.1). As expected, CRFs and DGPS1 performed very similar. When 2^{nd} order features were incorporated implicitly using second degree polynomial kernel (DGPS2), the performance increased dramatically. Extracting 2^{nd} order features explicitly results in a 12 million dimensional feature space, where CRFs slow down dramatically. We observed that 3^{rd} order features do not provide significant improvement over DGPS2. HM-SVM3 performs slightly worse than DGPS2.

To investigate how the sparsity of SGPS affects its performance, we report the test accuracy with respect to the sparseness of SGPS solution in Figure 5.2 using the random training sequence selection criteria. The results reported here and below are obtained using a different set of features where the performance of DGPS is 76.48%.

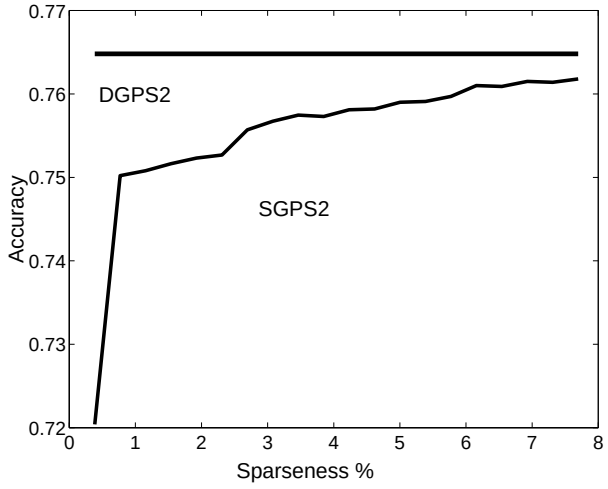

Figure 5.2: Test accuracy of Pitch Accent Prediction w.r.t. the sparseness of GPS solution.

Sparseness is measured by the percentage of the parameters selected by SGPS. The straight line is the performance of DGPS using second degree polynomial kernel. Using 1% of the parameters, SGPS achieves 75% accuracy (1.48% less than the accuracy of DGPS). When 7.8% of the parameters are selected, the accuracy is 76.18% which is not significantly different than the performance of DGPS (76.48%). We observed that these parameters were related to 6.2% of the observations along with 1.13 label pairs on average. Thus, during inference one needs to evaluate the kernel function only at 6% of the observations which reduces the inference time dramatically.

In order to experimentally verify how useful the predictive probabilities are as confidence scores, we forced DGPS to abstain from predicting a label

when the probability of a micro-label is lower than a threshold value. In Figure 5.3, we plot precision-recall values for different thresholds. We observed that the error rate for DGPS decreased 8.54%, abstaining on 14.93% of the test data. The improvement on the error rate shows the validity of the probabilities generated by DGPS.

Because our current implementation of GPSC is in Matlab, the size of the data sets are limited in our experiments. We used the Spanish newswire corpus to randomly select 1000 sentences (21K words). We used the word and its spelling properties of the current, previous and next observations.

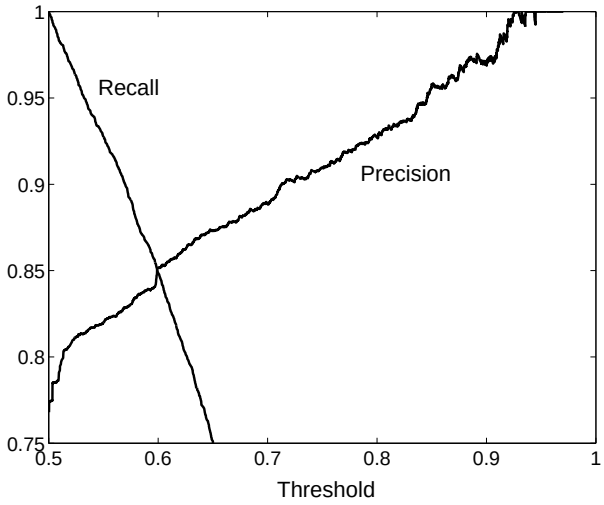

Figure 5.3: Precision-Recall curves for different threshold probabilities to abstain on Pitch Accent Prediction

	DGPS1	DGPS2	SGPS2	CRF	CRF-B
Error	4.58	4.39	4.48	4.92	4.56

Table 5.1: Test error of NER over a window of size 3 using 5-fold cross validation.

The experimental setup was similar to pitch accent prediction task. We compared the performance of CRFs with and without the regularizer term (CRF-R, CRF) with the GPSC dense and sparse methods. Qualitatively, the behavior of the different optimization methods is comparable to the pitch accent prediction task. The results are summarized in Table 5.1. Second degree polynomial DGPS outperformed the other methods. We set the sparseness parameter of SGPS to 25%, i.e. $p = 0.25nlr^2$, where $r = 9$ and $nl = 21$K on average. SGPS with 25% sparseness achieves an accuracy that is only 0.1% below DGPS. We observed that 19% of the observations are selected along with 1.32 label pairs on average, which means that one needs to compute only one fifth of the gram matrix.

