Machine Learning, 60, 73–96, 2005
2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

Ranking and Reranking with Perceptron

LIBIN SHEN libin@linc.cis.upenn.edu
ARAVIND K. JOSHI joshi@linc.cis.upenn.edu
Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA 19104

Editors: Dan Roth and Pascale Fung

Abstract. This work is inspired by the so-called reranking tasks in natural language processing. In this paper, we first study the ranking, reranking, and ordinal regression algorithms proposed recently in the context of ranks and margins. Then we propose a general framework for ranking and reranking, and introduce a series of variants of the perceptron algorithm for ranking and reranking in the new framework. Compared to the approach of using pairwise objects as training samples, the new algorithms reduces the data complexity and training time. We apply the new perceptron algorithms to the parse reranking and machine translation reranking tasks, and study the performance of reranking by employing various definitions of the margins.

Keywords: natural language processing, perceptron, ranking, reranking, margin

1. Introduction

Ranking is one of the most important tasks of relation learning. In recent years, ranking has become an important research topic in the field of machine learning. In the field of Natural Language Processing (NLP), the so-called *reranking* techniques (Collins, 2004) have been successfully used in the parameter estimation in many applications, which were previously modeled as generative models.

In NLP research, it has been noticed that the use of global features is critical to the performance of many NLP systems. However, the introduction of global features results in difficulty for the dynamic programming of the generative models. The basic idea of the reranking is as follows. A baseline generative model generates N-best candidates, and then these candidates are reranked by using a rich set of local and global features. Usually, only the top candidate of the reranked results is used, so reranking is also called *selection* or *voting* in some cases.

Much research has been done on ranking and reranking recently, and various machine learning algorithms have been adapted to the ranking and reranking tasks We will review these works in the next section, emphasizing the similarity and difference in the ranking and reranking problems. In this paper, we will explore the adaptation of the ranking algorithms to the reranking problem.

Obviously, ranking is a problem that lies between classification and regression. In some previous works, ranking was reduced to a classification problem by using pairwise *items* as training samples (Herbrich, Graepel, & Obermayer, 2000), by item we mean a candidate in the N-best list. This will increase the data complexity from $O(n)$ to $O(n^2)$ in the case of full set of pairs, where n is the size of the candidate list. To avoid this, a subset of the pairs are utilized. However, this results in the loss of some information in the training data for reranking. In some other approaches (Crammer & Singer, 2001), extra biases are introduced to avoid using pairwise items as samples; each bias represents the boundary of two neighboring ranks on the score metric. But the use of extra biases prevents it from being used in reranking tasks, as we will explain later.

In this paper, we will introduce a series of variants of the traditional perceptron algorithm (Rosenblatt, 1958; Novikoff, 1962) for ranking and reranking, which fully utilize the key properties of the ranking problem itself. The basic idea of these algorithms is that we search dynamically 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 Novikoff (1962) and Krauth and Mezard (1987).

Based on the new perceptron algorithms, we study the margin selection problem for the parse reranking and machine translation reranking. Experimental results show the superiority of the uneven margins.

The paper is organized as follows. In Section 2, we summarize the previous works on ranking and reranking. Then we investigate these works in the context of ranks and margins in Section 3, and propose general models for ranking and reranking in Section 4. In Section 5 we propose two new perceptron based algorithms in the new framework. We will justify these two algorithms in Section 6. The new algorithms are applied to the parse reranking and the machine translation reranking problems in Section 7.

2. Previous works

2.1. Ranking and ordinal regression

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.

Harrington (2003) extended the PRank algorithm by approximating the Bayes point, thus giving a good performance for generalization. The Bayes point is approximated by averaging the weight vectors and boundaries associated with several PRank algorithms. It is worth mentioning that the large margin effect achieved by approximating the Bayes point can also be implemented by using margin updates (Krauth & Mezard, 1987) in PRank. But there is added complication of the set of boundaries which are essential to the PRank algorithm.

PRank and its extensions work very well for the ranking problems in which each sample is associated with a rank. However, due to the introduction of a set of biases they are constrained from their use in other ranking-like problems. For example, the PRank algorithm cannot be trained on the data associated with a partial order instead of order on ranks.

Furthermore, as we will show later, the PRank algorithm cannot handle the reranking problems.

Herbrich, Graepel, and Obermayer (2000) proposed a margin based approach for ranking, or *ordinal regression* as they called it 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 (Vapnik, 1998) (SVMs) were used to compute the linear 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 for training. For example, there are two samples, $\mathbf{u}$ and $\mathbf{v}$, where $\mathbf{u}$ ranks i and $\mathbf{v}$ ranks $i+1$, 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 to incorporate kernels defined on the space of single items.

The underlying assumption of ordinal regression is that samples between consecutive ranks are separable. This may become a problem in the case that ranks are unreliable when ranking is too fine. This is just what happens in machine translation reranking. On the other hand, if ranking is rather sparse, the size of generated training samples will be very large. Suppose there are n samples evenly distributed on k ranks. The total number of pairwise samples in Herbrich et al. (2000) is roughly n^2/k.

2.2. *Reranking in NLP*

Now we present a brief survey of the reranking task. 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 could be improved 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 in parse reranking. Two approaches were proposed in that paper; one used Boost Loss and the other used Log-Likelihood Loss. The approach of Boost Loss achieved better results. The Boost Loss model is as follows. Let $\mathbf{x}_{i,j}$ be the feature vector of the jth parse of the ith sentence. Let $\tilde{\mathbf{x}}_i$ be the best parse for the ith sentence.[1] 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 example $\mathbf{x}_{i,j}$ 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

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

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.

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

then

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

So the Boost Loss approach in Collins (2000) is the same as 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, although implicitly. The perceptron updating step was defined as

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

where $\mathbf{w}^t$ was the weight vector at the tth updating. This is equivalent to using pairwise sample $\mathbf{s}_{i,j}$ which 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 below, were used as positive samples and negative samples respectively.

