Flexible Margin Selection for Reranking with Full Pairwise Samples

Libin Shen and **Aravind K. Joshi**
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{libin,joshi}@linc.cis.upenn.edu

Abstract

Perceptron like large margin algorithms are introduced for the experiments with various margin selections. Compared to the previous perceptron reranking algorithms, the new algorithms use full pairwise samples and allow us to search for margins in a larger space. Our experimental results on the data set of (Collins, 2000) show that a perceptron like ordinal regression algorithm with uneven margin can achieve Recall/Precision of 89.5/90.0 on section 23 of WSJ PTB. Our result on margin selection can be employed in other large margin machine learning algorithms as well as in other NLP tasks.

1 Introduction

In recent years, the so-called *reranking* techniques (Collins, 2000) have been successfully used in parameter estimation in many applications which were previously modeled as generative models. A baseline generative model generates N-best candidates, and then these candidates are reranked by using a rich set of local and global features. Various machine learning algorithms have been adapted to the reranking tasks.

In the field of machine learning, a class of tasks, which are called *ranking* or *ordinal regression*, are similar to the reranking tasks in NLP. A primary motivation of the present paper is to apply ranking or ordinal regression algorithms to the reranking tasks in NLP, especially because we observe that there is no direct way to apply these ranking algorithms to reranking. More specifically, we will compare the existing reranking and ranking algorithms in the framework of *margin selection.* The goal then is to look for a desirable margin for the reranking tasks in NLP.

In order to experiment with various margins, we will introduce variants of the traditional perceptron algorithm (Rosenblatt, 1958; Novikoff, 1962) for reranking, which allows the use of various margins; The training is also very fast. The basic idea of these perceptron like algorithms is that we dynamically search for pairs of inconsistent objects and use them to update the weight vector. Since the ranks are ordered, the dynamical search can be implemented efficiently. These algorithms will be justified by modifying the proof for the perceptron training in (Krauth and Mezard, 1987).

Compared to previous work on perceptron for parse reranking (Collins and Duffy, 2002), our new algorithms use full pairwise samples instead of partial pairwise samples. This allows us to search for margins desirable for reranking tasks in a larger space, which is unavailable in the previous work.

In this paper, we focus on the parse reranking task. However, the methods can, of course, be applied to other NLP reranking tasks. Our experimental results on the data set of (Collins, 2000) show that a perceptron like ordinal regression algorithm with uneven margin can achieve Recall/Precision of 89.5/90.0 on section 23 of WSJ PTB, which is comparable to 89.6/89.9 with the boosting algorithm in Collins' paper, although boosting is believed to have more generalization capability. Our results also show that the new margins introduced in this paper are superior to the margins used in the previous works on reranking. The results on margin selection can be employed in reranking systems based on other machine learning algorithms, such as Winnow, Boosting and SVMs, as well as other NLP tasks, e.g. machine translation reranking.

The paper is organized as follows. In section 2, we summarize the previous works on ranking and reranking, and investigate these works in the context of ranks and margins. Then we propose a desirable margin selection for reranking in section 3. In section 4 we propose two new perceptron based algorithms. The new algorithms are applied to the parse reranking problem in section 5. Finally, we will describe some new experiments related to the parse reranking task.

2 Previous Works

2.1 Reranking

In recent years, reranking has been successfully applied to some NLP problems, especially to the problem of parse reranking. Ratnaparkhi (1997) noticed that by ranking the 20-best parsing results generated by his maximal entropy parser, the F-measure went to 93% from 87%, if the oracle parse was successfully detected. Charniak (2000) reranked the N-best parses by reestimating a language model on a large number of features.

