Investigating Loss Functions and Optimization Methods for Discriminative Learning of Label Sequences

Yasemin Altun
Computer Science
Brown University
Providence, RI 02912
altun@cs.brown.edu

Mark Johnson
Cognitive and Linguistic Sciences
Brown University
Providence, RI 02912
Mark_Johnson@brown.edu

Thomas Hofmann
Computer Science
Brown University
Providence, RI 02912
th@cs.brown.edu

Abstract

Discriminative models have been of interest in the NLP community in recent years. Previous research has shown that they are advantageous over generative models. In this paper, we investigate how different objective functions and optimization methods affect the performance of the classifiers in the discriminative learning framework. We focus on the sequence labelling problem, particularly POS tagging and NER tasks. Our experiments show that changing the objective function is not as effective as changing the features included in the model.

1 Introduction

Until recent years, generative models were the most common approach for many NLP tasks. Recently, there is a growing interest on discriminative models in the NLP community, and these models were shown to be successful for different tasks(Lafferty et al., 2001; Ratnaparkhi, 1999; Collins, 2000). Discriminative models do not only have theoretical advantages over generative models, as we discuss in Section 2, but they are also shown to be empirically favorable over generative models when features and objective functions are fixed (Klein and Manning, 2002).

In this paper, we use discriminative models to investigate the optimization of different objective functions by a variety of optimization methods. We focus on label sequence learning tasks. Part-of-Speech (POS) tagging and Named Entity Recognition (NER) are the most studied applications among these tasks. However, there are many others, such as chunking, pitch accent prediction and speech edit detection. These tasks differ in many aspects, such as the nature of the label sequences (chunks or individual labels), their difficulty and evaluation methods. Given this variety, we think it is worthwhile to investigate how optimizing different objective functions affects performance. In this paper, we varied the *scale* (exponential vs logarithmic) and the *manner* of the optimization (sequential vs pointwise) and using different combinations, we designed 4 different objective functions. We optimized these functions on NER and POS tagging tasks. Despite our intuitions, our experiments show that optimizing objective functions that vary in scale and manner do not affect accuracy much. Instead, the selection of the features has a larger impact.

The choice of the optimization method is important for many learning problems. We would like to use optimization methods that can handle a large number of features, converge fast and return sparse classifiers. The importance of the features, and therefore the importance of the ability to cope with a larger number of features is well-known. Since training discriminative models over large corpora can be expensive, an optimization method that converges fast might be advantageous over others. A sparse classifier has a shorter test time than a denser classifier. For applications in which the test time is crucial, optimization methods that result in sparser classifiers might be preferable over other methods

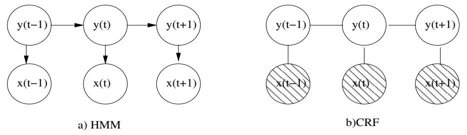

Figure 1: Graphical representation of HMMs and CRFs. Shaded areas indicate variables that the model conditions on.

even if their training time is longer. In this paper we investigate these aspects for different optimization methods, i.e. the number of features, training time and sparseness, as well as the accuracy. In some cases, an approximate optimization that is more efficient in one of these aspects might be preferable to the exact method, if they have similar accuracy. We experiment with exact versus approximate as well as parallel versus sequential optimization methods. For the exact methods, we use an off-the-shelf gradient based optimization routine. For the approximate methods, we use a perceptron and a boosting algorithm for sequence labelling which update the feature weights parallel and sequentially respectively.

2 Discriminative Modeling of Label Sequences Learning

Label sequence learning is, formally, the problem of learning a function that maps a sequence of observations $\mathbf{x} = (x_1, x_2, ..., x_T)$ to a label sequence $\mathbf{y} = (y_1, y_2, ..., y_T)$, where each $y_t \in \mathcal{Y}$, the set of individual labels. For example, in POS tagging, the words x_t's construct a sentence $\mathbf{x}$, and $\mathbf{y}$ is the labelling of the sentence where y_t is the part of speech tag of the word x_t. We are interested in the supervised learning setting, where we are given a corpus, $\mathcal{C} = (\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), ..., (\mathbf{x}^n, \mathbf{y}^n)$ in order to learn the classifier.