We also tried a sparse algorithm that does not exploit the kernel structure and optimizes Equation 5.9 to obtain sparse solutions in terms of observation sequences $\mathbf{x}$ and macro-label $\mathbf{y}$, as opposed to SPGS, where the sparse solution is in terms of

observations and label pairs. This method achieved 92.7% of accuracy, hence, was clearly outperformed by all the other methods. The failure of the method can be explained by the fact that the algorithm is searching for a single parameter $\alpha_{i,\mathbf{y}}$ for a sequence that consists of many position that might have different behavior and needed to be explained separately (with different parameter assignments). This is, in fact, the case in the other sparse method.

The nature of the extracted support sequences chosen by the GPS classification are very different than the ones chosen by HM-SVMs. In HM-SMVs, most of the support sequences only differ in a few positions from the correct label sequence, resulting in sparse solutions, whereas in GPS classification the support sequences are different from the correct label sequence in many positions.

5.7 Discussion

We now briefly point out the relationship between GPS classification and other discriminative methods of sequence learning, in particular, CRFs, dual HM-Perceptron, HM-SVMs and MMMs.

We mentioned previously that CRF is a natural generalization of logistic regression to label sequence learning whose objective function is the minimization of the negative conditional likelihood of training data. To avoid overfitting, we multiply the conditional likelihood by a Gaussian with zero mean and diagonal covariance matrix $\mathbf{K}$, resulting in an additive term in log scale.

$$\log p(\theta|\mathbf{x},\mathbf{y}) = -\sum_{i=1}^{n} \log p(\mathbf{y}_i|\mathbf{x}_i,\theta) + \theta^T\mathbf{K}\theta \tag{5.40}$$

From a Bayesian point of view, CRFs assume a uniform prior $p(\mathbf{F})$, if there is no regularization term. When regularized, CRFs define a Gaussian distribution over a finite vector space θ. In GPSC, on the other hand, the prior is defined as a Gaussian distribution over the function space of possibly infinite dimension. Thus, GPSC generalizes CRFs by defining a more sophisticated prior on the discriminative function F. This prior leads to the ability of using kernel function in order to construct and learn over Reproducing Kernel Hilbert Spaces. So, GPSC can overcome the limitations of CRFs by generalizing a parametric statistical model. When the kernel that defines the covariance matrix $\mathbf{K}$ in GPSC is linear, F in both models become equivalent.

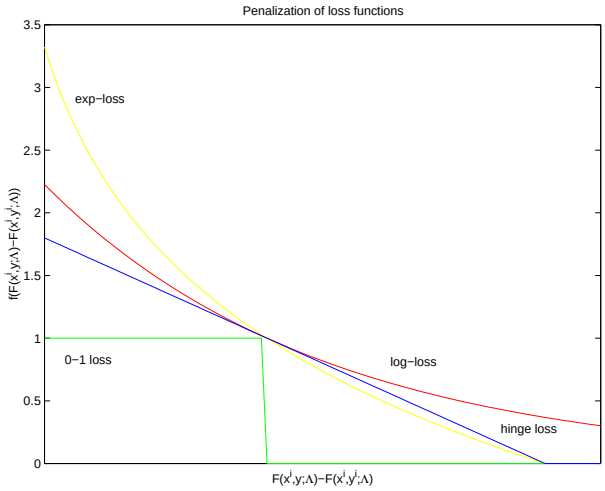

Figure 5.4: Exp-loss, log-loss and hinge-loss as upper bounds on zero-one loss.

As the primal form of HM-Perceptron can be viewed as an approximation of CRFs, dual HM-Perceptron can be interpreted as an approximation of dense GPSC, where instead of using the exact derivatives, the approximation is used due to an online update (only one training instance is considered) and the Viterbi assumption (only the most likely macro label is considered).

The difference between SVM and GP formulations to sequence learning is the utilized loss function over the training data, i.e. hinge loss vs. log loss. GPSC objective function parameterized with α, Equation 5.9, corresponds to HM-SVMs where the number of parameters scale exponentially with the length of sequences. The objective function parameterized with γ, Equation 5.19, corresponds to MMMs, where the number of parameters scale only linearly. GPSC has the advantage of providing a probabilistic framework, which is advantageous for cascaded architecture of classifiers and the incorporation of prior knowledge.