Collins (2000) first used machine learning algorithms for parse reranking. Two approaches were proposed in that paper; one used Boosting Loss and the other used Log-Likelihood Loss. Boosting Loss achieved better results. The Boosting Loss model is as follows. Let $\mathbf{x}_{i,j}$ be the feature vector of the j^{th} parse of the i^{th} sentence. Let $\tilde{\mathbf{x}}_i$ be the feature vector of the best parse for the i^{th} sentence. Let F_α be a score function

$$F_\alpha(\mathbf{x}_{i,j}) \equiv \alpha' \cdot \mathbf{x}_{i,j},$$

where α is a weight vector. The *margin* $M_{\alpha,i,j}$ on sample $\mathbf{x}_{i,j}$ is defined as

$$M_{\alpha,i,j} \equiv F_\alpha(\tilde{\mathbf{x}}_i) - F_\alpha(\mathbf{x}_{i,j})$$

Finally the Boost Loss function is defined as

$$\begin{aligned} BoostLoss(\alpha) &\equiv \sum_i \sum_j e^{F_\alpha(\tilde{\mathbf{x}}_i) - F_\alpha(\mathbf{x}_{i,j})} \\ &= \sum_i \sum_j e^{-M_{\alpha,i,j}} \end{aligned}$$

The Boosting algorithm was used to search the weight vector α to minimize the Boost Loss.

We may rewrite the definition of the margin $M_{\alpha,i,j}$ by using pairwise samples as follows.

$$\mathbf{s}_{i,j} \equiv \tilde{\mathbf{x}}_i - \mathbf{x}_{i,j} \text{ , then}$$

$$\begin{aligned} M_{\alpha,i,j} = F_\alpha(\tilde{\mathbf{x}}_i) - F_\alpha(\mathbf{x}_{i,j}) &= F_\alpha(\tilde{\mathbf{x}}_i - \mathbf{x}_{i,j}) \\ &= F_\alpha(\mathbf{s}_{i,j}) \end{aligned}$$

So the Boosting Loss approach in (Collins, 2000) is similar to maximizing the margin (Schapire et al., 1997) between 0 and $F_\alpha(\mathbf{s}_{i,j})$, where $\mathbf{s}_{i,j}$ are pairwise samples as we have described above.

In (Collins and Duffy, 2002), the voted perceptron and the Tree kernel were applied to parse reranking. Similar to (Collins, 2000), pairwise samples were used as training samples. The perceptron updating step was defined as

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \tilde{\mathbf{x}}_i - \mathbf{x}_{i,j},$$

where $\mathbf{w}^t$ is the weight vector at the t th updating. This is equivalent to using pairwise sample $\mathbf{s}_{i,j}$ as we have defined above.

$$\mathbf{w}^{t+1} = \mathbf{w}^t + \mathbf{s}_{i,j}$$

Shen and Joshi (2003) applied Support Vector Machines (SVMs) and Tree kernels to parse reranking. In that paper, pairwise samples were used explicitly through the Preference kernel. $\mathbf{u}^+_{i,j}$ and $\mathbf{u}^-_{i,j}$ defined as follows were used as positive samples and negative samples respectively.

$$\mathbf{u}^+_{i,j} \equiv (\tilde{\mathbf{x}}_i, \mathbf{x}_{i,j}), \;\; \mathbf{u}^-_{i,j} \equiv (\mathbf{x}_{i,j}, \tilde{\mathbf{x}}_i)$$

SVM is used to maximize the margin between positive samples and negative samples, which in turn is proportional to the margin between the best parse of each sentence and the rest of the N-best parses.

In the works on reranking, the margin is defined as the distance between the best candidate and the rest. The reranking problem is reduced to a classification problem by using pairwise samples implicitly or explicitly.

2.2 Ranking

In the previous works on ranking or ordinal regression, the margin is defined as the distance between two consecutive ranks. Two large margin approaches have been used. One is to extend the perceptron algorithm by using multiple biases to represent the boundaries between every two consecutive ranks (Crammer and Singer, 2001). The other approach is to reduce the ranking problem to a classification problem by using the trick of pairwise samples (Herbrich et al., 2000).

Crammer and Singer (2001) proposed the *PRank* algorithm, a perceptron based ranking algorithm. In their framework each instance is associated with a rank which is an integer from 1 to k. The goal of their ranking algorithm is to predict the correct rank of each instance. The PRank algorithm is a variant of the perceptron algorithm. The difference is that the PRank algorithm maintains a set of biases which are used as boundaries between two neighboring ranks.

PRank works very well for the ranking problems in which each sample is associated with a integer rank. However, due to the introduction of a set of biases it is not possible to use them in other ranking-like problems. For example, the PRank algorithm cannot be trained on the data associated with a partial order instead of total order on ranks. Furthermore, as we will show in section 3, the PRank algorithm cannot handle the reranking problems.

