

Large Margin Methods for Part of Speech Tagging

Yasemin Altun¹

Max Planck Institute for Biological Cybernetics, Tuebingen, Germany
altun@tuebingen.mpg.de

Abstract. Part of speech tagging, an important component of speech recognition systems, is a sequence labeling problem which involves inferring a state sequence from an observation sequence, where the state sequence encodes a labeling, annotation or segmentation of an observation sequence. In this paper we give an overview of discriminative methods developed for this problem. Special emphasis is put on large margin methods by generalizing multiclass Support Vector Machines and Adaboost to the case of label sequences. Experimental evaluation on Part of Speech Tagging demonstrates the advantages of these models over classical approaches like Hidden Markov Models and their competitiveness with methods like Conditional Random Fields.

1 Introduction

A language model is one of the two important components of speech recognition systems. By assigning probabilities to sequences of words, the language model acts as setting a prior distribution on what utterances can be produced. The acoustic model, which is the second important component of the speech recognition systems, defines a conditional probability distribution of observing an acoustic signal given that a particular sequence of words are uttered. The predominant approach in speech recognition, namely *source-channel model*, combines the distributions given by these two system in order to maximize the joint probability of the acoustic signal and the word sequence and in turn to minimize the error rate.

Until last decade, standard word-based n -gram models (i.e. n -gram models that are trained generatively) were the most common tools for language models. One of the important improvements over these models came from the idea of class-based n -gram models [3]. These models outperform the word-based n -gram models when the training data is small and sparse [13]. Moreover, combining these models with word-based models on large datasets improve the perplexity [12].

In class-based models, the classes can be defined either syntactically or semantically with various refinement levels. In this paper, we consider syntactic classes, in particular Part of Speech (POS) tags. POS tagging is the process of marking up the words in a natural language text with their corresponding

grammatical type (part of speech) e.g. noun, verb, adjective, adverb, based on the word itself as well as its context such as the adjacent words and the related words in a phrase, sentence, or paragraph. Given a sequence of words, we are interested in finding the best POS tag sequence. Since neighboring POS tags give very informative statistics, we model this task as a *sequence labeling* problem, where the goal is to learn a model that uses both the input and the output context to make accurate predictions. In this paper, we consider bigrams of POS tags as output context, however generalization to higher order models is trivial. It is worthwhile to note that one can use the methods described in this paper to learn other class-based models, as long as they respect a sequence model.

An important design choice is how to train the class model, i.e. generatively (using standard log-likelihood techniques) or discriminatively (using generalization of state of the art discriminative learning methods). The generative approach models the joint distribution of observation and label sequences and predict a label sequence of an observation sequence using the conditional probability obtained by the Bayes rule. This model is the well-known Hidden Markov Models (HMMs). Efficient inference and learning in this setting often requires making questionable conditional independence assumptions. More precisely, it is assumed 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). This is particularly true for POS tagging. The discriminative approaches overcome this problem by either modeling the conditional distribution of the response variables given the observation or by directly learning a discriminative function over the observation-label sequence pairs such that the correct prediction achieves a higher score than other label sequences. This approach provides more flexibility in modeling additional dependencies such as direct dependencies of the t -th label on past or future observations.

In this paper, we formulate the sequence labeling problem and outline discriminative learning methods for sequence labeling. In particular, we focus on large margin approaches. We present the generalization of two of the most competitive large margin methods for classification, namely AdaBoost and Support Vector Machines (SVMs) to the problem of sequence labeling. These methods combine the advantages of the discriminative methods described about with the efficiency of dynamic programming. We then apply our methods to POS tagging. Experimental evaluation demonstrates the advantages of our models over classical approaches like Hidden Markov Models and their competitiveness with methods like Conditional Random Fields.

2 Modeling Sequence Labeling

We are interested in the inference of a sequence of labels $\bar{y} = (y_1, \dots, y_t, \dots, y_T) \in \mathcal{Y}$ for a sequence of observations $\bar{x} = (x_1, \dots, x_t, \dots, x_T) \in \mathcal{X}$ where T denotes the length of the sequence. Let $y_t \in \Sigma$, then $\mathcal{Y} = \Sigma^+$ is the set of arbitrary length sequences generated from Σ . In POS tagging, x_t denotes t -th word, y_t denotes

the grammatical type (POS tag) of x_t and Σ denotes the set of all grammatical types, for example as given in Table 1.

The goal is to learn a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ where \mathcal{X} and \mathcal{Y} are observation and label sequence spaces. Hence, our learning problem is a generalization of supervised classification where values of the response variables (labels) are predicted not only with respect to the observations but also with respect to values of other response variables. The approach we pursue is to learn a *discriminant function* $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ where $F(\bar{x}, \bar{y})$ can be interpreted as measuring the compatibility of \bar{x} and \bar{y} . Each discriminant function F induces a mapping f ,

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

where \mathbf{w} denotes a parameter vector and ties are arbitrarily broken. We restrict the space of F to linear functions over some feature representation ϕ , which is defined on the joint input-output space. ϕ is chosen such that it models the sequential structure of \bar{y} and its dependency on \bar{x} . As in HMMs, we assume the model to be stationary. This results in the sharing of \mathbf{w} parameters across components in the sequence.

$$F(\bar{x}, \bar{y}; \mathbf{w}) = \langle \mathbf{w}, \phi(\bar{x}, \bar{y}) \rangle = \sum_t \langle \mathbf{w}, \phi(\bar{x}, \bar{y}; t) \rangle.$$

Given this definition of F , we need to define the features ϕ to complete the design of the architecture. Our main design goal in defining ϕ is to make sure that f can be computed from F efficiently, i.e. using a Viterbi-like decoding algorithm.