Kernel logistic regression is commonly interpreted as a large margin method [Bartlett et al., 2003]. This interpretation can be readily extended to GPSC and unifies the techniques presented in this study as *large margin methods for label sequence learning* optimizing different loss functions: exp loss of Sequence Boosting, hinge loss of HM-SVMs and log-loss of GPS classification. These functions are upper bounds on the zero-one loss. Figure 5.4 shows these loss functions on binary classification.

It is well known that if the model is correct, logistic regression, performing the maximum likelihood estimate, may lead to Bayes optimal classification error under some regularity conditions [Zhang, 2004]. If the model is not correct, then it computes the estimate that is closest to the true distribution with respect to the relative entropy. If the intrinsic loss function (optimized by the Bayes rule) is the zero-one loss, GPSC is consistent, i.e. it achieves the the optimal Bayes error rate in the large sample limit. Sequence AdaBoost also enjoys this property, since the multi-class exponential loss function, Equation 3.5, is proved to be consistent [Zhang, 2004]. Unfortunately, HM-SVM is not infinite sample consistent, since

$$\mathcal{L}(F(x,y) - \max_{\bar{y} \neq y} F(x,\bar{y})) \tag{5.41}$$

is shown to be inconsistent in general for convex loss functions $\mathcal{L}$, thus for hinge loss as well [Zhang, 2004]. We should also note that even though the asymptotic theoretical analysis provides useful statistical insights, the performance of a learning method on small-sample problems do not necessarily coincide with its performance on infinite sample. In fact, our experimental results show that HM-SVM which are inconsistent achieve better results than Sequence Boosting on all the applications.

Table 5.2 summarizes important properties of the three discriminative learning methods proposed in this study as well as CRFs. All of these methods optimize an upper bound on the 0/1 loss function and enjoy the properties of discriminative learning methods, such as efficient and easy incorporation of overlapping features into the model. They perform regularization by either adding a penalty term (CRFs), maximizing the margin (HM-SVMs) and the ensemble margin (Sequence Boost), or by enforcing a Gaussian prior over the function space (GPSC). Sequence Boosting provides sparse feature representations in the primal form, whereas HM-SVMs provide sparse representation in the dual form in terms of support vectors. CRFs and GPSC have the advantage of probabilistic interpretation. HM-SVMs and GPSC overcome the limitations of parametric models by using kernels. The basic computation block of HM-SVM is the Viterbi decoding, whereas in the other models Forward-Backward algorithm is used to compute the expectations.

	CRF	SBoost	HM-SVM	GPSC
Loss function	Log	Exp	Hinge	Log
Overlapping features	Yes	Yes	Yes	Yes
Regularization	Penalty	Ensemble margin	Margin	Gaussian prior
Feature sparseness	No	Yes	No	No
Probabilistic Semantics	Yes	No	No	Yes
Implicit features(kernels)	No	No	Yes	Yes
Computation	Forw-Backw	Forw-Backw	Viterbi	Forw-Backw

Table 5.2: Properties of discriminative label sequence learning methods.

Chapter 6

Conclusions and Future Work

In many real world problems, the goal is to predict best values for multiple interdependent response variables. The guiding line of this thesis was to generalize the most competitive discriminative learning techniques to such scenarios. We focused on structures where response variables form a sequence and we restricted ourselves to a supervised classification framework. We presented three novel discriminative learning techniques for the label sequence learning problem, namely a generalization of boosting, Sequence Boosting, a generalization of Support Vector Machines, Hidden Markov Support Vector Machines, and a Gaussian Process formulation for label sequence learning, Gaussian Process Sequence Classification. The key idea of these methods is to learn a (kernelized) linear discriminative function of a feature representation defined over the joint observation-label space. These methods optimize various loss functions that are upper bounds on the zero-one loss which is NP complete to optimize. The discriminative nature of these methods provide advantages over generative methods, such as HMMs.

Sequence Boosting, a generalization of Boosting to label sequence learning, is an ensemble method optimizing an exponential loss function of sequences. It implicitly minimizes the ensemble margin defined over sequences. Taking advantage of the convexity of the exponential function, we defined an efficient algorithm that chooses the best weak learner at each iteration by using the Forward-Backward algorithm. Its performance in accuracy is competitive with the state-of-the-art sequence model of recent years, CRFs. As in standard AdaBoost, Sequence Boosting induces sparse solutions and therefore is preferable over CRFs or other sequence methods, in case where

efficiency during inference is crucial.

We presented a generalization of SVMs to label sequence learning problem, HM-SVMs. They inherit the the maximum-margin principle and the kernel-centric approach of SVMs. The training problem can be represented as a quadratic program and is exponential in terms of the length of the sequence. We presented an algorithm that makes use of the sparseness properties of the hinge loss and the structure of the parameters. This algorithm is proved to convergence in polynomial time in terms of the length of the sequences. Experimental results show that the algorithm outperforms CRFs in terms of accuracy and is computationally feasible.