The most popular model for label sequence learning is the Hidden Markov Model (HMM). An HMM, as a generative model, is trained by finding the joint probability distribution over the observation and label sequences $p(\mathbf{x}, \mathbf{y})$ that explains the corpus $\mathcal{C}$ the best (Figure 1a). In this model, each random variable is assumed to be independent of the other random variables, given its parents. Because of the long distance dependencies of natural languages that cannot be modeled by sequences, this conditional independence assumption is violated in many NLP tasks. Another shortcoming of this model is that, due to its generative nature, overlapping features are difficult to use in HMMs. For this reason, HMMs have been standardly used with current word-current label, and previous label(s)-current label features. However, if we incorporate information about the neighboring words and/or information about more detailed characteristics of the current word directly to our model, rather than propagating it through the previous labels, we may hope to learn a better classifier.

Many different models, such as Maximum Entropy Markov Models (MEMMs) (McCallum et al., 2000), Projection based Markov Models (PMMs) (Punyakanok and Roth, 2000) and Conditional Random Fields (CRFs) (Lafferty et al., 2001), have been proposed to overcome these problems. The common property of these models is their discriminative approach. They model the probability distribution of the label sequences given the observation sequences: $p(\mathbf{y}|\mathbf{x})$.

The best performing models of label sequence learning are MEMMs or PMMs (also known as Maximum Entropy models) whose features are carefully designed for the specific tasks (Ratnaparkhi, 1999; Toutanova and Manning, 2000). However, maximum entropy models suffer from the so called *label bias* problem, the problem of making local decisions (Lafferty et al., 2001). Lafferty et al. (2001) show that CRFs overcome the label-bias problem and outperform MEMMs in POS tagging.

CRFs define a probability distribution over the whole sequence $\mathbf{y}$, globally conditioning over the whole observation sequence $\mathbf{x}$ (Figure 1b). Because they condition on the observation (as opposed to generating it), they can use overlapping features. The features $f(\mathbf{x}, \mathbf{y}, t)$ used in this paper are of the form:

1. Current label and information about the observation sequence, such as the identity or spelling features of a word that is within a window of the word currently labelled. Each of these features corresponds to a choice of y_t and x_s where $s \in \{t - w, ..., t, ..., t + w\}$ and w is the half window size

2. Current label and the neighbors of that label, i.e. features that capture the inter-label dependencies. Each of these features corresponds to

a choice of y_t and the neighbors of y_t, e.g. in a bigram model, $f(y_{t-1}, y_t)$.

The conditional probability distribution defined by this model is :

$$P_\theta(\mathbf{y}|\mathbf{x}) = \frac{\exp\left[\sum_t \sum_k \theta_k f_k(\mathbf{x}, \mathbf{y}, t)\right]}{Z_\theta(\mathbf{x})}$$

where θ_k's are the parameters to be estimated from the training corpus C and $Z_\theta(\mathbf{x})$ is a normalization term to assure a proper probability distribution. In order to simplify the notation, we introduce $F_k(\mathbf{x}, \mathbf{y}) = \sum_t f_k(\mathbf{x}, \mathbf{y}, t)$, which is the number of times feature f_k is observed in $(\mathbf{x}, \mathbf{y})$ pair and, $H_\theta(\mathbf{x}, \mathbf{y}) = \sum_k \theta_k F_k(\mathbf{x}, \mathbf{y})$, which is the linear combination of all the features with θ parameterization. F_k is the sufficient statistic of θ_k. Then, we can rewrite $P_\theta(\mathbf{y}|\mathbf{x})$ as: $\exp\left[H_\theta(\mathbf{x}, \mathbf{y})\right] / \sum_{\mathbf{y}} \exp\left[H_\theta(\mathbf{x}, \mathbf{y})\right]$.