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

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

In summary, 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. The other approach is to reduce the ranking problem to a classification problem by using pairwise samples.

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.

3. Ranks and margins

Our initial goal is to adapt ranking algorithms to reranking. However, we note that there are a few fundamental differences between ranking and reranking, which make it hard to use ranking directly on reranking. On the other hand, both ranking and reranking employ pairwise samples to transform the original problem into a classification problem. We will examine these issues in the this section.

3.1. Locality of ranks

First, 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 defined over a *cluster* of the samples in the data set. For example, in the parse reranking problem, the rank of a parse is only the rank among all the parses for the same sentence. The training data include 36,000 sentence, with an average of over 27 parses per sentence (Collins, 2000).

As a result, we cannot use the PRank algorithm in the reranking task, since there are no global ranks or boundaries for all the samples. 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 still works. By pairing up two samples, we actually compute the relative distance between these two samples with respect to the scoring metric. In the training phase, we are only interested in whether the relative distance is positive or negative, and we do not need to compare it with any specific numbers.

To sum up, the locality of ranks in the reranking problem leads us to the approach of using pairwise samples.

3.2. Density of ranks

In the previous works on ranking, the number of samples is much larger than the number of ranks. For example, the Information Retrieval (IR) data used in Herbrich, Graepel, and Obermayer (2000) has only 3 ranks (“document is relevant”, “document is partially relevant”, and “irrelevant document”), the synthetic data sets used in Herbrich, Graepel, and Obermayer (2000), and Crammer and Singer (2001) have 5 ranks, and the EachMovie dataset used in Crammer and Singer (2001) and Harrington (2003) has 6 ranks. In these applications, difference between samples of different ranks is significant, so it is natural to define margin as distance between consecutive ranks.

However, this approach is problematic if ranks become denser and the samples of consecutive ranks become linearly inseparable. For example, in the IR data used in Herbrich, Graepel, and Obermayer (2000), we are not only interested in whether a document is relevant or not, but also the extent to which the document is relevant. In this case, pairwise samples defined only on the consecutive ranks are not enough for training, since ranks within a small range is unstable; it is hard to say if a document ranked i is more relevant than a document ranked $i + 1$.

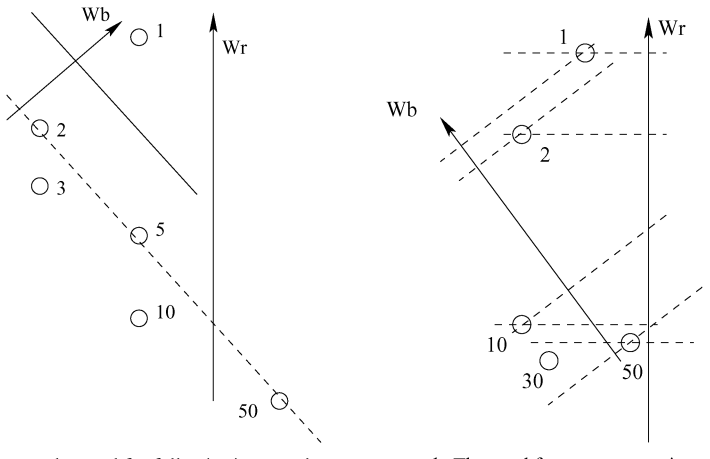
Wb
Wr
1
2
3
5
10
50
a. The need for full pairwise samples
Wr
Wb
1
2
10
30
50
b. The need for uneven margins

Figure 1 Comparison of score metrics.

However, ranks in a large range is reliable; a document ranked 5 is more relevant than a document ranked 15. So we may want to use more pairwise samples. For example, we may use (doc_5, doc_{15}) as a positive sample, and use (doc_{15}, doc_5) as a negative sample. The extreme case is pairing up all the samples. This will increase the size of sample space by N times, where N is size of a cluster. To sum up, in the case of dense ranks, the full pairwise samples or a subset ofthe full pairwise samples are required in training.

Obviously, the ranks in reranking are *dense*. Each training cluster has a ranked list of N-best candidate. For parse reranking $N = 27$ on average (Collins, 2000), and for machine translation reranking $N = 1000$ or more (Och et al., 2004). Therefore, pairwise samples defined only on consecutive ranks are not enough in the training of reranking.

3.3. Full pairwise for reranking

As we have shown above, in the previous works on reranking, we search for the hyperplane that separates the best candidate from the rest of the candidates within each cluster. Given a cluster, let $\mathbf{r}_i$ be the ith best item within the cluster. 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 $\mathbf{r}_2$ is better than $\mathbf{r}_{50}$ in the training, as shown in Figure 1(a).

We may regard the weight vector $\mathbf{w}$ as a score metric, whichassigns the highest score to the best item within each cluster, i.e. $\mathbf{r}_1$. So the question is whether it should assign the second highest score to $\mathbf{r}_2$, the third highest score to $\mathbf{r}_3$, and so on so forth, since $\mathbf{w}$ is a score metric. Therefore, in Figure 1(a), $\mathbf{w}_r$ is more preferable than $\mathbf{w}_b$ since $\mathbf{w}_r$ is able to keep the order of items with respect to their goodness.

In order to search for $\mathbf{w}_r$ instead of $\mathbf{w}_b$, we need to employ more ordering relations in the training. One solution is to use a set of items as *good* candidates, $\mathbf{r}_{\text{good}}$, and use the rest as bad candidates, $\mathbf{r}_{\text{bad}}$. For candidates $\mathbf{r}_{\text{good}}$, $\mathbf{r}_{\text{bad}}$ within the same cluster, $\mathbf{r}_{\text{good}} - \mathbf{r}_{\text{bad}}$ is defined as a positive sample, and $\mathbf{r}_{\text{bad}} - \mathbf{r}_{\text{good}}$ is defined a negative sample. In Section 4, we will use the *splitting* model to capture this idea.