We also investigated a Gaussian Process formulation for label sequence learning, GPSC. This method combines the advantages of CRFs and HM-SVMs, leading to a kernel based sequence method with a probabilistic interpretation. The original training problem which is exponential in the length of the sequence is re-parameterized so that it scales only polynomially with the length of the sequence. We used sparse greedy approximation methods in order to apply our approach to large scale problems. GPSC achieved the best accuracy among the methods we have investigated.

In this thesis, we focused on a sequence structure of inter-dependent variables. However, the methods presented here are exactly applicable to dependency structure for which there is an efficient dynamic programming algorithm to compute the best labeling of the observations or to compute the expectations with respect to some parameter settings. For example, results on parsing using HM-SVM algorithm have been published in [Tsochantaridis et al., 2004].

Using this line of research as a foundation, one may investigate a variety of problems on discriminative methods for graphical models.

The first natural direction is extending the proposed methods to more general graphs, for which computation of the best labeling or the sufficient statistics exactly is not tractable. One possible approach is to reformulate the optimization problem as an iterative search and update method using the ideas in [Collins and Roark, 2004] to approximate the intractable computations.

A second exciting direction is pursuing a semi-supervised or unsupervised approach to discriminative graphical models as proposed by Lafferty et al. [2004]. There is an extensive literature on semi-supervised learning. The ideas from the manifold techniques, where the assumption is that similar observations should have the same label, can be

extended to inter-dependent multiple variables. There are different settings of semi-supervised learning in structured problems. There might be partial or no labeling of some subset of the training data. It might be also be the case that some of the response variables are never observed. An interesting instance of such a problem is a sequence model where there is not a one-to-one mapping between the observations and labels. The correspondence between the observations and the response variables, commonly called *alignments*, may never be observed. One can treat the alignments and the labels as a joint structured response variable. Machine translation and pronunciation system are possible applications of such problems.

The last, but not the least direction is the investigation of the kernel design for applications with different characteristics. In kernel based methods, such as HM-SVM and GPSC, the design of the kernel plays a crucial role for reliable inference. Hence, the kernels should be tailored with respect to the characteristics of the problem. For example, a kernel designed for NLP applications, where the segments within a sequence are relatively short, is not appropriate for protein sequence applications, where a sequence usually consists of long segments of the same label.

The research on discriminative learning techniques for structured problems is still in its early stages and is a promising and exciting field of machine learning.

Appendix A

Notation

n	Number of training instances
$\mathbf{x}^i$	i^{th} observation sequence in the training set
x_t	Observation at the t^{th} position in the sequence $\mathbf{x}$
$\mathbf{y}$	Label sequence, macro-label
y_t	Label at the t^{th} position, micro-label
Σ	Micro-label set
$\mathcal{Y}$	Macro-label set
S	Size of the micro-label set, $\|\Sigma\|$
l^i	Length of $\mathbf{x}^i$
Ψ	Feature representation (defined jointly on $\mathbf{x}$ and $\mathbf{y}$)
Φ	Attributes (defined on $\mathbf{x}$)
λ_k	Weight of feature ψ_k

Appendix B

Applications

B.1 Pitch Accent Prediction

The detection of prosodic characteristics is an important aspect of both speech synthesis and speech recognition. Pitch Accent Prediction is the task of identifying more prominent words in a sentence. The goal, then, is to label each word in a sentence with 'A', accented, or 'N', not accented ($|\Sigma| = 2$). Correct placement of pitch accents aids in more natural sounding speech, while automatic detection of accents can contribute to better word-level recognition and better textual understanding.

Pitch accents are never absolute; they are relative to individual speakers, gender, dialect, discourse context, local context, phonological environment, and many other factors. Two of these factors are rhythm and timing. A word that might typically be accented may be unaccented because the surrounding words also bear pitch accent. Intonational phrase boundaries also affect pitch accent: the first word of intonational phrases (IP) is less likely to be accented while the last word of an IP tends be accented. In short, accented words within the same IP are not independent of each other. Because of this dependency, we model this problem as a label sequence learning problem.

Previous work on pitch accent prediction, however, neglected the dependency between the labels. Different machine learning techniques, such as decision trees [Hirschberg, 1993], rule induction systems [Pan and Keown, 1999], bagging [Sun, 2002], boosting [Sun, 2002] have been used in a scenario where the accent of each word is predicted independently. One exception to this line of research is the use of Hidden Markov Models (HMM) for pitch accent prediction [Pan and Keown, 1999, Conkie et al., 1999]

Variable	Definition	Example
Unigram	$\log p(w_i)$	and, I
Bigram	$\log p(w_i \mid w_{i-1})$	roughing it
Rev Bigram	$\log p(w_i \mid w_{i+1})$	rid of, wound up
Joint	$\log p(w_{i-1}, w_i)$	and I, kind of
Rev Joint	$\log p(w_i, w_{i+1})$	and I, kind of

Table B.1: Definition of probabilistic variables.

where only acoustic cues or part of speech tags combined with word frequency are used as features.