Herbrich et al. (2000) proposed a margin based approach for ranking, or *ordinal regression* as they called in their paper. In their framework, each training sample is associated with a rank which is an integer. The target function is required to maximize the margins between the samples of neighboring ranks. The Support Vector Machines (SVMs) (Vapnik, 1998) were used to compute the unique function maximizing the margins. In contrast to PRank, rank boundaries were not used explicitly in the training. Their approach is implemented by using pairwise samples in training. For example, there are two samples $\mathbf{u}$, $\mathbf{v}$, where $\mathbf{u}$ ranks i and $\mathbf{v}$ ranks $i+j$, then $\mathbf{u}-\mathbf{v}$ is used as a positive sample and $\mathbf{v} - \mathbf{u}$ is used a negative sample. The *Preference Kernel* was used for the kernel trick to be applied on the input samples.

The underlying assumption of the ordinal regression is that samples between consecutive ranks are separable. This may become a problem in the case that the ranks are unreliable when ranking is too fine. On the other hand, the size of generated training samples will be very large. Suppose there are n samples. The total number of pairwise samples in (Herbrich et al., 2000) is roughly n^2.

2.3 Large Margin Classifiers

There are quite a few linear classifiers[1] that can separate samples with large margin, such as SVMs (Vapnik, 1998), Boosting (Schapire et al., 1997), Winnow (Zhang, 2000) and Perceptron (Krauth and Mezard, 1987). The performance of SVMs is superior to other linear classifiers because of their ability to maximize the margin.

However, SVMs are extremely slow in training since they need to solve quadratic programming problems for optimization. Often speed is achieved by dividing the data into sections (e.g. in (Shen and Joshi, 2003)). However, the SVMs' capability of global margin optimization is compromised.

For margin selection, we do need an algorithm that runs fast for training, so that we can test various margins. Then the result of the margin selection can be employed in other linear classifiers. For the purpose of margin selection we proposes perceptron like algorithms for the following two reasons. First, perceptron is fast in training which allows us to do experiments with various margin selections on real-world data. Furthermore, perceptron algorithms are simple in principle, which makes it easy to implement modification.

3 Ranks and Margins for Reranking

In the previous works on ranking, ranks are defined on the whole training and test data. Thus we can define boundaries between consecutive ranks on the whole data. In the reranking problem, ranks are *local*. They are defined over a sub set of the samples in the data set. For example, in the parse reranking problem, the rank of a parse is only defined as the rank among all the parses for the same sentence. The training data includes 36,000 sentence, with an average of about 27 parses per sentence (Collins, 2000).

As a result, we cannot use the PRank algorithm in the reranking task, since there are no *global*

[1] Here we do not consider kernels of infinite dimension

ranks or boundaries in reranking, as the PRank algorithm is designed to estimate the global rank boundaries over all the samples during the training. If we introduce auxiliary variables for the boundaries for each cluster, the number of the parameters will be as large as the number of samples. Obviously this is not a good idea. However, the approach of using pairwise samples works. By pairing up two samples, we actually compute the relative distance between these two samples in the scoring metric.

Let $\mathbf{r}_i$ be the candidate parse that ranks as the i^{th} best for a sentence. The parses of the same sentence are ranked with respect to their *f-scores*, which measure the similarity to the gold standard parse. A parse with a large *f-score* is assigned a high rank. In reranking tasks, the margins between the best candidate and the rest are more useful. A hyperplane successfully separating $\mathbf{r}_1$ and $\mathbf{r}_2 ... \mathbf{r}_N$ is more predictive than a hyperplane successfully separating $\mathbf{r}_1 ... \mathbf{r}_{10}$ and $\mathbf{r}_{11} ... \mathbf{r}_N$, if we are only interested in the topmost result in test. This is also how the existing reranking systems are designed. However there are some problems with this approach.

There is a practical problem for the definition of the *best* parse in a sentence. In parse reranking, we may find several best parses for each training sentence instead of one. In order to break the tie, usually one selects just one of them arbitrarily as the top ranked parse and discard all others.