However, there is still a problem with this solution, i.e. it does not distinguish the differences between the good samples. In the case of machine translation for which N can be as large as 1000 or even more, $\mathbf{r}_1$ and $\mathbf{r}_{300}$ will be both regarded as good candidates for example, so the *splitting* model does not try to assign a higher score to $\mathbf{r}_1$ than to $\mathbf{r}_{300}$.

As far as this aspect is concerned, we apply the ordinal regression model to reranking with full pairwise samples. That is to say, for every two candidates $\mathbf{r}_i$ and $\mathbf{r}_j$, $i < j$, within the same cluster, $\mathbf{r}_i - \mathbf{r}_j$ is used as a positive sample, and $\mathbf{r}_j - \mathbf{r}_i$ is used as a negative sample. However, in the next section we will show that ordinal regression is not a desirable model for reranking either.

3.4. *Uneven margins*

Reranking is not an ordinal regression problem. In reranking evaluation, we are only interested in the quality of the item with the highest score in each cluster, and we do not care the order of the other items. Therefore we cannot simply regard a reranking problem as an ordinal regression problem, since they have different loss functions.

Especially, we want to maintain a larger margin in items of high ranks and a smaller margin in items of low ranks. For example, in Figure 1(b), $\mathbf{w}_b$ meets the condition of ordinal regression by keeping the order of all candidates within a cluster. However, there is a small margin between $\mathbf{r}_1$ and $\mathbf{r}_2$, which means that if we are given two candidates similar to $\mathbf{r}_1$ and $\mathbf{r}_2$ in the test set, $\mathbf{w}_b$ is very likely to switch their order. However, $\mathbf{w}_r$ maintains a large margin between $\mathbf{r}_1$ and $\mathbf{r}_2$. Hence, $\mathbf{w}_r$ is more desirable even it fails to keep the order of some bad candidates, i.e. $\mathbf{r}_{30}$ and $\mathbf{r}_{50}$, but they are far away from the top candidates. According to the loss function, we are penalized if we switch the order of $\mathbf{r}_1$ and $\mathbf{r}_2$, but not for $\mathbf{r}_{30}$ and $\mathbf{r}_{50}$.

In order to handle this, we will introduce the idea of uneven margins in the training. The technique of uneven margins has been previously employed in binary classification on unbalanced data (Li et al., 2002), for which there are many more negative samples than positive samples. Similarly, with respect to the loss function, the penalty on errors over positive samples is much more than the errors on negative samples. In that paper, the idea of uneven margins is to maintain a large margin for the positive samples, and a relatively narrow margin for the negative samples.

Therefore, Ordinal Regression with Uneven Margins (ORUM) is more desirable for reranking. Specifically, we search for a linear function separating items in each cluster with uneven margins, for example

$$margin(\mathbf{r}_1, \mathbf{r}_{30}) > margin(\mathbf{r}_1, \mathbf{r}_{10}) > margin(\mathbf{r}_{21}, \mathbf{r}_{30}) \tag{1}$$

It is not difficult to exhibit a function that satisfies (1), for example

$$g(\mathbf{r}_i, \mathbf{r}_j) = \frac{1}{i} - \frac{1}{j} \tag{2}$$

It would be interesting to investigate which margin function is theoretically desirable. However, in this paper, we will simply use this margin function in our experiments.

4. Models for ranking and reranking

4.1. Problem definition

We first formalize the reranking problem.

Item $\mathbf{x} \in \mathcal{R}^d$ is a d-dimensional vector on the real number. It represents a single parse, or a single translation in applications.

Cluster $\mathbf{c} = (\mathbf{x}_1, \ldots, \mathbf{x}_k) \in (\mathcal{R}^d)^k$ is a vector of k items. It represents a ranked list of parses or translations for the same source sentence in applications.

Rank $\mathbf{r_c} = (y_1, \ldots, y_k) \in \mathcal{N}^k$, where y_i is the rank of item $\mathbf{x}_i$ in $\mathbf{c}$, $1 \leq y_i \leq k$.

Learning Let $\mathcal{X}$ be the space of clusters, $\mathcal{X} = (\mathcal{R}^d)^k$, and let $\mathcal{Y}$ be the space of ranks, $\mathcal{Y} = \mathcal{N}^k$. The hypothesis class $\mathcal{H}$ is $\mathcal{X} \rightarrow \mathcal{Y}$. A learning algorithm $\mathcal{L}((\mathcal{X} \times \mathcal{Y})^m)$ takes m *i.i.d.* drawn training clusters associated with ranks from $\mathcal{X} \times \mathcal{Y}$ according to distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$, and outputs a hypothesis $h \in \mathcal{H}$.

Margin is a function on $\mathcal{X} \times \mathcal{Y} \times \mathcal{H} \rightarrow \mathcal{R}$.

4.2. Splitting

In this framework, we will first propose a *splitting* model, which is similar to previous works on reranking, but in a more general form.

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 clusters and k is the length of ranks for each cluster.

Let $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ be a linear function, where $\mathbf{x}$ is an item, and $\mathbf{w}$ is a weight vector. We construct a hypothesis function $h_f : \mathcal{X} \rightarrow \mathcal{Y}$ with f as follows.

$$h_f(\mathbf{x}_1, \ldots \mathbf{x}_k) = rank(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_k)),$$

where *rank* is a function that returns the vector of ranks for the input vector. For example *rank*(4.2, 9.0, 6.5, 8.8) = (4, 1, 3, 2).