3 Loss Functions for Label Sequences

Given the theoretical advantages of discriminative models over generative models and the empirical support by (Klein and Manning, 2002), and that CRFs are the state-of-the-art among discriminative models for label sequences, we chose CRFs as our model, and trained by optimizing various objective functions $\mathcal{F}_\theta(\mathcal{C})$ with respect to the corpus $\mathcal{C}$.

The application of these models to the label sequence problems vary widely. The individual labels might constitute chunks (e.g. Named-Entity Recognition, shallow parsing), or they may be single entries (e.g. POS tagging). The difficulty, therefore the accuracy of the tasks are very different from each other. The evaluation of the systems differ from one task to another, and the nature of the statistical noise level is task and corpus dependent. Given this variety, using objective functions tailored for each task might result in better classifiers. We consider two dimensions in designing objective functions: exponential versus logarithmic loss functions, and sequential versus pointwise optimization functions.

3.1 Exponential vs Logarithmic Loss functions

Most estimation procedures in NLP proceed by maximizing the likelihood of the training data. To overcome the numerical problems of working with a product of a large number of small probabilities, usually the logarithm of the likelihood of the data is optimized. However, most of the time, these systems, sequence labelling systems in particular, are tested with respect to their error rate on test data, i.e. the fraction of times the function H_θ assigns a higher score to a label sequence $\mathbf{y}$ (such that $\mathbf{y} \neq \mathbf{y}^i$) than the correct label sequence $\mathbf{y}^i$ for every observation $\mathbf{x}^i$ in test data. Then, the rank loss of H_θ might be a more natural objective to minimize.

$$\mathcal{F}'(\theta; \mathcal{C}) = \sum_i \sum_{\mathbf{y} \neq \mathbf{y}^i} \Theta(H_\theta(\mathbf{x}^i, \mathbf{y}) - H_\theta(\mathbf{x}^i, \mathbf{y}^i))$$

$\mathcal{F}'$is the total number of label sequences that H_θ ranks higher than the correct label sequences for the training instances in the corpus $\mathcal{C}$. Since optimizing the rank loss is NP-complete, one can optimize an upper bound instead, e.g. an exponential loss function:

$$\mathcal{F}''(\theta; \mathcal{C}) = \sum_i \sum_{\mathbf{y} \neq \mathbf{y}^i} \exp\left[H_\theta(\mathbf{x}^i, \mathbf{y}) - H_\theta(\mathbf{x}^i, \mathbf{y}^i)\right]$$

The exponential loss function is well studied in the Machine Learning domain. The advantage of the exp-loss over the log-loss is its property of penalizing incorrect labellings very severely, whereas it penalizes almost nothing when the label sequence is correct. This is a very desirable property for a classifier. Figure 2 shows this property of exp-loss in contrast to log-loss in a binary classification problem. However this property also means that, exp-loss has the disadvantage of being sensitive to noisy data, since systems optimizing exp-loss spends more

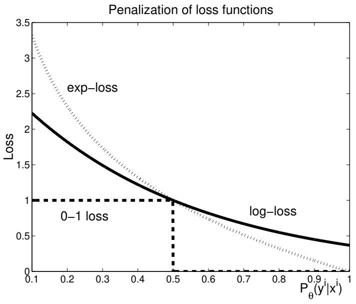

Figure 2: Loss values of 0-1, exp and log loss functions in a binary classification problem

effort on the outliers and tend to be vulnerable to noisy data, especially label noise.