Label	Description	Label	Description	Label	Description
CC	Conjunction	CD	Cardinal	DT	Determiner
EX	Existential	FW	Foreign word	IN	Preposition
JJ	Adjective (Adj)	JJR	Adj comparative	JJS	Adj superlative
LS	List marker	MD	Modal	NN	Noun, singular
NNS	Noun, plural	NNP	Proper noun, singular	NNPS	Proper noun, plural
PDT	Predeterminer	POS	Possessive ending	PRP	Personal pronoun
PRP	Possessive pronoun	RB	Adverb (Adv)	RBR	Adv comparative
RBS	Adv superlative	RP	Particle	SYM	Symbol
TO	to	UH	Interjection	VB	Verb
VBD	Verb, Past	VBG	Verb, Pres Participle	VBN	Verb, Past Participle
VBP	Verb, Present	VBZ	Verb, 3P Sing Pres	WDT	Wh-determiner
WP	Wh-pronoun	WP	Possessive wh-pronoun	WRB	Wh-adverb

Table 1. POS tags used in the Penn TreeBank Project

2.1 Feature Representation

We extract two types of features from an observation-label sequence pair (\bar{x}, \bar{y}) . The first type of features capture inter-label dependencies

$$\phi_{s\sigma\sigma'}^1(\bar{x}, \bar{y}; t) = \mathbb{1}_{\{y_s=\sigma\}} \mathbb{1}_{\{y_t=\sigma'\}},$$

where $\sigma, \sigma' \in \Sigma$ denote labels for single observations and $\mathbb{1}_{\{\cdot\}}$ denotes the indicator function. Here, $\phi_{s\sigma\sigma'}^1(\bar{x}, \bar{y}; t)$ corresponds to a feature $\phi_k^1(\bar{x}, \bar{y}; t)$ where k is indexed by (s, σ, σ') , the label σ at position s and the label σ' . 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 an efficient dynamic programming algorithm.

The second type of features capture the dependency between a label and the observation sequence. Let φ be the set of observation attributes relevant for the particular application. For example, in POS tagging, the information that a word ends with ‘-ing’ can be very informative and therefore should be an attribute of the observation. We describe the attributes for POS tagging problem in detail in Section 5.1. The observation-label dependency features combine the attributes with individual labels, $\sigma \in \Sigma$. These features indicate the statistics of how often an attribute occurs with a particular label conjunctively

$$\phi_{sr\sigma}^2(\bar{x}, \bar{y}; t) = \mathbb{1}_{\{y_t=\sigma\}} \varphi_r(x_s).$$

Again $\phi_{sr\sigma}^2(\bar{x}, \bar{y}; t)$ corresponds to a feature $\phi_k^2(\bar{x}, \bar{y}; t)$ where k is indexed by (s, r, σ) , the attribute r of the observation at position s and the label σ . For example, in POS tagging, if $\varphi_r(x)$ denotes whether the word x is ending with ‘-ing’, σ is ‘VBG’ (Verb, Present Participle) and $s = t - 1$, then the feature $\phi_{sr\sigma}^2(\bar{x}, \bar{y}; t)$ denotes whether the t -th label in the label sequence \bar{y} is a ‘VBG’ and the previous word ($t - 1^{th}$ word) ends with ‘-ing’.

Since in the discriminative framework we **condition on** the observation sequence (as opposed to *generating* it as in the generative framework), 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 for observation-label features. The features for which $s \neq t$ are usually called *overlapping* features (or *sliding window* features), since the same input attribute $\varphi_k(x_s)$ is encoded in the model multiple times within different contexts. These features are used widely in discriminative sequence learning [11, 6] and have been shown to improve classification accuracy in different applications.

Finally, our feature map is the concatenation of the inter-label and observation-label dependency features (where \oplus denotes concatenation),

$$\phi(\bar{x}, \bar{y}; t) = \phi^1(\bar{x}, \bar{y}; t) \oplus \phi^2(\bar{x}, \bar{y}; t).$$

Various generalizations of this feature representation are possible, such as extracting different input features dependent on the relative distance $|ts|$ in the chain. We use this representation as a prototypical example. In Section 4, we investigate how to extend this via kernel functions.

2.2 Empirical Risk Minimization

We investigate a supervised learning setting, where observation-label sequence pairs (\bar{x}, \bar{y}) are generated from a fixed but unknown distribution P over $\mathcal{X} \times \mathcal{Y}$ and the goal is to find the discriminant function F such that the *risk*, i.e. the cost of making an incorrect prediction, is minimized. Since P is unknown, we use a sample of input-output pairs $S = \{(\bar{x}^1, \bar{y}^1), \dots, (\bar{x}^n, \bar{y}^n)\}$, that are assumed to be generated i.i.d. according to P and we minimize the empirical risk with respect to S .

The standard zero-one risk typically used in classification is not appropriate for sequence labeling problem, since zero-one risk cannot capture the degrees of incorrect prediction of sequences. In particular, in POS tagging the evaluation measure is the Hamming distance, which measures the number of incorrect components of the predicted sequence. The empirical Hamming risk is given by

$$\mathbb{R}^{\text{hm}}(\mathbf{w}; S) = \sum_{i=1}^n \sum_{t=1}^T \mathbb{1}_{\{y_t^i \neq f(\bar{x}^i; \mathbf{w})_t\}},$$

where $f(\bar{x}; \mathbf{w})_t$ denotes the t -th component of the predicted label sequence.

Another risk measure that can be used for label sequences is the rank function [9, 15] which measures the fraction of incorrect label sequences that are ranked higher than the correct one. The empirical rank risk is given by

$$\mathbb{R}^{\text{rk}}(\mathbf{w}; S) = \sum_{i=1}^n \sum_{\bar{y} \neq \bar{y}^i} \mathbb{1}_{\{F(\bar{x}^i, \bar{y}; \mathbf{w}) \geq F(\bar{x}^i, \bar{y}^i; \mathbf{w})\}}.$$

2.3 Conditional Random Fields and Sequence Perceptron

Due to the nonconvexity of the risk functions, it is common to optimize convex surrogate functions that are upper bounds of the risk functions. Logarithmic, exponential and hinge loss functions are the most common surrogate functions of the zero-one risk. Among these functions, the logarithmic loss has been proposed for sequence labeling, where the goal is to minimize the negative conditional log-likelihood of the data,

$$\begin{aligned} \mathbb{R}^{\text{log}}(\mathbf{w}; S) &= - \sum_{i=1}^n \log p(\bar{y}^i | \bar{x}^i; \mathbf{w}), \text{ where} \\ p(\bar{y} | \bar{x}; \mathbf{w}) &= \frac{e^{F(\bar{x}, \bar{y}; \mathbf{w})}}{\sum_{\bar{y}' \in \mathcal{Y}} e^{F(\bar{x}, \bar{y}'; \mathbf{w})}}. \end{aligned} \quad (2)$$