As the methods proposed in this study are discriminative, one can use many different incorporate that influence pitch accent prediction in to the model. In particular, we use probabilistic, syntactic, and phonological features in a sequence labelling setting.

The only syntactic category we used was a four-way classification for part of speech: Function, Noun, Verb, Other, where Other includes all adjectives and adverbs.

The probabilistic variables we used were the unigram frequency, the predictability of a word given the preceding and following words, and the joint probability of a word with the preceding and following word. Table B.1 provides the definition for these, as well as high probability examples from the corpus.

The phonological variables ranged from information regarding the number of syllables and phones of a word to information involving rhythm and timing, including normalized word durations, speech rate, IP length and where in the IP the target word falls. We have also included variables such as surrounding pauses and filled pauses. The variable set includes:

- Log of duration in milliseconds normalized by number of canonical phones
- Number of canonical Syllables
- Number of canonical and transcribed Phones
- Log Speech Rate; calculated on strings of speech bounded on either side by pauses of 300 ms or greater
- The length of the IP
- Preceding or following IP boundary

- Preceding or following pause
- Preceding or following filled pause (uh, um)

Gregory and Altun [2004] gives a detailed explanation of most of the features extracted from the data.

The data for this study were taken from the Switchboard Corpus [Godfrey et al., 1992] which consists of 2430 telephone conversations between adult speakers (approximately 2.4 million words). Participants were both male and female and represented all major dialects of American English. We used a portion of this corpus that was phonetically hand-transcribed [Greenberg et al., 1996] and segmented into speech boundaries at turn boundaries or pauses of more than 500 ms on both sides. This corpus consists of 1824 fragments of seven words length on average. Additionally, each word was coded for probabilistic and contextual information, such as word frequency, conditional probabilities, the rate of speech, and the canonical pronunciation [Fosler-Lussier and Morgan, 1998].

We extracted observation-label features from a window of size $\{1, 3, 5\}$ centered at the word to be labeled and extracted 1^{st} order Markov features to capture the dependencies between neighboring labels. Since our current implementation of CRF and boosting only accepts categorical variables, all probabilistic variables were binned into 25 equal categories.

The experiments were run with 10-fold cross validation. In order to adjust some free variables, we extracted 1/10'th of the training data to use as development data. We computed the baseline by simply assigning the most common label, unaccented. The per-label accuracy was 60.53%. Previous research has demonstrated that part of speech and frequency, or a combination of these two, are very reliable predictors of pitch accent. HMMs using these features achieved an accuracy of 68.62%.

B.2 Named Entity Recognition

Named Entity Recognition (NER) is a subtask of Information Extraction. The goal is to find the phrases that contain person, location and organization names, times and quantities. Each word is tagged with the *type* of the name as well as its *position* in the name phrase (i.e. whether it is the first item of the phrase or not) in order to represent the boundary information.

We used three different feature sets:

- $S1$ is the set of HMM-features, i.e. the features of the form *"The current word is X and the current label is σ"* and the features of the form *"The previous label is τ and the current label is σ"*.

- $S2$ consists of $S1$ features and spelling attributes of the current word conjoined with the current tags, e.g. *"The current word capitalized and the current tag is σ"*. The list of the spelling attributes, which are mostly adapted from [Bikel et al., 1999] are given in Table B.2.

- $S3$ includes $S2$ features not only for the current word but also for the words within a fixed window of size w. An example of $S3$ features for $w \geq 3$ is *"The previous word ends with a dot and the current tag is σ"*.

Notice that $S2$ is an instance of $S3$ where $w = 1$.

We used a Spanish corpus which was provided for the Special Session of CoNLL2002 on NER [Tjong Kim Sang, 2002]. The data is a collection of news wire articles and is labelled for person, organization, location and miscellaneous names. Thus, micro label set consists of 9 labels: the beginning and continuation of *Person, Organization, Location* and *Miscellaneous* names and nonname tags. By definition, continuation of a name type has to be preceeding by the beginning or the continuation of the same type. We fixed the value of the inter-label dependency features to some value forcing such a combination to be impossible. This value varied across optimization methods.

The training data consists of 7230 sentences of average length 36 [1]. We used *esp.testa* for testing and *esp.testb* for development purposes.