An *r-splitting* algorithm searches for a linear function $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$ that successfully splits the top r-ranked items from the rest of the items in every cluster. Let $\mathbf{y}^f = (y_1^f, \ldots, y_k^f) = h_f(\mathbf{x}_1, \ldots \mathbf{x}_k)$ for any linear function f. We look for a function f such that

$$y_i^f \leq r \quad \text{if } y_i \leq r \tag{3}$$

$$y_i^f > r \quad \text{if } y_i > r \tag{4}$$

Suppose there exists a linear function *f* satisfying (3) and (4), we say $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ is *r-splittable* by *f*. Furthermore, we can define the *splitting margin* γ for cluster $\mathbf{c}_i$ as follows.

$$\gamma(f, r, i) = \min_{j:y_{i,j}\leq r} f(\mathbf{x}_{i,j}) - \max_{j:y_{i,j}>r} f(\mathbf{x}_{i,j})$$

The *minimal splitting margin*, γ^{split}, for *f* and *r* is defined as follows.

$$\gamma^{split}(f, r) = \min_i \gamma(f, r, i) = \min_i \left(\min_{y_{i,j}\leq r} f(\mathbf{x}_{i,j}) - \max_{y_{i,j}>r} f(\mathbf{x}_{i,j})\right)$$

Obviously, if $r = 1$, the *r-splitting* searches for a linear function *f* that can successfully separate the best item from the rest in every cluster, which is similar to some previous reranking algorithms (Collins, 2000; Collins, & Duffy, 2002; Shen & Joshi, 2003).

4.3. Ordinal regression on clusters

In this section we will model a more complicated problem, ordinal regression on clusters. Let $\mathbf{x}_{i,j}, \mathbf{x}_{i,l}$ be two items where $y_{i,j} < y_{i,l}$. It means that the rank of $\mathbf{x}_{i,j}$ is higher than the rank of $\mathbf{x}_{i,l}$. We are interested in finding a weight vector $\mathbf{w}$, such that

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

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 clusters and k is the length of ranks for each cluster. Let $f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}$. We say *f* is an ordinal regression function for the training set *S* if

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

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

Suppose *f* is an ordinal regression function for the training set *S*, the *regression margin* for cluster $\mathbf{c}_i$ is defined as follows

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

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

$$\gamma^{\text{order}}(f) = \min_i \gamma(f, i)$$

4.4. Pairwise classification

The two models described in the previous two sections can be generalized as a classification problem by using pairwise samples. Let the training samples be

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

where m is the number of clusters and k is the length of ranks for each cluster.

Let $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$. We say the training data is *separable* with respect to T by f if

$$\mathbf{w} \cdot \mathbf{x}_{i,j} > \mathbf{w} \cdot \mathbf{x}_{i,l}, \quad \text{if } (y_{i,j}, y_{i,l}) \in T,$$

for $1 \le i \le m,\ 1 \le j, l \le k$, and $T \subseteq \mathcal{K} \times \mathcal{K}$ is a partial order on $\mathcal{K}$, where $\mathcal{K} = \{1, 2, \ldots, k\}$.

It is not difficult to see that both splitting and ordinal regression can be reduce to the model given above. For splitting,

$$(y_{i,j}, y_{i,l}) \in T \Leftrightarrow y_{i,j} \le r < y_{i,l},$$

and for ordinal regression

$$(y_{i,j}, y_{i,l}) \in T \Leftrightarrow y_{i,j} < y_{i,l}.$$

Furthermore, this model can be transformed into a binary classification problem as follows. $\forall (y_{i,j}, y_{i,l}) \in T$, $\mathbf{x}_{i,j} - \mathbf{x}_{i,l}$ is defined as a positive sample, and $\mathbf{x}_{i,l} - \mathbf{x}_{i,j}$ is defined as a negative sample. Therefore, the model above is equivalent to finding a linear function separating the positive and negative samples. Furthermore, the classification margin is equivalent to splitting and regression margin respectively.

4.5. Pairwise classification with uneven margins

Let the training samples be

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

where m is the number of clusters and k is the length of ranks for each cluster.

Let $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$, where $\|\mathbf{w}\| = 1$. We say the training data is *separable* with respect to T by f with margin τ weighted with margin function g if

$$\mathbf{w} \cdot (\mathbf{x}_{i,j} - \mathbf{x}_{i,l}) > g(y_{i,j}, y_{i,l})\tau, \quad \text{if } (y_{i,j}, y_{i,l}) \in T, \tag{5}$$

for $1 \le i \le m,\ 1 \le j, l \le k$, $T \subseteq \mathcal{K} \times \mathcal{K}$ is a partial order on $\mathcal{K}$, and $g \in \mathcal{K}^2 \to \mathcal{R}$ is a margin function, where $\mathcal{K} = \{1, 2, \ldots, k\}$.

4.6. Single cluster

These models for multi-cluster samples can be easily reduced to 1-cluster case, which can be used to model the ranking problem.

Item $\mathbf{x} \in \mathcal{R}^d$ is a d-dimensional vector on the real number.
Rank $y \in \mathcal{N}$, where $1 \leq y \leq k$ for some fixed number k.
Learning Let $\mathcal{X}$ be the space of items, $\mathcal{X} = \mathcal{R}^d$, and let $\mathcal{Y}$ be the space of ranks, $\mathcal{Y} = \mathcal{N}$. The hypothesis class $\mathcal{H}$ is $\mathcal{X} \to \mathcal{Y}$. A learning algorithm $\mathcal{L}((\mathcal{X} \times \mathcal{Y})^m)$ takes m *i.i.d.* drawn training clusters associated with ranks from $\mathcal{X} \times \mathcal{Y}$ according to distribution $\mathcal{D}_{\mathcal{X}\times\mathcal{Y}}$, and outputs a hypothesis $h \in \mathcal{H}$.
Margin is a function defined on $\mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathcal{R}$.

In the case of a single cluster, items, instead of clusters, are drawn *i.i.d.* with respect to some distribution.

5. Perceptron algorithms

There are quite a few linear classifiers that can separate samples with large margin, such as Support Vector Machines (Vapnik, 1998), Boosting (Schapire et al., 1997), Winnow (Zhang 2000) and Perceptron (Krauth & Mezard, 1987). In this paper, we will investigate the perceptron algorithm for the following reasons. First, the perceptron is fast in training, which allows us to do experiments with various margin selections on real-world data. Further, the perceptron algorithm is simple, which makes modification easy to implement.