This gives the optimization problem of the widely used Conditional Random Fields (CRFs) [11]. \mathbb{R}^{log} is a convex function and is generally optimized using gradient methods. The gradient of $\mathbb{R}^{\text{log}}(\mathbf{w}; S)$ is given by

$$\nabla_{\mathbf{w}} \mathcal{R}^{\text{log}} = \sum_i \mathbf{E}_{\bar{y} \sim p(\cdot | \bar{x})} \left[\sum_t \phi(\bar{x}, \bar{y}; t) | \bar{x} = \bar{x}^i \right] - \sum_t \phi(\bar{x}^i, \bar{y}^i; t). \quad (3)$$

The expectation of the feature functions can be computed using the Forward-Backward algorithm. When it is not regularized, \mathbb{R}^{log} is prone to overfitting, especially with noisy data. For this reason, it is common to regularize this objective function by adding a Gaussian prior (a term proportional to the squared L_2 norm of the parameter \mathbf{w}) to avoid overfitting the training data [10, 4].

Another discriminative learning method for sequences is Hidden Markov Perceptron [6], which generalizes Perceptron algorithm for sequence labeling. This approach is outlined in Algorithm 1. Investigating the parameter update of Algo-

Algorithm 1 Hidden Markov Perceptron algorithm.

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

rithm 1, we can view Hidden Markov Perceptron as an online approximation for CRFs, where the expectation term in (3) is approximated with the contribution from the most likely label. This approximation can be tight, if the distribution $p(\bar{y}|\bar{x})$ is peaked. Then, the contribution of the most likely label can dominates the expectation values.

In this paper, we focus on the exponential loss and the hinge loss as surrogate functions of the empirical risk. This leads to the generalizations of two well-known large margin methods, Boosting and Support Vector Machines, to label sequence learning problems. These methods, either explicitly or implicitly, maximize the margin of the data, where the margin for an observation sequence \bar{x} is defined as

$$\gamma(\bar{x}, \bar{y}; \mathbf{w}) = F(\bar{x}, \bar{y}; \mathbf{w}) - \max_{\bar{y}' \neq \bar{y}} F(\bar{x}, \bar{y}'; \mathbf{w}). \quad (4)$$

This is a simple generalization of the multiclass separation margin [7] and it measures the differences of the compatibility scores of (\bar{x}, \bar{y}) and (\bar{x}, \bar{y}') where \bar{y}' is the most competitive labeling of \bar{x} that is not \bar{y} . Intuitively, when this value is positive for the correct labeling of an observation sequence \bar{x} , the prediction function (1) makes a correct prediction for \bar{x} . The goal of large margin classifiers is to maximize the margin all training instances in the sample, which in turn may lead to correct predictions not only on the sample but also on unobserved observations.

3 Sequence Boosting

AdaBoost [8] is one of the most powerful learning ideas introduced to the machine learning community in the last decade. We present a generalization of AdaBoost to label sequence learning. This method, which we refer as Sequence AdaBoost, was first proposed in [1]. It employs the advantages of large margin methods as well as its implicit feature selection property. This aspect is quite important for efficient inference in high dimensional feature spaces and for cascaded systems, for which speech recognition is a canonical example. Sequence AdaBoost also makes use of the availability of a dynamic programming (DP) algorithm for efficient learning and inference due to the Markov chain dependency structure between labels.

3.1 Objective Function

The key idea of Boosting is to obtain a high accuracy classifier by combining weak learners that are trained on different distributions of the training examples. In Sequence AdaBoost, we consider the features functions ϕ defined in Section 2.1 as weak learners and learn their combination coefficients \mathbf{w} by training over the sample S with distributions D . D assigns a weight to each training instance \bar{x}^i and its possible labelings $\bar{y} \in \mathcal{Y}^{T_i}$, where T_i is the length of \bar{x}^i .

In particular, motivated by [16], we propose optimizing the exponential risk function on the sequential data weighted according to D ,

$$\mathbb{R}^{\text{exp}}(\mathbf{w}; S) \equiv \sum_{i=1}^n \sum_{\bar{y} \neq \bar{y}^i} D(i, \bar{y}) e^{F(\bar{x}^i, \bar{y}; \mathbf{w}) - F(\bar{x}^i, \bar{y}^i; \mathbf{w})}.$$

Intuitively, by minimizing this function, we increase the compatibility of the correct observation-label sequence pairs, while decreasing the compatibility of all the other incorrect pairs. Due to the exponential function, \mathbb{R}^{exp} is dominated by the term(s) corresponding to (x^i, \bar{y}) with largest relative compatibility score for $\bar{y} \neq \bar{y}^i$ ($\max_{\bar{y} \neq \bar{y}^i} F(\bar{x}^i, \bar{y}; \mathbf{w}) - F(\bar{x}^i, \bar{y}^i; \mathbf{w}) = -\gamma(\bar{x}^i, \bar{y}^i; \mathbf{w})$). Hence, \mathbb{R}^{exp} performs an implicit margin maximization where the margin is defined in (4). Moreover, it is easy to show that \mathbb{R}^{exp} is a convex surrogate of the rank risk \mathbb{R}^{rk} .

Proposition 1. *The exponential risk is an upper bound on the rank risk, $\mathbb{R}^{\text{rk}} \leq \mathbb{R}^{\text{exp}}$.*

Proof. (i) If $F(\bar{x}^i, \bar{y}^i; \mathbf{w}) > F(\bar{x}^i, \bar{y}; \mathbf{w})$ then $\mathbb{1}_{\{F(\bar{x}^i, \bar{y}; \mathbf{w}) \geq F(\bar{x}^i, \bar{y}^i; \mathbf{w})\}} = 0 \leq e^z$ for any z .

(ii) Otherwise, $\mathbb{1}_{\{F(\bar{x}^i, \bar{y}; \mathbf{w}) \geq F(\bar{x}^i, \bar{y}^i; \mathbf{w})\}} = 1 = e^0 \leq e^{F(\bar{x}^i, \bar{y}; \mathbf{w}) - F(\bar{x}^i, \bar{y}^i; \mathbf{w})}$.