Furthermore, if we only look for the hyperplane to separate the best one from the rest, we, in fact, discard the order information of $\mathbf{r}_2 ... \mathbf{r}_N$. For example, we did not employ the information that $\mathbf{r}_{10}$ is better than $\mathbf{r}_{11}$ in the training. Knowing $\mathbf{r}_{10}$ is better than $\mathbf{r}_{11}$ may be useless for training to some extent, but knowing $\mathbf{r}_2$ is better than $\mathbf{r}_{11}$ is useful.

On other the hand, the resulting weight vector $\mathbf{w}$ is supposed to assign the highest score to $\mathbf{r}_1$. Should it not assign the second highest score to $\mathbf{r}_2$? Although we cannot give an affirmative answer at this time, it is at least reasonable to use more pairwise samples. This approach was avoided in the previous works on reranking, due to the problem of complexity of both the data size and the execution time. Thus we have provided a strong motivation for investigating some new reranking algorithms such that

- They utilize all the ordinal relations encoded in the ranked lists.
- The size of training data remains the same as the original size of the ranked lists.
- The training time increases only moderately, although more information is used in training.

4 Perceptron for Ordinal Regression

4.1 Ordinal Regression

Let $\mathbf{x}_{i,j}, \mathbf{x}_{i,l}$ be the feature vectors of two parses for sentence i and $y_{i,j}, y_{i,l}$ be their ranks respectively, where $y_{i,j} + \epsilon < y_{i,l}$, and ϵ is a non-negative real number. It means that the rank of $\mathbf{x}_{i,j}$ of ϵ higher than the rank of $\mathbf{x}_{i,l}$. In this case, we say $\mathbf{x}_{i,j}$ is *significantly better* than $\mathbf{x}_{i,l}$. We are interested in finding a weight vector $\mathbf{w}$, such that

$$\mathbf{w} \cdot \mathbf{x}_{i,j} > \mathbf{w} \cdot \mathbf{x}_{i,l} + \tau, \text{ if } y_{i,j} + \epsilon < y_{i,l}$$

We ignore any pair of parses in which the difference in the ranks is $\leq \epsilon$. Hence, this problem is called *ϵ-insensitive ordinal regression.*

Let the training samples be

$$S = \{(\mathbf{x}_{i,j}, y_{i,j}) \mid 1 \leq i \leq m, \ 1 \leq j \leq k\},$$

where m is the number of sentences and k is the size of the N-best list. Let $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$. We say the training data is *ϵ-distinguishable* by f if

$$\mathbf{w} \cdot \mathbf{x}_{i,j} > \mathbf{w} \cdot \mathbf{x}_{i,l}, \text{ if } y_{i,j} + \epsilon < y_{i,l},$$

for $1 \leq i \leq m, \ 1 \leq j, l \leq k$.

Suppose the training data is *ϵ-distinguishable* by f, the *ordinal margin* for parses for sentence i is defined as follows

$$\gamma(f, \epsilon, i) = \min_{j,l: y_{i,j} + \epsilon < y_{i,l}} f(\mathbf{x}_{i,j}) - f(\mathbf{x}_{i,l})$$

The *minimal ordinal margin*, γ^{order}, for f and ϵ is defined as follows.

$$\begin{aligned} \gamma^{order}(f, \epsilon) &= \min_i \gamma(f, \epsilon, i) \\ &= \min_{y_{i,j} + \epsilon < y_{i,l}} f(\mathbf{x}_{i,j}) - f(\mathbf{x}_{i,l}) \end{aligned}$$

If $\epsilon = 0$, then it is almost a regression problem, since it requires all the parses to keep the original order under function f.

4.2 Dynamic Pairing

A straightforward method of using pairwise samples is to define positive and negative samples on the differences of vectors as in (Herbrich et al., 2000). For each sentence i, $\mathbf{x}_{i,j} - \mathbf{x}_{i,l}$ is a positive sample if $y_{i,j} < y_{i,l}$, where $y_{i,j}$ is the rank of parse $\mathbf{x}_{i,j}$. Similarly, $\mathbf{x}_{i,j} - \mathbf{x}_{i,l}$ is a negative sample if $y_{i,l} < y_{i,j}$.