In this paper, we will investigate uneven margins for reranking with the new perceptron algorithms. The results with perceptron experiments can be extended to other machine learning algorithms too.

5.1. Perceptron over full pairwise samples

Algorithm 1 is used to solve the Pairwise Classification with Uneven Margins (PCUM) model proposed in Section 4.5. The idea of Algorithm 1 is as follows. For every two items $\mathbf{x}_{i,j}$ and $\mathbf{x}_{i,l}$, if

- $(y_{i,j}, y_{i,l}) \in T$, and
- the weight vector $\mathbf{w}$ can not successfully separate $\mathbf{x}_{i,j}$ and $\mathbf{x}_{i,l}$ with a learning margin $g(y_{i,j}, y_{i,l})\tau$,

then we need to update $\mathbf{w}$ with the addition of $g(y_{i,j}, y_{i,l})(\mathbf{x}_{i,j} - \mathbf{x}_{i,l})$.

In Section 6 we will give the theoretic justification of Algorithm 1.

If T is the full order, the size of T is $k(k-1)/2$, where k is the size of a cluster. If we represent all the pairwise sample explicitly, the data complexity will be $dmk(k-1)/2$, where d is the dimensionality of a sample. In the research of machine translation reranking (Och et al., 2004), we have roughly 1000 source sentences, each of which has 1000 best translations, and each candidate is represented with a vector of 20 real-valued features. In

Algorithm 1 Pairwise Classification with Uneven Margins

```
Require: $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ is *separable* with respect to $T$.
Require: a margin function $g$
Require: a positive learning margin $\tau$.
 1: $t \leftarrow 0$, initialize $\mathbf{w}^0$;
 2: repeat
 3:   for $(i = 1, ..., m)$ do
 4:     for $(1 \leq j, l \leq k)$ do
 5:       if $((y_{i,j}, y_{i,l}) \in T$ and $\mathbf{w}^t \cdot (\mathbf{x}_{i,j} - \mathbf{x}_{i,l}) \leq g(y_{i,j}, y_{i,l})\tau)$ then
 6:         $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + g(y_{i,j}, y_{i,l})(\mathbf{x}_{i,j} - \mathbf{x}_{i,l})$
 7:         $t \leftarrow t + 1$
 8:       end if
 9:     end for
10:   end for
11: until no updates made in the outer for loop
```

this case, the size of pairwise samples is roughly 40G bytes. Thus, we can only compute pairwise samples dynamically in our algorithm.

In Algorithm 1, for each iteration of a cluster, we need execute the dot product operation for $|T|$ times, which is usually $O(k^2)$. Thus the complexity of outer iteration is $O(k^2d)$. In the next section, we will give an alternative algorithm, with which we reduce the complexity of outer iteration to $O(k^2 + kd)$.

5.2. *Fast perceptron training*

Algorithm 2 is similar to Algorithm 1 except that the updating operation is executed on the cluster level instead of sample level. For each iteration of a cluster, we first compute $\mathbf{w} \cdot \mathbf{x}_{i,j}$ for each item $\mathbf{x}_{i,j}$. Then we use these values to check whether an update on $\mathbf{w}$ is required for each pair in T. However, the update is not executed immediately. Instead, for each item, there is a valuable $u_{i,j}$ that records all the updates. After all the pairs are checked, updating is executed once and for all.

It is easy to show that the complexity of each iteration of a cluster is $O(k^2 + kd)$. In Section 6 we will show that Algorithm 2 is theoretically correct, and in the experimental section we will show the superiority of Algorithm 2 over Algorithm 1.

5.3. *The landscape of perceptron based algorithms*

So far we have presented two new perceptron based algorithms which are inspired by the ranking and reranking problems. Compared to the previous works, our perceptron-like algorithm is more general as shown in Table 1. Nov62 = (Novikoff, 1962), KM87 = (Krauth & Mezard, 1987), CS01 = (Crammer & Singer, 2001), WH60 = (Widrow & Hoff, 1960), and CD02 = (Collins & Duffy, 2002).

First, we have three different tasks—classification, ranking, and regression, in the order of increasing difficulty. Classification is the easiest and regression is the most difficult. Now

Algorithm 2 Fast Pairwise Classification with Uneven Margins

```
Require: $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ is separable with respect to $T$.
Require: a margin function $g$
Require: a positive learning margin $\tau$.
1:  $t \leftarrow 0$, initialize $\mathbf{w}^0$;
2:  repeat
3:    for ($i = 1, ..., m$) do
4:      for ($j = 1, ..., k$) do
5:        compute $\mathbf{w}^t \cdot \mathbf{x}_{i,j}$ for all $j$'s;
6:        $u_j \leftarrow 0$;
7:      end for
8:      for ($1 \leq j, l \leq k$) do
9:        if ($(y_{i,j}, y_{i,l}) \in T$ and $\mathbf{w}^t \cdot (\mathbf{x}_{i,j} - \mathbf{x}_{i,l}) \leq g(y_{i,j}, y_{i,l})\tau$) then
10:         $u_j \leftarrow u_j + g(y_{i,j}, y_{i,l})$; $u_l \leftarrow u_l - g(y_{i,j}, y_{i,l})$;
11:       end if
12:     end for
13:     if ($\sum_j |u_j| > 0$) then
14:       $\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t + \sum_j u_j \mathbf{x}_{i,j}$;
15:       $t \leftarrow t + 1$;
16:     end if
17:   end for
18: until no updates made in the outer for loop
```

Table 1. The landscape of perceptron based algorithms.

Model	explicit + single	sub-model	implicit + single	implicit + multi
Classification	Nov62, KM87	$1 : n$	–	CD02
		$m : n$	**Alg 1**	
Ranking	CS01	–	**Alg 1**	
Regression	WH60	–	–	–

we have two different kinds of data. The simpler case is that the whole data contains only one cluster. The perceptron style algorithms for these three tasks are listed in column 2 of Table 1. They all use biases in training explicitly.