Performing a weighted sum over all instances and label sequences \bar{y} completes the proof.

Interestingly, when D is uniform for all \bar{y} of each training instance \bar{x}^i , this loss function is simply the weighted inverse conditional probability of the correct

label sequence as pointed out by [11],

$$\mathbb{R}^{\text{exp}}(\mathbf{w}; S) = \sum_i \left[\frac{1}{p(\bar{y}^i | \bar{x}^i; \mathbf{w})} - 1 \right].$$

where p is defined in (2). Introducing the distribution D allows the loss function to focus on difficult training instances.

3.2 Optimization Method

Clearly, \mathbb{R}^{exp} is a convex function and can be optimized using standard gradient based methods. Instead, we present a boosting algorithm that generalizes the AdaBoost.MR algorithm for multiclass classification [17] to label sequence learning. This is a sparse greedy approximate algorithm that optimizes an upper bound on \mathbb{R}^{exp} with sequential updates. In general, Sequence AdaBoost leads to a sparse feature representation due to these sequential updates.

As standard AdaBoost.MR, Sequence AdaBoost forms a linear combination of weak learners, an ensemble, by selecting the best performing weak learner and its optimal weight parameter to minimize \mathbb{R}^{exp} at each round. It also maintains a weight distribution over observation-label sequence pairs and updates this distribution after every round of boosting such that training instances \bar{x}^i with small margin and their most competitive incorrect labelings are weighted higher. This allows the weak learners in the next round to focus on instances that are difficult to predict correctly.

Weight distribution of examples We define a sequence of distributions $(D_1, \dots, D_r, \dots, D_R)$ where D_r denotes the distribution over (\bar{x}^i, \bar{y}) pairs in the r -th round of Boosting and R is the total number of rounds. $D_r(i, \bar{y})$ is given by the recursive formula of D_{r-1} , the feature selected in the r -th round (denoted by $\phi_{k(r)}$) and its optimal weight update (denoted by $\Delta w_{k(r)}$)

$$D_{r+1}(i, \bar{y}) \equiv \frac{D_r(i, \bar{y})}{Z_r} e^{\Delta w_{k(r)} (\sum_t \phi_{k(r)}(\bar{x}^i, \bar{y}; t) - \phi_{k(r)}(\bar{x}^i, \bar{y}^i; t))}, \quad (5)$$

where Z_r is the normalization constant. We set $D_0(i, \bar{y})$ to be uniform over all $\bar{y} \neq \bar{y}^i$ and $D_0(i, \bar{y}^i) = 0$ for all i . We denote \mathbf{w}_r as the sum of all weight updates from round 1 to r . One can relate \mathbb{R}^{exp} to the weight distribution.

Proposition 2. \mathbb{R}^{exp} is the product of the normalization constants, $\mathbb{R}^{\text{exp}}(\mathbf{w}_R; S) = \prod_{r=1}^R Z_r$.

Proof. It can be shown that $D_r(i, \bar{y}) = \exp [F(\bar{x}^i, \bar{y}; \mathbf{w}_r) - F(\bar{x}^i, \bar{y}^i; \mathbf{w}_r)] / Z_r$. Then, we see that $\mathbb{R}^{\text{exp}}(\mathbf{w}_R; S) = Z_R$. The recursive definition of the distribution (5) reveals that $Z_R = \prod_r Z_r$.

It can be shown that $D_r(i, \bar{y})$ can be decomposed into a relative weight for each training instance, $D_r(i)$, and a relative weight of each sequence, $\pi_{ri}(\bar{y})$, where

$$\begin{aligned} D_r(i, \bar{y}) &= D_r(i)\pi_{ri}(\bar{y}), \\ D_r(i) &\equiv \frac{[p(\bar{y}^i|\bar{x}^i; \mathbf{w}_r)^{-1} - 1]}{\sum_j [p(\bar{y}^j|\bar{x}^j; \mathbf{w}_r)^{-1} - 1]}, \forall i \text{ and} \\ \pi_{ri}(\bar{y}) &\equiv \frac{p(\bar{y}|\bar{x}^i; \mathbf{w}_r)}{1 - p(\bar{y}^i|\bar{x}^i; \mathbf{w}_r)}, \forall \bar{y} \neq \bar{y}^i. \end{aligned}$$

The relative weight of each training instance $D_r(i)$ enforces the optimization problem to focus on difficult instances \bar{x}^i in the sense that the probability of the correct labeling $p(\bar{y}^i|\bar{x}^i; \mathbf{w}_r)$ is low with respect to the current parameter values \mathbf{w}_r . $D_r(i) \rightarrow 0$ as we approach the perfect prediction case of $p(\bar{y}^i|\bar{x}^i; \mathbf{w}_r) \rightarrow 1$. The relative weight of each sequence $\pi_{ri}(\bar{y})$ is simply the re-normalized conditional probability $p(\bar{y}|\bar{x}^i; \mathbf{w}_r)$. For a given \bar{x}^i , it enforces the optimization problem to focus on the most competitive labelings wrt \mathbf{w}_r .

Selecting optimal feature and its weight update Proposition 2 motivates a boosting algorithm which minimizes Z_r w.r.t. a weak learner $\phi_{r(k)}$ and its optimal weight update $\Delta w_{r(k)}$ at each round. In order to simplify the notation, we define $u_k(\bar{x}^i, \bar{y}) \equiv \sum_t \phi_k(\bar{x}^i, \bar{y}; t) - \phi_k(\bar{x}^i, \bar{y}^i; t)$. Moreover, we drop the round index r focusing on one round of Boosting. Given these definitions, we can rewrite Z as

$$Z(\Delta w) \equiv \sum_i D(i) \sum_{\bar{y} \neq \bar{y}^i} \pi_i(\bar{y}) \exp[\Delta w u(\bar{x}^i, \bar{y})] \quad (6)$$

Minimizing Z exactly on large datasets and/or large number of features is not tractable, since this requires programming run with accumulators [11] for every feature. To overcome this problem, we propose minimizing an upper bound on Z that is linear in ϕ . Defining U_k and L_k as the maximum and minimum values of u_k for all \bar{x}^i and all \bar{y} , one can write an upper bound of Z as

$$Z(\Delta w) \leq s e^{\Delta w L} + (1 - s) e^{\Delta w U}, \text{ where} \quad (7)$$

$$s \equiv \sum_i D(i) \sum_{\bar{y} \neq \bar{y}^i} \pi_i(\bar{y}) \frac{U - u(\bar{x}^i, \bar{y})}{U - L}. \quad (8)$$

We refer the interested reader to [1] for the derivation of this bound. Note that $0 \leq s \leq 1$ and it is large when $\sum_t \phi_k(\bar{x}^i, \bar{y}^i; t) > \sum_t \phi_k(\bar{x}^i, \bar{y}; t)$. Then s can be interpreted as a measure of correlation of ϕ and the correct label sequences with respect to the distribution $D(i, \bar{y})$. We call s the *pseudo-accuracy* of feature ϕ .