3.2 Sequential vs Pointwise Loss functions

In many applications it is very difficult to get the whole label sequence correct since most of the time classifiers are not perfect and as the sequences get longer, the probability of predicting every label in the sequence correctly decreases exponentially. For this reason performance is usually measured pointwise, i.e. in terms of the number of individual labels that are correctly predicted. Most common optimization functions in the literature, however, treat the whole label sequence as one label, penalizing a label sequence that has one error and a label sequence that is all wrong in the same manner. We may be able to develop better classifiers by using a loss function more similar to the evaluation function. One possible way of accomplishing this may be minimizing pointwise loss functions. Sequential optimizations optimize the joint conditional probability distribution $P_\theta(\mathbf{y}|\mathbf{x})$, whereas pointwise optimizations that we propose optimize the marginal conditional probability distribution, $P_\theta(y_t|\mathbf{x}^i) = \sum_{\mathbf{y}:y_t=y_t^i} P_\theta(\mathbf{y}|\mathbf{x}^i)$.

3.3 Four Loss functions

We derive four loss functions by taking the cross product of the two dimensions discussed above:

- *Sequential Log-loss function:* This function, based on the standard maximum likelihood optimization, is used with CRFs in (Lafferty et al., 2001).

$$\begin{aligned} \mathcal{F}^1_\theta(\mathcal{C}) &= -\sum_i \log P_\theta(\mathbf{y}^i|\mathbf{x}^i) \\ &= -\sum_i H_\theta(\mathbf{x}^i,\mathbf{y}^i) + \log Z_\theta(\mathbf{x}^i) \end{aligned} \quad (1)$$

- *Sequential Exp-loss function:* This loss function, was first introduced in (Collins, 2000) for NLP tasks with a structured output domain. However, there, the sum is not over the whole possible label sequence set, but over the N best label sequences generated by an external mechanism. Here we include all possible label sequences; so we do not require an external mechanism to identify the best N sequences.. As shown in (Altun et al., 2002) it is possible to sum over all label sequences by using a dynamic algorithm.

$$\begin{aligned} \mathcal{F}^2_\theta(\mathcal{C}) &= \sum_{i,\mathbf{y}\neq\mathbf{y}^i} \exp\Big[H_\theta(\mathbf{x}^i,\mathbf{y}) - H_\theta(\mathbf{x}^i,\mathbf{y}^i)\Big] \\ &= \sum_i \Big[P_\theta(\mathbf{y}^i|\mathbf{x}^i)^{-1} - 1\Big] \end{aligned} \quad (2)$$

Note that the exponential loss function is just the inverse conditional probability plus a constant.

- *Pointwise Log-loss function:* This function optimizes the marginal probability of the labels at each position conditioning on the observation sequence:

$$\begin{aligned} \mathcal{F}^3_\theta(\mathcal{C}) &= -\sum_i \sum_t \log P_\theta(y_t^i|\mathbf{x}^i) \\ &= -\sum_{i,t} \log \sum_{\mathbf{y}:y_t=y_t^i} P_\theta(\mathbf{y}|\mathbf{x}^i) \end{aligned}$$

Obviously, this function reduces to the sequential log loss if the length of the sequence is 1.

- *Pointwise Exp-loss function:* Following the parallelism in log-loss vs exp-loss functions of sequential optimization (log vs inverse conditional probability), we propose minimizing the pointwise exp-loss function below, which reduces to the standard multi-class exponential loss when the length of the sequence is 1.

$$\begin{aligned} \mathcal{F}^4_\theta(\mathcal{C}) &= \sum_{i,t,\mathbf{y}} \exp\left[H_\theta(\mathbf{x}^i,\mathbf{y}) - \log \sum_{\mathbf{y}',y'_t=y_t^i} \exp\Big[H_\theta(\mathbf{x}^i,\mathbf{y}')\Big]\right] \\ &= \sum_{i,t} P_\theta(y_t^i|\mathbf{x}^i)^{-1} \end{aligned}$$

4 Comparison of the Four Loss Functions

We now compare the performance of the four loss functions described above. Although (Lafferty et al., 2001) proposes a modification of the iterative scaling algorithm for parameter estimation in sequential log-loss function optimization, gradient-based methods have often found to be more efficient for minimizing the convex loss function in Eq. (1) (Minka, 2001). For this reason, we use a gradient based method to optimize the above loss functions.