In some cases, the training data contains more than one cluster, such as the reranking tasks in NLP. For these problems, we cannot use biases explicitly in training, and the solution is to use pairwise samples so that bias is treated implicitly. This method can also be used in a one-cluster problem (Herbrich, Graepel, & Obermayer, 2000).

There are two sub-models for the classification problem using pairwise samples. One is to divide, in every cluster, top candidate from the rest, the $1 : n$ model. Almost all the previous works on NLP reranking tasks fall into this category. A more general model, $m : n$ splitting, is handled by the splitting model as described in Section 4.2.

It is worth mentioning that, in recent years, the perceptron-like algorithms have also been adapted to multi-label setting and topic ranking problems, such as Crammer and Singer (2003), which are related to our work, but are not shown in Table 1.

6. Theoretical justification

In the previous sections, we have proposed two perceptron based large margin algorithms for splitting and ordinal regression models. Now we show that these two algorithms stop in finite steps if the training data is separable, and present a lower bound of the resulting margin which is used to constrain the expected error rate.

The method of reducing a complicated task into simple binary classification tasks has been used in problems like multi-class classification (Crammer & Singer, 2002) and constraint classification (Har-Peled, Roth, & Zimak, 2002). One potential problem is that, in the binary classification tasks, the expanded samples are not generated *i.i.d.*, which is important for margin-based bounds. The remedy is to reduce the dependent samples (Herbrich, Graepel, & Obermayer, 2000) and to introduce a new distribution with respect to which classification samples are drawn independently (Har-Peled, Roth, & Zimak, 2002).

6.1. *Justification for Algorithm 1*

Theorem 1 gives us an upper bound on the number of steps needed for Algorithm 1 to stop, if the training data is separable. The proof for Theorem 1 is given the Appendix.

Theorem 1. *Suppose the training samples $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ are separable with respect to T by a linear function defined on the weight vector $\mathbf{w}^*$ with a margin γ weighted on margin function g as in (5), where $\|\mathbf{w}^*\| = 1$. Let $R = \max_{i,j} \|\mathbf{x}_{i,j}\|$ and $\lambda = \min_{i,j,l} \tilde{g}(y_{i,j}, y_{i,l})$. Then we have*
(a) Algorithm 1 stops in t steps of updates, where

$$t \leq \frac{2\tau + 4R^2}{\lambda^2\gamma^2} \tag{6}$$

(b) The resulting weight vector separates any two item $\mathbf{x}_{i,j}$ and $\mathbf{x}_{i,l}$, where $y_{i,j} < y_{i,l}$, with margin

$$\gamma_{i,j,l} \geq g(y_{i,j}, y_{i,l})\gamma \frac{\tau}{2\tau + 4R^2} \tag{7}$$

According to (7), if $\tau >> R$, Algorithm 1 stops with resulting margins at least half of the optimum margins, $g(y_{i,j}, y_{i,l})\gamma$.

Then we can apply margin-based generalization bounds (Vapnik, 1998; Cristianini & Shawe-Taylor, 2000). A potential problem is that pairwise samples defined on $S = \{(\mathbf{x}_{i,j}, y_{i,j})\}$ are not independent, thus violating the *i.i.d.* assumption. The way to avoid this problem is to employ only $k - 1$ independent pairwise samples for each cluster, as in Herbrich et al. (2000).

Further, we transform the uneven margins to a unique margin by redefining the pairwise samples as

$$S' = \left\{ \frac{\mathbf{x}_{i,j} - \mathbf{x}_{i,l}}{g(y_{i,j}, y_{i,l})} \middle| (y_{i,j}, y_{i,l}) \in T \right\} \tag{8}$$

In order to give the error bound on the test data, we transform the test data in the same way.

Given the resulting margin, $\gamma \frac{\tau}{2\tau+4R^2}$ according to (7) and (8), on S', we can easily bound the error on the transformed test data under PAC frame by using margin-based generalization bounds given in Vapnik (1998) Cristianini & Shawe-Taylor (2000).

Then we can obtain the expected error bound for the binary classification. However, it is well known that the bound given in this way is a loose one. It will be interesting to search for a tight bound for the reranking problem, for example by employing the growth function of ranking (Har-Peled, Roth, & Zimak, 2002) and utilizing its difference with ordinal regression as described above.

6.2. *Justification for Algorithm 2*

For Algorithm 2, we will show that if the training data is separable, the algorithm will stop in a finite number of step with a resulting margin on the training data. For the sake of simplicity, we will only take a weak version of the algorithm, by assuming $g(y_{i,j}, y_{i,l}) \equiv 1$ for all i, j, l. However, the theorem and the proof given here can be extended to the original algorithm.

Theorem 2. *Suppose the training samples* $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ *are separable by a linear function defined on the weight vector* $\mathbf{w}^*$ *with a margin* γ, *where* $\|\mathbf{w}^*\| = 1$. *Let* $R = \max_{i,j} \|\mathbf{x}_{i,j}\|$. *Then we have*

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

(b) Algorithm 2 stops in t steps of updates, where

$$t \le \frac{2\tau + k^2R^2}{\gamma^2} \tag{9}$$

(c) The resulting margin on the training data is at least

$$\gamma \frac{\tau}{2\tau + k^2R^2} \tag{10}$$

The proof of Theorem 2 is given in the Appendix.

It is worth mentioning that Algorithm 2 needs more iterations for convergence than Algorithm 1 theoretically, while the former runs much faster in each iteration. Experiments given in the next section show that the running time for Algorithm 2 is shorter, and the result is more stable.

7. Experiments and discussion

In this section, we show the experimental results on two NLP tasks, parse reranking and discriminative reranking for machine translation, which were partially reported in Shen and Joshi (2004) and Shen et al. (2004) respectively. A single Pentium III 1.13 GHz cpu with 2 GB memory is used for all experiments. The algorithms are coded with Java, running on the Linux operating system.