The advantage of (7) over Z is that one can compute s for all features simultaneously in by running Forward-Backward algorithm once, since s is simply

an expectation of the sufficient statistics. The optimal weight update Δw that minimizes (7) is given by

$$\Delta w = \frac{1}{U - L} \log \left(\frac{-Ls}{U(1-s)} \right). \quad (9)$$

Algorithm 2 describes the iterative procedure of Sequence AdaBoost. In each iteration, Forward-Backward algorithm is performed over the training data. This provides s_k values of all features. The optimal updates are computed with respect to (9). These values are plugged into (7) to find the feature with minimum $Z(\Delta w)$. This feature is added to the ensemble and the distribution is updated according to (5).

Algorithm 2 Sequence AdaBoost.

- 1: Initialize $D_0(i, \bar{y}) = \frac{1}{\sum_j (|\mathcal{Y}^{T_j}| - 1)}$ for $\bar{y} \neq \bar{y}^i$, $D_0(i, \bar{y}^i) = 0$.
 - 2: Initialize $\mathbf{w} = 0$.
 - 3: **for** $r = 1, \dots, R$ **do**
 - 4: Perform Forward-Backward algorithm over S to compute $s_k, \forall k$ via (8)
 - 5: Select ϕ_k that minimizes the upper bound in (7).
 - 6: Compute optimal increment Δw_k using (9).
 - 7: Update weight $w_k \leftarrow w_k + \Delta w_k$.
 - 8: Update D_{r+1} using (5).
 - 9: **end for**
-

A Tighter Approximation In sequence learning, U_k and $-L_k$ can be as large as the maximum length of the training sequences if the feature ϕ_k occurs frequently. Then, the bound in (7) can be very loose, especially when there exists a very long sequence in the training set. The bound can be refined with piece-wise linear functions defined with respect to each training instance. Defining U_k^i and L_k^i as the maximum and minimum values of u_k for all \bar{y} for a fixed observation sequence \bar{x}^i , we give a tighter bound on Z as

$$Z(\Delta w) \leq \sum_i D(i) \left(s^i e^{\Delta w L^i} + (1 - s^i) e^{\Delta w U^i} \right), \text{ where} \quad (10)$$

$$s^i \equiv \sum_{\bar{y} \neq \bar{y}^i} \pi_i(\bar{y}) \frac{U^i - u(\bar{x}^i, \bar{y})}{U^i - L^i}. \quad (11)$$

Figure 1 illustrates the difference between the two bounds on an example of 4 observation-label sequence pairs of equal length. The line labeled **LB** is the loose bound given by (7). The gap between Z (6) and the loose bound (7) is four times the area between **LB** and the exponential function (blue line). The gap between Z and the tight bound (10) is the sum of the area between each

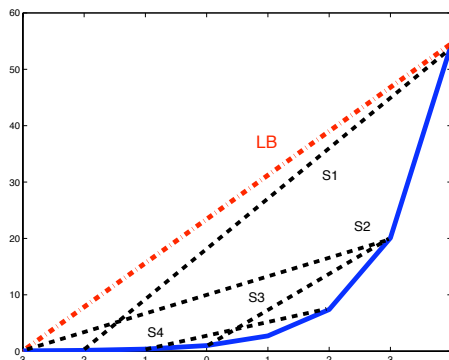


Fig. 1. An example of the tight and loose bounds on the normalization constant Z .

line S^* and the exponential curve, which is much smaller than the gap of the loose bound.

Unfortunately, there is no analytical solution of Δw that minimizes (10). However, since it is convex in Δw , it can be minimized with a simple line search. The algorithm optimizing this tighter bound is given by changing Lines 4, 5 and 6 in Algorithm 2 accordingly.

Algorithmic Details 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 $\Delta w_{k(r)}$, once $\phi_{k(r)}$ has been selected. For example, Z_r can be optimized exactly for the feature $\phi_{k(r)}$ that minimizes the upper bound on Z_r . This involves performing a dynamic program with accumulators on the sequences in which $\phi_{k(r)}$ can be observed.

As observed in [5], the computation can be reduced drastically by taking sparsity into account. To achieve this, we perform some bookkeeping before Sequence AdaBoost. In particular, for each training instance \bar{x}^i we keep a list of features that can be observed in (\bar{x}^i, \bar{y}) pairs for any \bar{y} . Moreover, for each feature ϕ_k we keep a list of the training sequences \bar{x}^i in which ϕ_k can be observed. During Boosting, we store the pseudo-accuracy s_k of all features. Then, at each round we only need to update s_k of features ϕ_k that may co-occur with $\phi_{k(r-1)}$. This can be achieved efficiently by restricting the dynamic program to only the training instances in which $\phi_{k(r-1)}$ can be observed using the lists described above.

4 Hidden Markov Support Vector Machines

In this section, we present the second large margin method for sequence which is a generalization of SVMs to label sequence learning first presented in [2]. 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.