The best results reported on this data set is 78.47% on F1-measure by [Carreras et al., 2002], where separate recognition and classification modules are trained as well as the use of POS tags and gazetteer features such as geographical names, surnames and first names. The results we report are significantly lower than this performance due to these differences. As our goal is not to induce the state-of-the art classifier NER, but rather comparte our methods wrt other classifiers, the current feature setting suffices our needs.

[1] A weather report has been coded as a sentence of length 1238. We divided this sequence into subsequences of average length 36 without dividing a name into pieces.

CUR WORD
SENT INI
TYPE FIRST LETTER
ENDINGS ONE
CAP INI SENT INI
CAP INI DOT END
CAP INI CONTAINS DOT
CAP INI CONTAINS HYPEN
CAP INI CONTAINS DIGIT
CONTAINS DOT CONTAINS DIGIT
CONTAINS DOT CONTAINS HYPEN
CONTAINS DIGIT CONTAINS HYPEN
ALL CAPS
CAP INI
CONTAINS DIGIT
DOT END

Table B.2: Observation attributes used in NER.

B.3 Part-of-Speech Tagging

We used the Penn TreeBank corpus for the part-of-speech tagging experiments. This corpus consists of approximately 7 million words of Part-of-Speech tagged Wall Street Journal articles. We used the standard experiment setup for Penn TreeBank: Sections 2-21 training, Sections 24 development, Section 23 testing. The observation attributes consist of the ones in Table B.2 as well as some more spelling features that were designed specifically for POS tagging task (Table B.3).

ENDS WITH ING
ENDS WITH ED
ENDS WITH EN
ENDS WITH LY
ENDS WITH ER
ENDS WITH EST
ENDS WITH TH
BEGINS WITH WH

Table B.3: More observation attributes used in POS.

Bibliography

Steve Abney, Robert Schapire, and Yoram Singer. Boosting applied to tagging and pp attachment. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 38–45, 1999.

Yasemin Altun, Thomas Hofmann, and Mark Johnson. Discriminative learning for label sequences via boosting. In *Proceedings of Advances in Neural Information Processing Systems (NIPS*15)*, pages 977–984, 2003a.

Yasemin Altun, Mark Johnson, and Thomas Hofmann. Loss functions and optimization methods for discriminative learning of label sequences. In *Proceedings of Empirical Methods of Natural Language Processing (EMNLP)*, pages 145–152, 2003b.

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden markov support vector machines. In *Proceedings of ICML '03: Twentieth international conference on Machine learning*, 2003c.

Yasemin Altun, Thomas Hofmann, and Alexander J. Smola. Gaussian process classification for segmenting and annotating sequences. In *Proceedings of ICML '04: Twenty-first international conference on Machine learning*, 2004a.

Yasemin Altun, Alex J. Smola, and Thomas Hofmann. Exponential families for conditional random fields. In *Proceedings of AUAI '04: Twentieth conference on Uncertainty in artificial intelligence*, pages 2–9, 2004b.

PeterL. Bartlett, MichaelI. Jordan, and JonD. McAuliffe. Large margin classifiers: convex loss, low noise, and convergence rates. In *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, 2003.

Kristin P. Bennett, Michinari Momma, and Mark J. Embrechts. Mark: a boosting algorithm for heterogeneous kernel models. In *Proceedings of KDD '02: Eighth ACM*

SIGKDD international conference on Knowledge discovery and data mining, pages 24–31, 2002.

Steven J. Benson, Lois Curfman McInnes, Jorge Moré, and Jason Sarich. TAO user manual (revision 1.7). Technical Report ANL/MCS-TM-242, Mathematics and Computer Science Division, Argonne National Laboratory, 2004.

Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. le Cun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik. Comparison of classifier methods: A case study in handwriting digit recognition. In *Proceedings of International Conference on Pattern Recognition ICPR94*, pages 77–87, 1994.

Leo Breiman. Prediction games and arcing algorithms. *Neural Computation*, 11(7): 1493–1517, 1999.

Xavier Carreras, Lluís Màrques, and Lluís Padró. Named entity extraction using adaboost. In *Proceedings of The sixth Conference on Natural Language Learning, CoNLL-2002*, pages 167–170, 2002.

Stenley F. Chen and Ronald Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University, 1999.

Massimiliano Ciaramita and Mark Johnson. Supersense tagging of unknown nouns in wordnet. In *Proceedings of Empirical Methods of Natural Language Processing (EMNLP)*, pages 168–175, 2003.

Michael Collins. Discriminative reranking for natural language parsing. In *Proceedings of ICML' 00: Seventeenth International Conference on Machine Learning*, pages 175–182, 2000.

Michael Collins. Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of Empirical Methods of Natural Language Processing (EMNLP)*, pages 1–8, 2002a.

Michael Collins. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In *Harry Bunt, John Carroll and Giorgio Satta, editors, New Developments in Parsing Technology*, 2002b.