7.1. Parse reranking

For parse reranking, we use the same data set as described in Collins (2000). Section 2-21 of the WSJ Penn Treebank (PTB) (Marcus, Santorini, & Marcinkiewicz, 1994) are used as training data, and Section 23 is used for testing. 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 in total. We use the feature set generated by Collins (2000). There are 521,498 features in all.

We use Algorithm 2 in the first set of experiments. By using different settings of the pairwise samples and the margins, we design the experiments as follows.

– **1-splitting, CD02**: This setting is used to simulate the perceptron algorithm introduced in Collins and Duffy (2002).[2] Only one of the best parses for each sentence is used as the good parse and the other best parses are dropped. Other parses are used as bad parses.
– **r-splitting, even margins**: all the best parses are used as good parses. Other parses are used as bad parses.
– **r-splitting, uneven margins**: all the best parses are used as good parses. The margin function $g(\mathbf{r}_i, \mathbf{r}_j) = \frac{1}{i} - \frac{1}{j}$, which was previously defined in (2), is employed.
– **ordinal regression, even margins**: $T = \{(j, l) \mid j < l\}$.
– **ordinal regression, uneven margins**: $T = \{(j, l) \mid j < l\}$. The margin function defined in (2) is employed.

By estimating the number of rounds of iterations on the development data, we get the results on the test data as shown in Table 2. Ordinal regression with uneven margins achieves the best result in f-score.[3] It verifies that using more pairs in training is helpful for the reranking problem. Uneven margins are crucial for employing full pairwise models to reranking.

We compare the performance of Algorithm 1 and Algorithm 2 on our best model, ordinal regression with uneven margins. Figure 2 shows the comparison of the learning curves of these two algorithms. For Algorithm 1, we compute the score of all pairwise samples, $\mathbf{w}^t \cdot \mathbf{x}_{i,j} - \mathbf{w}^t \cdot \mathbf{x}_{i,l}$, on the fly.

Algorithm 1 converges faster than Algorithm 2 in terms of rounds, which is consistent with the theoretical justification. However, Algorithm 2 converges faster in terms of running time. The results for f-score are similar, But the result of Algorithm 2 is more stable; it

Table 2. Experimental results.

Section 23, ≤100 words (2416 sentences)			
Model	Recall%	Precision%	F-score%
baseline Collins (1999)	88.1	88.3	88.2
1-splitting, CD02	89.2	89.8	89.5
r-splitting, even margins	89.1	89.8	89.5
r-splitting, uneven margins	89.3	90.0	89.6
ordinal regression, even margins	88.1	87.8	88.0
ordinal regression, uneven margins	89.5	90.0	**89.8**

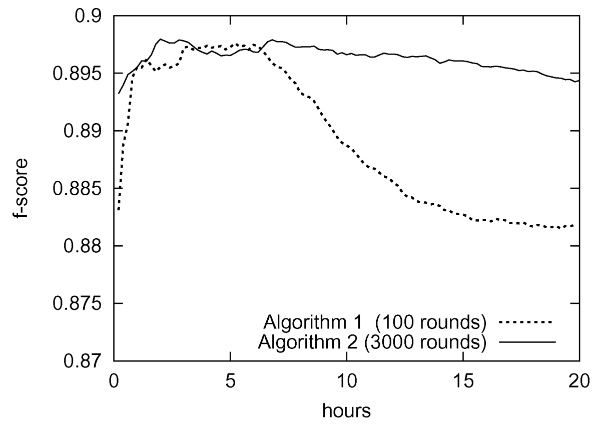

Figure. 2 Learning curves of Algorithm 1 and Algorithm 2 on PTB Section 23.

does not over-fit the training data even after 3000 rounds of iteration, while Algorithm 1 over-trains quickly in terms of rounds. We think the reason is that Algorithm 1 updates the weight vector whenever there is a classification error on a pairwise sample, thus it is more likely to drop into a local optimum. While Algorithm 2 to some extent alleviates the ill-posed problem[4] of the perceptron algorithm.

7.2. *Discriminative reranking for machine translation*

We provide experimental results on the NIST 2003 Chinese-English large data track evaluation, which was partially reported in Shen, Sarkar, and Och (2004). We use the data set used in Och et al. (2004). The training data consists of about 170 M English words, on which the baseline translation system is trained. The training data is also used to build language models which are used to define feature functions on various syntactic levels. The development data consists of 993 Chinese sentences. Each Chinese sentence is associated with 1000-best English translations generated by the baseline MT system. The

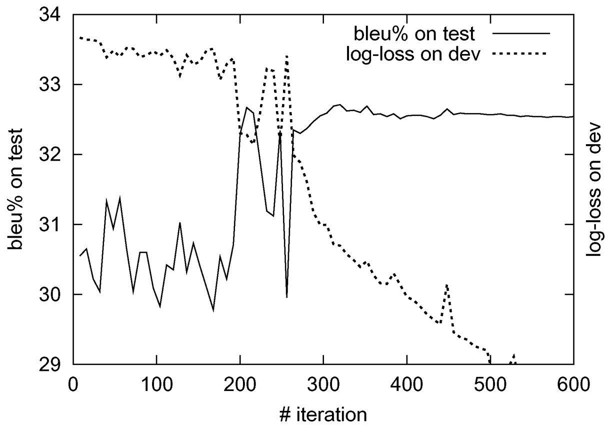

Figure. 3 Splitting with uneven margins.