4.1 Objective Function

In HM-SVMs, we define an optimization problem that uses the margin γ where the margin is defined in (4). Note that $\gamma(\bar{x}, \bar{y}; \mathbf{w}) > 0$ implies that the correct label sequence receives the highest score. In the maximum margin setting, the goal is to find a hyperplane that not only assigns maximum score to the correct label sequence of each observation sequence, but also separates them with some margin, which is measured by γ . Then, we propose maximizing the minimum of the margins,

$$\mathbf{w}^* = \operatorname{argmax}_{\mathbf{w}} \min_i \gamma(\bar{x}^i, \bar{y}^i; \mathbf{w}). \quad (12)$$

As in standard SVMs, the margin can be made arbitrary large by scaling \mathbf{w} , 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 \mathbf{w} , $\|\mathbf{w}\|^2$ subject to the margin constraints. In order to allow margin violations for non-separable cases, we add slack variables $\xi(i)$ for every training sequence

$$\begin{aligned} \text{SVM}_1: \min_{\mathbf{w}, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^n \xi(i), \quad \text{s.t. } \xi(i) \geq 0, \forall i \\ F(\bar{x}^i, \bar{y}^i; \mathbf{w}) - F(\bar{x}^i, \bar{y}; \mathbf{w}) \geq 1 - \xi(i), \quad \forall i, \bar{y} \neq \bar{y}^i, \end{aligned}$$

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

Proposition 3. *The hinge risk $\mathbb{R}^{svm}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \xi(i; \mathbf{w})$ is an upper bound on 0-1 risk.*

Proof. (i) If $\xi(i; \mathbf{w}) < 1$, then one gets $F(\bar{x}^i, \bar{y}^i; \mathbf{w}) - \max_{\bar{y} \neq \bar{y}^i} F(\bar{x}^i, \bar{y}; \mathbf{w}) = \gamma(\bar{x}^i, \bar{y}^i) > 0$ which means the data point is correctly classified and $\mathbb{1}_{\{f(\bar{x}^i; \mathbf{w}) \neq \bar{y}^i\}} = 0 \leq \xi(i; \mathbf{w})$.

(ii) If $\xi(i; \mathbf{w}) \geq 1$, then the bound holds automatically, since $\mathbb{1}_{\{f(\bar{x}^i; \mathbf{w}) \neq \bar{y}^i\}} \leq 1 \leq \xi(i; \mathbf{w})$.

Summing over all i completes the proof.

One can generalize SVM_1 by imposing the margin to be more than some cost function, such as Hamming distance [20, 19]. Then, the margin constraints in SVM_1 are stated as

$$F(\bar{x}^i, \bar{y}^i; \mathbf{w}) - F(\bar{x}^i, \bar{y}; \mathbf{w}) \geq \Delta(\bar{y}, \bar{y}^i) - \xi(i), \quad \forall i, \bar{y} \neq \bar{y}^i, \quad (13)$$

where Δ denotes the Hamming distance. By incorporating the cost function into the optimization problem, one can hope to achieve better performance than optimizing a surrogate function of the risk. This formulation provides an upper bound on the Hamming loss.

Proposition 4. *The risk $\mathbb{R}^{hs}(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n \xi^{hm}(i; \mathbf{w})$ is an upper bound on \mathbb{R}^{hm} , where ξ^{hm} is the optimal solution of SVM_1 with (13) constraints.*

Proof. $\xi^{hm}(i; \mathbf{w}) = \max\{0, \Delta(\bar{y}, \bar{y}^i) - \gamma(\bar{x}^i, \bar{y}^i; \mathbf{w})\}$ is guaranteed to upper bound $\Delta(\bar{y}, \bar{y}^i)$ for \bar{y} such that $\gamma(\bar{x}^i, \bar{y}^i; \mathbf{w}) \leq 0$.

As an alternative to SVM_1 , one can introduce one slack variable for every training instance and every sequence \bar{y} leading to a similar QP with slack variables $\xi(i, \bar{y}; \mathbf{w}) = \max\{0, 1 - [F(\bar{x}^i, \bar{y}^i; \mathbf{w}) - F(\bar{x}^i, \bar{y}; \mathbf{w})]\}$. This provides an upper bound on the rank loss, \mathbb{R}^{rk} .

Proposition 5. $\frac{1}{n} \sum_{i=1}^n \sum_{\bar{y} \neq \bar{y}^i} \xi(i, \bar{y}; \mathbf{w}) \geq \mathbb{R}^{rk}(\mathbf{w}, \mathbf{w})$.

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

4.2 Optimization Method

In this paper, we focus on SVM_1 , as we expect the number of active inequalities in SVM_1 to be much smaller compared to the latter formulation since SVM_1 only penalizes the *largest* margin violation for each example. The dual QP of SVM_1 is given by solving the Lagrangian with respect to the Lagrange multipliers $\alpha_{(i, \bar{y})}$ for every margin inequality.

$$\mathbf{DSVM}_1: \max_{\alpha} \frac{1}{2} \sum_{i, \bar{y} \neq \bar{y}_i} \alpha_{(i, \bar{y})} \quad (14a)$$

$$- \frac{1}{2} \sum_{i, \bar{y} \neq \bar{y}_i} \sum_{j, \bar{y}' \neq \bar{y}_j} \alpha_{(i, \bar{y})} \alpha_{(j, \bar{y}')} \langle \delta\phi(i, \bar{y}), \delta\phi(j, \bar{y}') \rangle$$

$$\text{s.t. } \alpha_{(i, \bar{y})} \geq 0, \sum_{\bar{y} \neq \bar{y}_i} \alpha_{(i, \bar{y})} \leq C, \quad \forall i, \bar{y}, \quad (14b)$$

where $\delta\phi(i, \bar{y}) = \phi(i, \bar{y}_i) - \phi(i, \bar{y})$. Note that the optimality equation of \mathbf{w} is given by

$$\mathbf{w}^* = \sum_j \sum_{\bar{y}} \alpha_{(j, \bar{y})} \delta\phi(j, \bar{y}).$$

Then, both the optimization problem \mathbf{DSVM}_1 and the discriminative function $F(\bar{x}, \bar{y})$ is computed with respect to inner products of two observation-label sequence pairs. As a simple consequence of the linearity of the inner products,

one can replace this with some kernel functions. The only restriction on the types of kernel function to consider here is to compute the argmax operator in the prediction function (1) efficiently. We discuss this issue later in Section 4.3

\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 \bar{x}^i , $\alpha_{(i,\cdot)}$, are independent of the parameters of other observation sequences, $\alpha_{(j,\cdot)}$. Our optimization method exploits this dependency structure of the parameters and the anticipated sparseness of the solution to achieve computational efficiency.