4.1 Gradient Based Optimization

The gradients of the four loss function can be computed as follows:

- *Sequential Log-loss function:*

$$\nabla_\theta \mathcal{F}^1 = \sum_i \mathbf{E}\left[F(\mathbf{x},\mathbf{y})|\mathbf{x}^i\right] - F(\mathbf{x}^i,\mathbf{y}^i) \quad (3)$$

  where expectations are taken w.r.t. $P_\theta(\mathbf{y}|\mathbf{x})$. Thus at the optimum the empirical and expected values of the sufficient statistics are equal. The loss function and the derivatives can be calculated with one pass of the forward-backward algorithm.

- *Sequential Exp-loss function:*

$$\nabla_\theta \mathcal{F}^2 = \sum_i \frac{\mathbf{E}\left[F(\mathbf{x},\mathbf{y})|\mathbf{x}^i\right] - F(\mathbf{x}^i,\mathbf{y}^i)}{P_\theta(\mathbf{y}^i|\mathbf{x}^i)} \quad (4)$$

  At the optimum the empirical values of the sufficient statistics equals their conditional expectations where the contribution of each instance is weighted by the inverse conditional probability of the instance. Thus this loss function focuses on the examples that have a lower conditional probability, which are usually the examples that the model labels incorrectly. The computational complexity is the same as the log-loss case.

- *Pointwise Log-loss function:*

$$\nabla_\theta \mathcal{F}^3 = \sum_{i,t} \mathbf{E}\left[F(\mathbf{x},\mathbf{y})|\mathbf{x}^i\right] - \mathbf{E}\left[F(\mathbf{x},\mathbf{y})|\mathbf{x}^i, y_t^i\right]$$

  At the optimum the expected value of the sufficient statistics conditioned on the observation $\mathbf{x}^i$ are equal to their expected value when also conditioned on the correct label sequence y_t^i. The computations can be done using the dynamic programming described in (Kakade et al., 2002), with the computational complexity of the forward-backward algorithm scaled by a constant.

- *Pointwise Exp-loss function:*

$$\nabla_\theta \mathcal{F}^4 = \sum_{i,t} \frac{\mathbf{E}\left[F(\mathbf{x},\mathbf{y})|\mathbf{x}^i\right] - \mathbf{E}\left[F(\mathbf{x},\mathbf{y})|\mathbf{x}^i, y_t^i\right]}{P_\theta(y_t^i|\mathbf{x}^i)}$$

  At the optimum the expected value of the sufficient statistics conditioned on x^i are equal to the value when also conditioned on y_t^i, where each point is weighted by $P_\theta(y_t^i|\mathbf{x}^i)^{-1}$. Computational complexity is the same as the log-loss case.

4.2 Experimental Setup

Before presenting the experimental results of the comparison of the four loss functions described above, we describe our experimental setup. We ran experiments on Part-of-Speech (POS) tagging and Named-Entity-Recognition (NER) tasks.

For POS tagging, we used the Penn TreeBank corpus. There are 47 individual labels in this corpus. Following the convention in POS tagging, we used a *Tag Dictionary* for frequent words. We used Sections 1-21 for training and Section 22 for testing.

For NER, we used a Spanish corpus which was provided for the Special Session of CoNLL2002 on NER. There are training and test data sets and the training data consists of about 7200 sentences. The individual label set in the corpus consists of 9 labels: the beginning and continuation of *Person, Organization, Location* and *Miscellaneous* names and nonname tags.

We used three different feature sets:

- $S1$ is the set of bigram features, i.e. the current tag and the current word, the current tag and previous tags.