However, for real tasks, this greatly increases the data complexity from $O(mk)$ to $O(mk^2)$, where m is the number of training sentences, and k is the size of n-best list. For parse reranking k is about 27, and for machine translation reranking k is about 1000. Due to the limit of memory space we cannot define pairwise samples explicitly in this way.

The method to avoid this problem is to look up pairwise samples dynamically, as shown in **Algorithm 1**, a perceptron like algorithm. The basic idea is that, for each pair of parses for the same sentence, if

- the rank of $\mathbf{x}_{i,j}$ is significantly higher than the rank of $\mathbf{x}_{i,l}$, $y_{i,j} + \epsilon < y_{i,l}$
- the weight vector $\mathbf{w}$ can not successfully separate ($\mathbf{x}_{i,j}$ and $\mathbf{x}_{i,l}$) with a learning margin τ, $\mathbf{w} \cdot \mathbf{x}_{i,j} < \mathbf{w} \cdot \mathbf{x}_{i,l} + \tau$,

then we need to update $\mathbf{w}$ with the addition of $\mathbf{x}_{i,j} - \mathbf{x}_{i,l}$. It is not difficult to show Algorithm 1 is equivalent to using pairwise samples in training.

4.3 Sentence-Level Updating

In Algorithm 1, for each **repeat** iteration, the complexity is $O(mk^2d)$, where m and k are defined as above, and d is the average number of active features in a sample. We notice that the score of a parse $\mathbf{x}_{i,j}$ will be computed for k times in each **repeat** iteration. However, in many cases this is not necessary. In this section, we will revise Algorithm 1 to speed up the training phase.

Algorithm 2 is similar to Algorithm 1 except that the updating is not executed until all the inconsistent pairs for the same sentence are found. Therefore we only need to compute $\mathbf{w} \cdot \mathbf{x}_{i,j}$ for only once in each **repeat** iteration. So the complexity of each **repeat** iteration is $O(mk^2 + mkd)$.

The following theorem will show that Algorithm 2 will stop in finite number of steps, outputting a function that *ϵ-distinguishes* the training data, if the training data is ϵ-distinguishable.

Algorithm 1 ordinal regression

```
Require: a positive learning margin $\tau$.
$t \leftarrow 0$, initialize $\mathbf{w}^0$;
repeat
    for (sentence $i = 1, ..., m$) do
        for ($1 \leq j < l \leq k$) do
            if ($y_{i,l} - y_{i,j} > \epsilon$ and $\mathbf{w}^t \cdot \mathbf{x}_{i,j} < \mathbf{w}^t \cdot \mathbf{x}_{i,l} + \tau$) then
                $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \mathbf{x}_{i,j} - \mathbf{x}_{i,l}$; $t \leftarrow t+1$;
            else if ($y_{i,j} - y_{i,l} > \epsilon$ and $\mathbf{w}^t \cdot \mathbf{x}_{i,l} < \mathbf{w}^t \cdot \mathbf{x}_{i,j} + \tau$) then
                $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \mathbf{x}_{i,l} - \mathbf{x}_{i,j}$; $t \leftarrow t+1$;
            end if
        end for
    end for
until no updates made in the outer for loop
```

Theorem 1 *Suppose the training samples $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ are ϵ-distinguishable by a linear function defined on the weight vector $\mathbf{w}^*$ with a splitting margin γ, where $||\mathbf{w}^*|| = 1$. Let $R = max_{i,j}||\mathbf{x}_{i,j}||$. We have*

a *Algorithm 2 makes at most $\frac{2k^2R^2+2\tau}{\gamma^2}$ mistakes on the pairwise samples during the training.*

b *Algorithm 2 stops in T steps of updates, where*

$$T \leq \frac{2k^2R^2 + 2\tau}{\gamma^2} \qquad (1)$$

Due to the space limitation, you omit the proof.

4.4 Uneven Margin

For ϵ-insensitive ordinal regression, suppose $\epsilon = 10$ and our ordinal regression algorithm made two errors. One is on $(\mathbf{r}_1, \mathbf{r}_{11})$, and the other is on $(\mathbf{r}_{21}, \mathbf{r}_{31})$. The algorithm cannot recognize that the former is more serious than the latter. On the other hand, the algorithm does not try to distinguish $\mathbf{r}_1$ and $\mathbf{r}_{10}$, which is even worse.