4.3 Algorithm

We propose using a row selection procedure to incrementally add inequalities to the problem. We maintain an active set of label sequences, S^i , for every instance which are initially $\{\bar{y}^i\}$, 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 label sequence \bar{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 [18]. The satisfaction of the margin constraint by this label sequence implies that all the other label sequences satisfy their margin constraints as well. If \bar{y} violates the margin constraint, we add it into the working set of \bar{x}^i and optimize the quadratic program with respect to the Lagrange parameters in the working set of \bar{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 3.

It is possible to show that this algorithm terminates in polynomial number of steps with respect to the length of sequences, if the constraints are satisfied up to some precision [20]. 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.

Algorithmic Details One can replace the inner products of two observation-label sequence pairs in \mathbf{DSVM}_1 and in the discriminative function F with a

Algorithm 3 Row selection optimization for HM-SVM.

```

1: Initialize  $S^i \leftarrow \{\bar{y}^i\}$ ,  $\alpha_{(i,\cdot)} = \mathbf{0}$ ,  $\forall i$ 
2: repeat
3:   for  $i = 1, \dots, n$  do
4:     Compute  $\bar{y}' = \arg \max_{\bar{y} \neq \bar{y}^i} F(\bar{x}^i, \bar{y}; \alpha)$ 
5:     if  $F(\bar{x}^i, \bar{y}^i; \alpha) - F(\bar{x}^i, \bar{y}'; \alpha) < 1 - \xi_i$  then
6:        $S^i \leftarrow S^i \cup \{\bar{y}'\}$ 
7:       Optimize DSVM1 over  $\alpha_{(i,\bar{y})}$ ,  $\forall \bar{y} \in S^i$ 
8:     end if
9:     Remove  $\bar{y} \in S^i$  with  $\alpha_{(i,\bar{y})} < \epsilon$ 
10:  end for
11: until no margin constraint is violated

```

kernel function. Let us recall the feature representation presented in Section 2.1,

$$\phi(\bar{x}, \bar{y}) = \sum_t \phi(\bar{x}, \bar{y}; t) = \sum_t \phi^1(\bar{x}, \bar{y}; t) \oplus \phi^2(\bar{x}, \bar{y}; t).$$

Then, the inner product can be written as

$$\langle \phi(\bar{x}, \bar{y}), \phi(\bar{x}', \bar{y}') \rangle = \sum_{s,t} \mathbb{1}_{\{y_{s-1}=y'_{t-1} \wedge y_s=y'_t\}} + \sum_{s,t} \mathbb{1}_{\{y_s=y'_t\}} k(x_s, x'_t) \quad (15)$$

$$\text{where } k(x_s, x'_t) = \langle \varphi(x_s), \varphi(x'_t) \rangle. \quad (16)$$

Hence the similarity between two sequences depends on the number of common two-label fragments as well as the inner product between the feature representation of patterns with common label. Here, one can replace the inner product on the input attributes with any kernel function. Although one can represent the indicator functions of labels as inner products, it is not beneficial to replace them with nonlinear kernels as this can introduce long range dependencies across labels and therefore it may render the dynamic programming algorithm intractable.

Given this decomposition, it is easy to see that the discriminative function F also decomposes into two terms, $F(\bar{x}, \bar{y}) = F_1(\bar{x}, \bar{y}) + F_2(\bar{x}, \bar{y})$, where

$$F_1(\bar{x}, \bar{y}) = \sum_{\sigma, \tau} \delta(\sigma, \tau) \sum_s \mathbb{1}_{\{y_{s-1}=\sigma \wedge y_s=\tau\}}, \quad (17a)$$

$$\delta(\sigma, \tau) = \sum_{i, \bar{y}'} \alpha_{(i, \bar{y}')} \sum_{t: y_{t-1}^i \neq \sigma \vee y_t^i \neq \tau} \mathbb{1}_{\{y_{t-1}^i=\sigma \wedge y_t^i=\tau\}} \quad (17b)$$

and where

$$F_2(\bar{x}, \bar{y}) = \sum_{s, \sigma} \mathbb{1}_{\{y_s=\sigma\}} \sum_{i, t} \beta(i, t, \sigma) k(x_s, x_t^i), \quad (18a)$$

$$\beta(i, t, \sigma) = \begin{cases} 0 & \text{if } y_t^i = \sigma \\ \sum_{\bar{y}} \mathbb{1}_{\{y_t=\sigma\}} \alpha_{(i, \bar{y})} & \text{o.w.} \end{cases} \quad (18b)$$

Thus, we only need to keep track of the number of times an individual 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 \bar{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 transition matrix is given by δ and the observation matrix is given by

$$H_{s\sigma}^i = \sum_j \sum_t \beta(j, t, \sigma) k(x_s^i, x_t^j) \quad (19)$$

Given these two matrices, Viterbi decoding amounts to finding the values that maximizes the potential function at each position in the sequence.

5 Experiments

5.1 Data and Features for Part of Speech Tagging

We experimentally evaluated Sequence AdaBoost and Hidden Markov Support Vector Machines on part-of-speech tagging (POS). 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 are the standard features used in computational linguistics, such as the word identity, sentence initial, last (n)-letters of the word, contains dot/hyphen/digit, etc. Overlapping features with respect to these attributes are extracted from a window of size 3.

5.2 Results of Sequence AdaBoost

We ran experiments comparing optimization of \mathbb{R}^{\log} and \mathbb{R}^{\exp} using the BFGS, a quasi-Newton convex optimization method [14]. We also performed experiments with different formulations of Sequence AdaBoost. Note that the BFSG method performs parallel updates, thus uses all the features available to the model, whereas Sequence AdaBoost perform sequential updates leading to sparse solution. The number of Boosting rounds R is selected by cross-validation. We used the tight bound (10) to select the features and optimized Z exactly to find the optimal weight update for the selected feature, unless stated otherwise.

We experimented the feature sparseness properties of Sequence AdaBoost, by defining incremental sets of features. The first set $S1$ consists of only HMM features, i.e. inter-label dependency features (POS tag pairs) as well as word identity features (thus we only do not use the other attributes defined above in $S1$). $S2$ also includes features generated from all attributes as well as the inter-label features. These results are summarized in Table 2. Sequence AdaBoost performs substantially worse than the BFSG optimization of \mathbb{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, Sequence AdaBoost needs to include almost all weak learners in the ensemble and cannot exploit feature sparseness. When 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 problem is necessary for the success of Sequence AdaBoost.