- $S2$ consists of $S1$ features and spelling features of the current word (e.g. "Is the current word capitalized and the current tag is *Person-Beginning*?"). Some of the spelling features, which are mostly adapted from (Bikel et al., 1999) are the last one, two and three letters of the word; whether the first letter is lower case, upper case or alphanumeric; whether the word is capitalized and contains a dot; whether all the letters are capitalized; whether the word contains a hyphen.

- $S3$ includes $S2$ features not only for the current word but also for the words within a fixed window of size w. $S2$ is an instance of $S3$ where $w = 1$. An example of $S3$ features for $w \geq 3$ is "Does the previous word ends with a dot and the current tag is *Organization-Intermediate*?".

POS	$\mathcal{F}^1$	$\mathcal{F}^2$	$\mathcal{F}^3$	$\mathcal{F}^4$
$S1$	**94.91**	94.57	94.90	94.66
$S2$	95.68	95.25	**95.71**	95.31

Table 1: Accuracy of POS tagging on Penn TreeBank.

For NER, we used a window of size 3 (i.e. considered features for the previous and next words). Since the Penn TreeBank is very large, including $S3$ features, i.e. incorporating the information in the neighboring words directly to the model, is intractable. Therefore, we limited our experiments to $S1$ and $S2$ features for POS tagging.

4.3 Experimental Results

As a gradient based optimization method, we used an off-the-shelf optimization tool that uses the *limited-memory updating method*. We observed that this method is faster to converge than the conjugate gradient descent method. It is well known that optimizing log-loss functions may result in overfitting, especially with noisy data. For this reason, we used a regularization term in our cost functions. We experimented with different regularization terms. As expected, we observed that the regularization term increases the accuracy, especially when the training data is small; but we did not observe much difference when we used different regularization terms. The results we report are with the Gaussian prior regularization term described in (Johnson et al., 1999).

Our goal in this paper is not to build the best tagger or recognizer, but to compare different loss functions and optimization methods. Since we did not spend much effort on designing the most useful features, our results are slightly worse than, but comparable to the best performing models.

We extracted corpora of different sizes (ranging from 300 sentences to the complete corpus) and ran experiments optimizing the four loss functions using different feature sets. In Table 1 and Table 2, we report the accuracy of predicting every individual label. It can be seen that the test accuracy obtained by different loss functions lie within a relatively small range and the best performance depends on what kind of features are included in the model.

NER	$\mathcal{F}^1$	$\mathcal{F}^2$	$\mathcal{F}^3$	$\mathcal{F}^4$
$S1$	**59.92**	59.68	56.73	58.26
$S2$	**69.75**	67.30	68.28	69.51
$S3$	73.62	72.11	73.17	**73.82**

Table 2: F1 measure of NER on Spanish newswire corpus. The window size is 3 for $S3$.

We observed similar behavior when the training set is smaller. The accuracy is highest when more features are included to the model. From these results we conclude that when the model is the same, optimizing different loss functions does not have much effect on the accuracy, but increasing the variety of the features included in the model has more impact.

5 Optimization methods

In Section 4, we showed that optimizing different loss function does not have a large impact on the accuracy. In this section, we investigate different methods of optimization. The conjugate based method used in Section 4 is an exact method. If the training corpus is large, the training may take a long time, especially when the number of features are very large. In this method, the optimization is done in a parallel fashion by updating all of the parameters at the same time. Therefore, the resulting classifier uses all the features that are included in the model and lacks sparseness.

We now consider two approximation methods to optimize two of the loss functions described above. We first present a perceptron algorithm for labelling sequences. This algorithm performs parallel optimization and is an approximation of the sequential log-loss optimization. Then, we present a boosting algorithm for label sequence learning. This algorithm performs sequential optimization by updating one parameter at a time. It optimizes the sequential exp-loss function. We compare these methods with the exact method using the experimental setup presented in Section 4.2.