Our solution is to apply *uneven margin* to the ϵ-insensitive ordinal regression. For example, we want to find a hyperplane for each sentence such that there is larger margin between $\mathbf{r}_1$ and $\mathbf{r}_{10}$, but

Algorithm 2 ordinal regression, sentence updating

```
Require: a positive learning margin $\tau$.
 1: $t \leftarrow 0$, initialize $\mathbf{w}^0$;
 2: repeat
 3:   for (sentence $i = 1, ..., m$) do
 4:     compute $\mathbf{w}^t \cdot \mathbf{x}_{i,j}$ and $u_j \leftarrow 0$ for all $j$;
 5:     for ($1 \leq j < l \leq k$) do
 6:       if ($y_{i,l} - y_{i,j} > \epsilon$ and $\mathbf{w}^t \cdot \mathbf{x}_{i,j} < \mathbf{w}^t \cdot \mathbf{x}_{i,l} + \tau$) then
 7:         $u_j \leftarrow u_j + 1$; $u_l \leftarrow u_l - 1$;
 8:       else if ($y_{i,j} - y_{i,l} > \epsilon$ and $\mathbf{w}^t \cdot \mathbf{x}_{i,l} < \mathbf{w}^t \cdot \mathbf{x}_{i,j} + \tau$) then
 9:         $u_j \leftarrow u_j - 1$; $u_l \leftarrow u_l + 1$;
10:       end if
11:     end for
12:     $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \sum_j u_j \mathbf{x}_{i,j}$; $t \leftarrow t + 1$;
13:   end for
14: until no updates made in the outer for loop
```

a smaller margin between $\mathbf{r}_1$ and $\mathbf{r}_2$, where $\mathbf{r}_j$ is the parse that ranks j for a sentence. Similarly, we want a larger margin between $\mathbf{r}_1$ and $\mathbf{r}_2$, but a smaller margin between $\mathbf{r}_{10}$ and $\mathbf{r}_{11}$. Thus

$$\begin{aligned} margin(\mathbf{r}_1, \mathbf{r}_{10}) &> margin(\mathbf{r}_1, \mathbf{r}_2) \\ &> margin(\mathbf{r}_{10}, \mathbf{r}_{11}) \end{aligned} \quad (2)$$

So the solution is to search for a hyperplane such that

$$score(\mathbf{r}_p) - score(\mathbf{r}_q) > g(p,q)\tau$$

where $g(1,10) > g(1,2) > g(10,11)$. Specifically, we replace one of the updating conditions

$$\mathbf{w} \cdot \mathbf{x}_{i,j} < \mathbf{w} \cdot \mathbf{x}_{i,l} + \tau$$

in line 6 of Algorithm 2 with

$$\frac{\mathbf{w} \cdot \mathbf{x}_{i,j} - \mathbf{w} \cdot \mathbf{x}_{i,l}}{g(y_{i,j}, y_{i,l})} < \tau, \quad (3)$$

and replace the updating condition

$$y_{i,l} - y_{i,j} > \epsilon \text{ with } g(y_{i,j}, y_{i,l}) > \epsilon, \quad (4)$$

which means that we ignore irrelevant inconsistent pairs with respect to g. We also replace the updating operation in line 7 with

$$u_j \leftarrow u_j + g(y_{i,j}, y_{i,l}),\ u_l \leftarrow u_l - g(y_{i,j}, y_{i,l}) \quad (5)$$

A similar modification is made in line 8 and 9.

It can be shown that modifying Algorithm 2 in this way is equivalent to using $(\mathbf{x}_{i,j} - \mathbf{x}_{i,l})/g(y_{i,j}, y_{i,l})$ as pairwise samples, so it is well defined. Due to the space limitation, we omit the proof of the equivalence in this paper.

There are many candidates for the function g. The following function is one of the simplest solutions.

$$g(p,q) \equiv \frac{1}{p} - \frac{1}{q}$$

We will use this function in our experiments on parse reranking.