Michael Collins. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of Association of Computational Linguistics, ACL'02*, pages 489–496, 2002c.

Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2001.

Michael Collins and Brian Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of Association of Computational Linguistics, ACL'04*, 2004.

Alistair Conkie, Guiseppe Riccardi, and Richard Rose. Prosody recognition from speech utterances using acoustic and linguistic based models of prosodic events. In *Proceedings of Fourth European Conference of Speech Communication and Technology, EUROSPEECH'99*, pages 523–526, 1999.

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

Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

Nello Cristianini and John Shawe-Taylor. *Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

Thomas G. Dietterich. Machine learning for sequential data: A review. In *T. Caelli (Ed.) Lecture Notes in Computer Science. Springer-Verlag.*, 2002.

Richard Durbin, Sean Eddy, Anders Krogh, and Graema Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

Eric Fosler-Lussier and Nelson Morgan. Effects of speaking rate and word frequency on conversational pronunciations. In *Proceedings of the Workshop on Modeling Pronunciation Variation for Automatic Speech Recognition*, pages 35–40, 1998.

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.

Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Computational Learing Theory*, pages 325–332, 1996.

Yoav Freund and Robert E. Schapire. Large margin classification using the perceptron algorithm. In *Computational Learing Theory*, pages 209–217, 1998.

Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. In *Proceedings of ICML '98: Fifteenth International Conference on Machine Learning*, pages 170–178, 1998.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:337–374, 2000.

Mark N. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997.

Federico Girosi. An equivalence between sparse approximation and support vector machines. *Neural Computation*, 10(6):1455–1480, 1998.

James Godfrey, Ellen Holliman, and John McDaniel. SWITCHBOARD: Telephone speech corpus for research and develo pment. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 517–520, 1992.

Steven Greenberg, Dan Ellis, and Joy Hollenback. In praise of imprecision:insights into spoken language gleaned from phonetic transcription of the Switchboard corpus. In *Proceedings of the 1996 CLSP/JHU Workshop on Innovative Techniques for Large Vocabulary Continous Speech Recognition*, 1996.

Michelle Gregory and Yasemin Altun. Using conditional random fields to predict pitch accents in conversational speech. In *Proceedings of ACL'04:Fourty-second Annual Meeting of the Association for Computational Linguistics*, 2004.

Wolfgang Hackbusch. *Iterative Solution of Large Sparse Systems of Equations*. Springer-Verlag, 1994.

Keith Hall and Mark Johnson. Attention shifting for parsing speech. In *Proceedings of Association of Computational Linguistics, ACL'04*, 2004.

Julia Hirschberg. Pitch accent in context: Predicting intonational prominence from text. *Artificial Intelligence*, 63(1-2):305–340, 1993.

Sujun Hua and Zhirong Sun. A novel method for protein secondary structure prediction with high segment overlap measure: Support vector machine approach. *Journal of Molecular Biology*, 308:397–407, 2001.

Raj D. Iyer, David D. Lewis, Robert E. Schapire, Yoram Singer, and Amit Singhal. Boosting for document routing. In *Proceedings of CIKM '00: Ninth international conference on Information and knowledge management*, pages 70–77, 2000.

Mark Johnson, Stuart Geman, Stephen Canon, Zhiyi Chi, and Stefan Riezler. Estimators for stochastic "unification-based" grammars. In *Proceedings of the Thirty-seventh Conference on Association for Computational Linguistics*, pages 535–541, 1999.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, 2000.

Sham Kakade, Yee Whye Teh, and Sam Roweis. An alternate objective function for Markovian fields. In *Proceedings of the Nineteenth International Conference (ICML 2002)*, 2002.

George Kimeldorf and Grace Wahba. Some results on tchebychean spline functions. *Journal of Mathematics Analysis and Applications*, 33:82–95, 1971.

Ulrich Kressel. Pairwise classification and support vector machines. In *B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods - Support Vector Learning*, 1999.

Taku Kudo and Yuji Matsumoto. Chunking with support vector machines. In *Proceedings North American Association of Computational Linguistics NAACL'01*, 2001.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of Eighteenth International Conference on Machine Learning (ICML01)*, pages 282–289, 2001.

John Lafferty, Yan Liu, and Xiaojin Zhu. Kernel conditional random fields: Representation, clique selection, and semi-supervised learning. In *Proceedings of ICML '04: Twenty-first international conference on Machine learning*, 2004.

Zhang Q. Luo and Paul Tseng. Convergence of block coordinate descent method for non-differentiable minimization,. *Journal of Optimization Theory and Applications*, 72(1):7–35, January 2001.

David J. C. Mackay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):698–714, 1992.

Olvi Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24(1-2):15–23, 1999.