5.1 Perceptron Algorithm for Label Sequences

Calculating the gradients, i.e. the expectations of features for every instance in the training corpus can be computationally expensive if the corpus is very large. In many cases, a single training instance might be as informative as all of the corpus to update the parameters. Then, an online algorithm which makes updates by using one training example may converge much faster than a batch algorithm. If the distribution is peaked, one label is more likely than others and the contribution of this label dominates the expectation values. If we assume this is the case, i.e. we make a *Viterbi assumption*, we can calculate a good approximation of the gradients by considering only the most likely, i.e. the best label sequence according to the current model. The following online perceptron algorithm (Algorithm 1), presented in (Collins, 2002), uses these two approximations:

Algorithm 1 Label sequence Perceptron algorithm .

initialize $\forall k, \theta_k = 0$
repeat
  for all training patterns $\mathbf{x}^i$ **do**
    compute $\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} H_\theta(\mathbf{x}^i, \mathbf{y})$
    if $\mathbf{y}^i \neq \hat{\mathbf{y}}$ **then**
      $\theta_k \leftarrow \theta_k + F_k(\mathbf{x}^i, \mathbf{y}^i) - F_k(\mathbf{x}^i, \hat{\mathbf{y}})$
    end if
  end for
until stopping criteria

At each iteration, the perceptron algorithm calculates an approximation of the gradient of the sequential log-loss function (Eq. 3) based on the current training instance. The batch version of this algorithm is a closer approximation of the optimization of sequential log-loss, since the only approximation is the Viterbi assumption. The stopping criteria may be convergence, or a fixed number of iterations over the training data.

5.2 Boosting Algorithm for Label Sequences

The original boosting algorithm (AdaBoost), presented in (Schapire and Singer, 1999), is a sequential learning algorithm to induce classifiers for single random variables. (Altun et al., 2002) presents a boosting algorithm for learning classifiers to predict label sequences. This algorithm minimizes an upper bound on the sequential exp-loss function (Eq. 2). As in AdaBoost, a distribution over observations is defined:

$$D(i) \equiv \frac{P_\theta(\mathbf{Y}^i|\mathbf{X}^i)^{-1} - 1}{\sum_j \left[P_\theta(\mathbf{Y}^j|\mathbf{X}^j)^{-1} - 1\right]} \tag{5}$$

NER	$\mathcal{F}^1$	Perceptron	$\mathcal{F}^2$	Boosting
$S1$	59.92	59.77	59.68	48.23
$S2$	69.75	69.29	67.30	66.11
$S3$	73.62	72.97	72.11	71.07

Table 3: F1 of different methods for NER

This distribution which expresses the importance of every training instance is updated at each round, and the algorithm focuses on the more difficult examples. The sequence Boosting algorithm (Algorithm 2) optimizes an upper bound on the sequential exp-loss function by using the convexity of the exponential function. $u_k^{\max}$ is the maximum difference of the sufficient statistic F_k in any label sequence and the correct label sequence of any observation $\mathbf{x}^i$. $u_k^{\min}$ has a similar meaning. $u_k^d \equiv u_k^{\max} - u_k^{\min}$.

Algorithm 2 Label sequence Boosting algorithm.

1: initialize $D(i, \mathbf{y}) = \frac{1}{\sum_i |x^i|}, \forall i$
2: **repeat**
3:   **for** all features f_k **do**
4:     $r_k = \sum_i \frac{D(i)}{u_k^d}\left(u_k^{\max} + \frac{\mathbf{E}[F_k(\mathbf{x},\mathbf{y})|\mathbf{x}^i] - F_k(\mathbf{x}^i\mathbf{y}^i)}{P_\theta(\mathbf{y}^i|\mathbf{x}^i)-1}\right)$
5:     $\triangle\theta_k = \frac{1}{u_k^d}\log\left(\frac{-r_k u_k^{\min}}{(1-r_k)u_k^{\max}}\right)$
6:   **end for**
7:   $f = \arg\min_k \left[r_k e^{\triangle\theta_k u_k^{\min}} + (1-r_k)e^{\triangle\theta_k u_k^{\max}}\right]$
8:   $\theta_f \leftarrow \theta_f + \triangle\theta_f$
9:   Update $D(i)$
10: **until** stopping criteria