4.5 Margin Based Generalization Bounds

So far we have proposed two perceptron based large margin algorithms for ordinal regression. We need to show the relation between the expected error rate and the *ordinal margin* that we have defined above. We give a sketch of the proof.

Suppose the mk parses for the m sentences are i.i.d. However, the pairwise samples are not independent. In this way, there is no straightforward application of the results from learning theory on all the pairwise samples.

However, we can use the same technique used in (Herbrich et al., 2000). The idea is to generate only $k - 1$ i.i.d. pairwise samples for each sentence. We can get upper bounds on classification risk with these $m(k - 1)$ pairwise samples with margin based bounds. Then we can relate the ordinal regression risk to the classification risk.

5 Experimental Results

In this section, we will report the experimental results for parse reranking task. We use the same data set as described in (Collins, 2000). Section 2-21 of the WSJ Penn Treebank (PTB)(Marcus et al., 1994) are used as training data, and section 23 is used for test. The training data contains around 40,000 sentences, each of which has 27 distinct parses on average. Of the 40,000 training sentences, the first 36,000 are used to train perceptrons. The remaining 4,000 sentences are used as development data for parameter estimation, such as the number of rounds of iteration in training. The 36,000 training sentences contain 1,065,620 parses totally. We use the feature set generated by Collins (Collins, 2000).

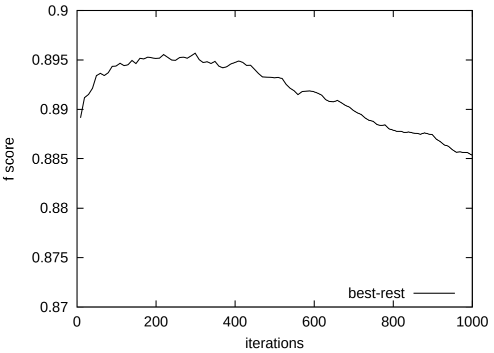

Figure 1: f scores on section 23 of PTB by separating the best parse from the rest in training

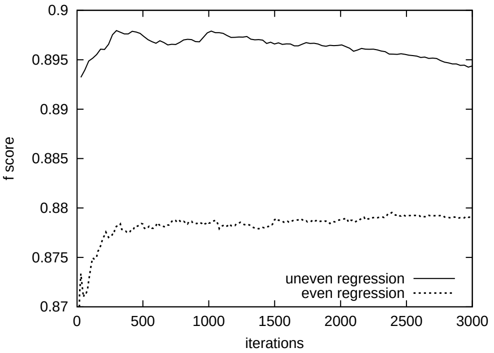

Figure 2: f scores on section 23 of PTB with ordinal regression algorithms

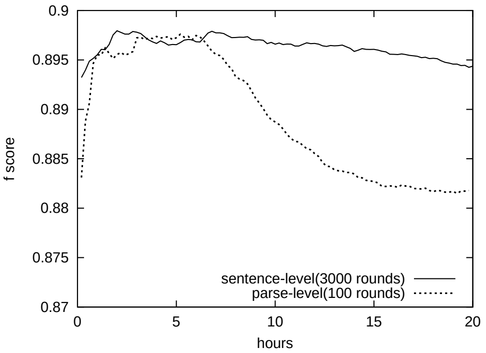

Figure 3: Convergence of cluster-level updating and item-level updating

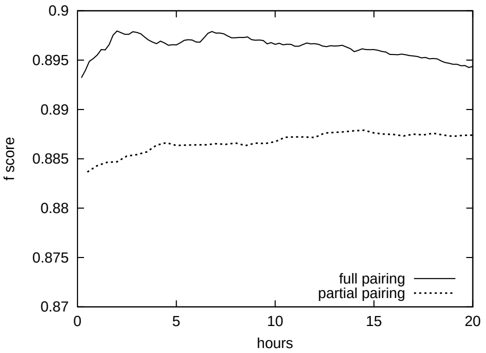

Figure 4: Using all pairwise samples vs. using partial pairwise samples

In all of our experiments, we have employed the voted perceptron as in (Freund and Schapire, 1999; Collins and Duffy, 2002). The voted version makes the result on the test set more stable.