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

Table 2. Accuracy of POS tagging on PennTreeBank.

We investigated different methods of optimizing Z in Sequence AdaBoost, namely using the loose and tight bounds, (7) and (10) respectively, and optimizing Z exactly when 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 observed a speed up on the convergence of the boosting algorithm when Z is optimized exactly. In order to evaluate this, we collected the features selected in the first 100 rounds and recorded the number of times each feature is selected. We found 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 renders (7) very loose. 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).

Overall, we observed that the performance of Sequence AdaBoost is comparable to CRFs. 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. The tighter bound leads to more accurate step sizes and therefore is preferable over the loose bound. 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.

5.3 Results of HM-SVMs

We extracted a corpus consisting of 300 sentences from the Penn TreeBank corpus for the Part-Of-Speech (POS) tagging experiments. We compared the performance of HMMs and CRFs (optimizing \mathbb{R}^{\log}) with HM-SVM as well as a HM-Perceptron, according to their test errors in 5-fold cross validation. We used second degree polynomial kernel for both the HM-Perceptron and the HM-SVM. C in SVM₁ is set to be 1. Although in a generative model like an HMM, overlapping features violate the model, we observed that HMMs using the overlapping features described above outperformed the ordinary HMMs. For this reason, we only report the results of HMMs with overlapping features. In Figure 2, *CRF* refers to optimizing \mathbb{R}^{\log} , *CRF-B* refers to optimizing regularized \mathbb{R}^{\log} as described in Section 2.3, *HM-PC* refers to HM-Perceptron.

Figure 2 summarizes the experimental results obtained on this task. The results demonstrate the competitiveness of HM-SVMs. As expected, CRFs perform better than the HM-Perceptron algorithm (*HM-PC*), since CRFs use the derivative of the log-loss function at every step, whereas the Perceptron algorithm uses only an approximation of it (cf. [6]). HM-SVMs achieve the best results, which validates our approach of explicitly maximizing a soft margin criterion.

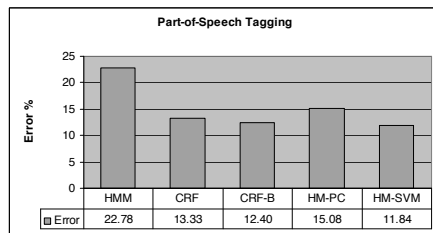


Fig. 2. Test error of POS task over a window of size 3 using 5-fold cross validation.

In order to examine the nature of the extracted support sequences, we investigated a subset of the support sequences \bar{y} . We observed that most of the support sequences only differ in a few positions from the correct label sequence, resulting in sparse solutions, on average 41 support sequences whereas the size of \mathcal{Y} is exponential in the length of sentences. It should also be noted that there are no support sequences for many of the training examples, i.e. $\alpha_i(\bar{y}_i) = 0$, since these examples already fulfill the margin constraints.

6 Discussion

In this paper, we formulated sequence labeling problem and outlined discriminative learning methods for sequence labeling with a special focus on large margin approaches.

Sequence Boosting, a generalization of Boosting to label sequence learning, 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 situations where efficiency during inference is crucial. This is of special important in speech recognition.

HM-SVMs, a generalization of SVMs to label sequence learning problem, inherit the the maximum-margin principle and the kernel-centric approach of SVMs. We presented an algorithm that makes use of the sparseness properties of the hinge loss and the structure of the parameters. Experimental results show that the algorithm outperforms CRFs in terms of accuracy and is computationally feasible.

We used discriminative sequence labeling methods to predict POS tagging. POS tagging is a particular class-based model used in the language model of speech recognizers. Training these models for other class definitions is of interest. An interesting direction to pursue is using these methods to train multiple class-based models jointly in order to improve the accuracy of each of these models and use them in combination for the language model of speech recognition.

References

1. Y. Altun, T. Hofmann, and M. Johnson. Discriminative learning for label sequences via boosting. In *NIPS*15*, 2003.
2. Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden markov support vector machines. In *International Conference on Machine Learning*, 2003.
3. P. F. Brown, V. J. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.
4. S. Chen and R. Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMUCS-99-108, Carnegie Mellon University, 1999.
5. M. Collins. Discriminative reranking for natural language parsing. In *Proc. 17th International Conf. on Machine Learning*, pages 175–182. Morgan Kaufmann, San Francisco, CA, 2000.
6. M. Collins. Discriminative training methods for hidden markov models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.
7. K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. In N. Cesa-Bianchi and S. Goldman, editors, *Proceedings of the Annual Conference on Computational Learning Theory*, pages 35–46, San Francisco, CA, 2000. Morgan Kaufmann Publishers.
8. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proceedings of the European Conference on Computational Learning Theory*, pages 23–37. Springer, 1995.

9. Yoav Freund. Self bounding learning algorithms. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 247–258, Madison, Wisconsin, 1998.
10. M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. Estimators for stochastic unification-based grammars. In *Proceedings ACL '99*, Maryland, 1999.
11. J. D. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic modeling for segmenting and labeling sequence data. In *ICML*, 2001.
12. T. Niesler and P. Woodland. Combination of word-based and category-based language models. In *Proc. ICSLP '96*, volume 1, pages 220–223, Philadelphia, PA, 1996.
13. T. Niesler and P. Woodland. A variable-length category-based n-gram language model. In *Proc. ICASSP '96*, pages 164–167, Atlanta, GA, 1996.
14. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, Cambridge, 1992. ISBN 0 - 521 - 43108 - 5.
15. Robert E. Schapire. Drifting games. In *Proc. 12th Annu. Conf. on Comput. Learning Theory*, pages 114–124. ACM Press, New York, NY, 1999.
16. Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
17. Robert E. Schapire and Yoram Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.
18. R. Schwarz and Y.L. Chow. The n-best algorithm: An efficient and exact procedure for finding the n most likely hypotheses. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 81–84, 1990.
19. B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Neural Information Processing Systems*, 2003.
20. I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 2005.