As it can be seen from Line 4 in Algorithm 2, the feature that was added to the ensemble at each round is determined by a function of the gradient of the sequential exp-loss function (Eq. 4). At each round, one pass of the forward backward algorithm over the training data is sufficient to calculate r_k's for all k. Considering the sparseness of the features in each training instance, one can restrict the forward backward pass only to the training instances that contain the feature that is added to the ensemble in the last round. The stopping criteria may be a fixed number of rounds, or by cross-validation on a heldout corpus.

6 Experimental Results

The results summarized in Table 3 compares the perceptron and the boosting algorithm with the gradient based method. Performance of the standard perceptron algorithm fluctuates a lot, whereas the average perceptron is more stable. We report the results of the average perceptron here. Not surprisingly, it does slightly worse than CRF, since it is an approximation of CRFs. The advantage of the Perceptron algorithm is its dual formulation. In the dual form, explicit feature mapping can be avoided by using the kernel trick and one can have a large number of features efficiently. As we have seen in the previous sections, the ability to incorporate more features has a big impact on the accuracy. Therefore, a dual perceptron algorithm may have a large advantage over other methods.

When only HMM features are used, Boosting as a sequential algorithm performs worse than the gradient based method that optimizes in a parallel fashion. This is because there is not much information in the HMM features other than the identity of the word to be labeled. Therefore, the boosting algorithm needs to include almost all the features one by one in the ensemble. When there are just a few more informative features, the boosting algorithm makes better use of them. This situation is more dramatic in POS tagging. Boosting gets 89.42% and 94.92% accuracy for $S1$ and $S2$ features, whereas the gradient based method gets 94.57% and 95.25%. The gradient based method uses all of the available features, whereas boosting uses only about 10% of the features. Due to the loose upper bound that Boosting optimizes, the estimate of the updates are very conservative. Therefore, the same features are selected many times. This negatively effects the convergence time, and the other methods outperform Boosting in terms of training time.

7 Conclusion and Future Work

In this paper, we investigated how different objective functions and optimization methods affect the accuracy of the sequence labelling task in the discriminative learning framework. Our experiments show that optimizing different objective functions does not have a large affect on the accuracy. Extending the feature space is more effective. We conclude that methods that can use large, possibly infinite number of features may be advantageous over others. We are running experiments where we use a dual formulation of the perceptron algorithm which has the property of being able to use infinitely many features. Our future work includes using SVMs for label sequence learning task.

References

Y. Altun, T. Hofmann, and M. Johnson. 2002. Discriminative learning for label sequences via boosting. In *Proceedings of NIPS*15*.

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

M. Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of ICML 2002*.

M. Collins. 2002. Ranking algorithms for named-entity extraction: Boosting and the voted perceptron. In *Proceedings of ACL'02*.

M. Johnson, S. Geman, S. Canon, Z. Chi, and S. Riezler. 1999. Estimators for stochastic unification-based grammars. In *Proceedings of ACL'99*.

S. Kakade, Y.W. Teh, and S. Roweis. 2002. An alternative objective function for Markovian fields. In *Proceedings of ICML 2002*.

Dan Klein and Christopher D. Manning. 2002. Conditional structure versus conditional estimation in nlp models. In *Proceedings of EMNLP 2002*.

J. Lafferty, A. McCallum, and F. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML2001*.

A. McCallum, D. Freitag, and F. Pereira. 2000. Maximum Entropy Markov Models for Information Extraction and Segmentation. In *Proceedings of ICML 2000*.

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

V. Punyakanok and D. Roth. 2000. The use of classifiers in sequential inference. In *Proceedings of NIPS*13*.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175.

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

Kristina Toutanova and Christopher Manning. 2000. Enriching the knowledge sources used in a maximum entropy pos tagger. In *Proceedings of EMNLP 2000*.