In the first set of experiments, Algorithm 2 and its uneven margin variants are used. In addition, we evaluate the performance of separating only the best parse from the rest in training by modifying the updating condition in Algorithm 2. Figure 1 and 2 show the learning curves of different models on the test data, section 23 of Penn Treebank. Ordinal regression with uneven margin shows great advantage over the same algorithm using even margin. Its performance is also better than perceptron that is only trained to separate the best parse from the rest.

By estimating the number of rounds of iterations on the development data, we get the results for the test data as shown in Table 1. Our ordinal regression algorithm with uneven margin achieves the best result in f-score. It verifies that using more pairs in training is helpful for the reranking problem. In addition, the uneven margin is crucial to the parse reranking task.

In Algorithm 2, we update the weight vector on the sentence level so as to speed up the training, while in Algorithm 1 we update the weight vector for each pair of parses. Figure 3 shows the comparison of the learning curves of the ordinal regression using parse level updating and the ordinal regression using sentence level updating. Algorithm 2 converges about 40% faster. The performance of Algorithm 2 is very good even within the first few rounds of iterations. Furthermore, the f-score on the test data remains at a high level although it is over-trained. Algorithm 1 easily leads to overfitting for the training data, while Algorithm 2 does not suffer from overfitting. This can be explained by an analog to the gradient methods. For Algorithm 1, we move in one direction at a time, so the result depends on the order of parses of a sentences, and it is easy to jump into a suboptimum. For Algorithm 2, we move in multiple-directions at a time, so the result is more stable.

section 23, $\leq$100 words (2416 sentences)			
model	recall%	prec%	f-score%
baseline	88.1	88.3	88.2
best-rest	89.2	89.8	89.5
ordinal	88.1	87.8	88.0
uneven ordinal	89.5	90.0	89.8

Table 1: Experimental Results

Our last set of experiments are about using all and partial pairwise samples. In order to theoretically justify Algorithm 2, we only use $k - 1$ pairwise parses for each sentence, e.g. pairs of parses with consecutive ranks. In Figure 4, we compare the results of using all pairs with the results when we use pairs of parses with consecutive ranks. Using only partial pairs makes the algorithm converge much slower.

6 Conclusions and Future Work

In this paper, we have proposed a general framework for reranking. In this framework, we have proposed two new variants of perceptron. Compared to the previous perceptron reranking algorithms, the new algorithms use full pairwise samples and allow us to search for margins in a larger space, which are unavailable in the previous works on reranking. We also keep the data complexity unchanged and make the training efficient for these algorithms. Using the new perceptron like algorithms, we investigated the margin selection problem for the parse reranking task. By using uneven margin on ordinal regression, we achieves an f-score of 89.8% on sentences with $\leq$ 100 words in section 23 of Penn Treebank. The results on margin selection can be employed in reranking systems based on other machine learning algorithms, such as Winnow, Boosting and SVMs.

We plan to apply the new perceptron algorithms to machine translation reranking. More pairwise samples are involved in MT reranking, so that the new algorithms are very suitable.

Acknowledgments

We thank Michael Collins for help concerning the data set.

References

E. Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of NAACL 2000*.

M. Collins and N. Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL 2002*.

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

K. Crammer and Y. Singer. 2001. PRanking with Ranking. In *NIPS 2001*.

Y. Freund and R. E. Schapire. 1999. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

R. Herbrich, T. Graepel, and K. Obermayer. 2000. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press.

W. Krauth and M. Mezard. 1987. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20:745–752.

M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

A. B. J. Novikoff. 1962. On convergence proofs on perceptrons. In *The Symposium on the Mathematical Theory of Automata*, volume 12.

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *the Second Conference on Empirical Methods in Natural Language Processing*.

F. Rosenblatt. 1958. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. 1997. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330.

L. Shen and A. K. Joshi. 2003. An SVM based voting algorithm with application to parse reranking. In *Proc. of CoNLL 2003*.

V. N. Vapnik. 1998. *Statistical Learning Theory*. John Wiley and Sons, Inc.

T. Zhang. 2000. Large Margin Winnow Methods for Text Categorization. In *KDD-2000 Workshop on Text Mining*.