Chris Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

Shie Mannor, Ron Meir, and Tong Zhang. Greedy algorithms for classification - consistency, convergence rates and adaptivity. *Journal of Machine Learning Research*, 4:713:741, 2003.

Georges Matheron. Principles of geostatistics. *Economic Geology*, 58:1246–66, 1963.

Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of ICML '00: Seventeenth International Conference on Machine Learning*, pages 591–598, 2000.

Thomas Minka. Algorithms for maximum-likelihood logistic regression. Technical report, CMU, Department of Statistics, TR 758, 2001a.

Thomas Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT Media Lab, 2001b.

Katta G. Murty. *Linear Complementarity, Linear and Nonlinear Programming*. Heldermann Verlag, 1998.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., 1996.

Radford M. Neal. Monte carlo implementation of gaussian process models for bayesian regression and classification, 1997.

Alexander Novikoff. On convergence proofs of perceptron. *Proceedings of the Sypmosium on the Mathematical Theory of Automata*, 12:615–622, 1963.

Manfred Opper and Ole Winther. Gaussian processes for classification: Mean-field algorithms. *Neural Computation*, 12(11):2655–2684, 2000.

James M. Ortega and Werner C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*. Society for Industrial and Applied Mathematics, 2000.

Edgar Osuna, Robert Freund, and Federico Girosi. Training support vector machines: an application to face detection. In *Proceedings of CVPR '97: Conference on Computer Vision and Pattern Recognition (CVPR '97)*, pages 130–138, 1997.

Shimei Pan and Kathleen Keown. Word informativeness and automatic pitch accent modeling, 1999.

Vasin Punyakanok and Dan Roth. The use of classifiers in sequential inference. In *Advances in Neural Information Processing Systems (NIPS)*, pages 995–1001, 2000.

Gunnar Ratsch, Manfred K. Warmuth, Sebastian Mika, Takashi Onoda, Steven Lemm, and Klaus-Robert Muller. Barrier boosting. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 170–179, 2000.

Gunnar Ratsch, Sebastian Mika, and Manfred K. Warmuth. On the convergence of leveraging. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

Robert Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

Robert Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39, 2000.

Robert Schapire, Yoav Freund, Peter Bartlett, and Wee S.Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.

Bernhard Schölkopf, Chris Burges, and Vladimir Vapnik. Extracting support data for a given task. In *Proceedings of First International Conference on Knowledge Discovery and Data Mining.*, 1995.

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

Matthias Seeger, Neil D. Lawrence, and Ralf Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *in Advances in Neural Information Processing Systems*, pages 609–616, 2003.

Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology-NAACL*, 2003.

Alex Smola. Discriminative hidden markov models are conditional random fields. Technical report, Machine Learning Program, National ICT Australia, 2004.

Alex J. Smola and Bernhard Schölkopf. Sparse greedy matrix approximation for machine learning. In *Proc. 17th International Conf. on Machine Learning*, pages 911–918. Morgan Kaufmann, San Francisco, CA, 2000.

Xuejing Sun. Pitch accent prediction using ensemble machine learning. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 953–956, 2002.

Koichi Takeuchi and Nigel Collier. Use of support vector machines in extended named entity recognition. In *Proceedings of Computational Natural Language Learning-2002*, 2002.

Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2004.

Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of Computational Natural Language Learning-2002*, pages 155–158, 2002.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of ICML '04: Twenty-first international conference on Machine learning*, 2004.

Vladimir Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1999.

Hannah Wallach. Efficient training of conditional random fields. Master's thesis, University of Edinburgh, 2002.

Jason Weston and Chris Watkins. Support vector machines for multi-class pattern recognition. In *Proceedings European Symposium on Artificial Neural Networks*, 1999.

Jason Weston, Oliver Chapelle, Andre Elisseeff, Bernhard Schölkopf, and Vladimir Vapnik. Kernel dependency estimation. Technical report, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, August 2002.

Chris K. I. Williams and Matthias Seeger. Using the nystrom method to speed up kernel machines. In *in Advances in Neural Information Processing Systems*, 2000.

Christopher K. I. Williams and David Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12): 1342–1351, 1998.

Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, 1996.

Dekai Wu, Grace Ngai, Marine Carpuat, Jeppe Larsen, and Yongsheng Yang. Boosting for named entity recognition. In *Proceedings of Compuational Natural Language Learning -2002*, 2002.

Tong Zhang. Statistical analysis of some multi-category large margin classification methods. *Journal of Machine Learning Research*, 5:1225–1251, 2004.

Ji Zhu and Trevor Hastie. Kernel logistic regression and the import vector machine. In *Advances in Neural Information Processing Systems (NIPS)*, 2001.

Jaakkola Tommi and Michael Jordan. Computing upper and lower bounds on likelihoods in intractable networks. In *Proceedings of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 340–348, 1996.