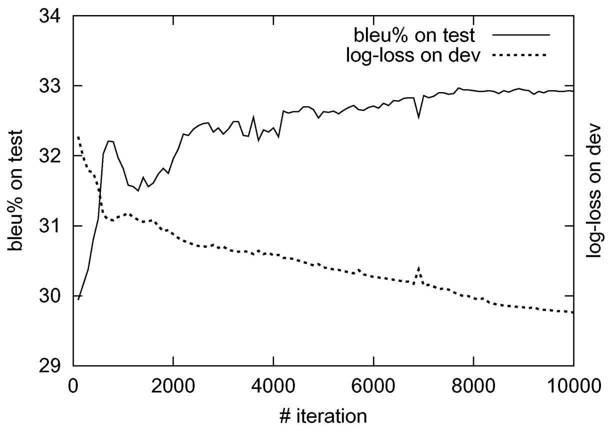

Figure. 4 Ordinal regression with uneven margins.

development data set is used to estimate the parameters for the feature functions for the purpose of reranking. The test data consists of 878 Chinese sentences. Each Chinese sentence is associated with 1000-best English translations too. The test set is used to assess the quality of the reranking output.

In Och et al. (2004), aggressive search was used to combine features (Och, 2003). After combining about a dozen features, the BLEU score (Papineni, Roukos, & Ward, 2001) did not improve any further, and the score was 32.9%. In our experiments, we will use the top 20 individual features developed by Och et al. (2004). Algorithm 2 is used with the three different settings as follows. A distinct dimension is introduced for each sample to make the training data separable, as in Li et al. (2002).

– **best vs rest**: $T = \{(1, j) \mid j > 1\}$ and even margin.
– **splitting, uneven margins**: $T = \{(j, l) \mid j <= 300 \text{ and } l >= 700\}$. The margin function defined in (2) is employed.
– **ordinal regression, uneven margins**: $T = \{(j, l) \mid j * 2 < l \text{ and } j + 20 < l\}$. The margin function defined in (2) is employed.

The best vs. rest setting is used to simulate the previous perceptron reranking algorithm (Collins & Duffy, 2002). It converged in 7 iterations, and achieved BLEU score of 30.9%. The learning curve of the Splitting and the ORUM is shown in Figures 3 and 4. The Splitting algorithm achieved BLEU score of 32.6%, and the ORUM algorithm achieved BLEU score of **32.9**%, which is significantly better than the best vs. rest model.

According to the learning curves, we notice that whenever the log-loss on the development set decreases, the BLEU score on the test set increases. This experimentally verifies the theoretical justification of these two algorithms.

8. Conclusions

In this paper, we have proposed a general framework for ranking and reranking. In this framework, we have proposed two variants of perceptron, which are trained on pairwise samples. Using these two algorithms, we can employ more pairwise samples, which are useful in training a ranker or reranker. We also keep the data complexity unchanged and make the training efficient with these algorithms.

Using these two algorithms, we investigated the margin selection problem for the reranking tasks. By using uneven margins on ordinal regression, we achieves an f-score of 89.8% on sentences with $\leq$ 100 words in Section 23 of Penn Treebank. In machine translation reranking, our perceptron-like algorithm matches the state-of-the-art discriminative MT reranking system reported in Och (2003). The idea on uneven margins on full pairwise samples can be employed in reranking systems based on other machine learning algorithms, such as Winnow, Boosting and SVMs, via the definition of the margins or loss functions.

Appendix

Proof of Theorem 1

Proof. We show it by bounding $\|\mathbf{w}^t\|^2$ from above and below. Suppose $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ is separable with respect to T with margin γ weighted on a margin function g as in (5), and $g(a, b) = -g(b, a)$ if $b < a$. The training data can be noted as $S = \{(\mathbf{u}_r, v_r)\}$ with margin weight $\{\lambda_r\}$ where

$$\begin{aligned}
\mathbf{u}_r &= \mathbf{x}_{i,j} - \mathbf{x}_{i,l},\\
v_r &= \begin{cases} 1 & \text{if } (y_{i,j}, y_{i,l}) \in T,\\ -1 & \text{if } (y_{i,l}, y_{i,j}) \in T,\end{cases}\\
\lambda_r &= v_r g(y_{i,j}, y_{i,l}),
\end{aligned}$$

for i, j, l s.t. $(y_{i,j}, y_{i,l}) \in T$ or $(y_{i,l}, y_{i,j}) \in T$. Therefore, we have

$$v_r(\mathbf{w}^* \cdot \mathbf{u}_r) > \lambda_r \gamma, \tag{11}$$

where $1 \leq r \leq \mid S \mid$, $\|\mathbf{w}^*\|^t = 1$. Thus

According to algorithm 1,

$$\mathbf{w}^t = \mathbf{w}^{t-1} + v^t \lambda^t \mathbf{u}^t, \text{ if } v^t \mathbf{w}^{t-1} \cdot \mathbf{u}^t \leq \lambda^t \tau$$

Thus

$$\begin{aligned}\|\mathbf{w}^t\|^2 &= (\mathbf{w}^{t-1} + v^t \lambda^t \mathbf{u}^t)^2 \\ &= \|\mathbf{w}^{t-1}\|^2 + 2\lambda^t v^t \mathbf{w}^{t-1} \cdot \mathbf{u}^t + \lambda^t \lambda^t \|\mathbf{u}^t\|^2 \\ &\leq \|\mathbf{w}^{t-1}\|^2 + 2\lambda^t \lambda^t \tau + \lambda^t \lambda^t (2R)^2 \\ &\leq \sum_{p=1}^{t} \lambda^p \lambda^p (2\tau + 4R^2)\end{aligned} \tag{12}$$

On the other hand,

$$\begin{aligned}\mathbf{w}^* \cdot \mathbf{w}^t &= \mathbf{w}^* \cdot (\mathbf{w}^{t-1} + v^t \lambda^t \mathbf{u}^t) = \mathbf{w}^* \cdot \mathbf{w}^{t-1} + v^t \lambda^t \mathbf{w}^* \cdot \mathbf{u}^t \\ &> \mathbf{w}^* \cdot \mathbf{w}^{t-1} + \lambda^t \lambda^t \gamma > \sum_{p=1}^{t} \lambda^p \lambda^p \gamma\end{aligned} \tag{13}$$

Combining (12) and (13), we have

$$\left(\sum_{p=1}^{t} \lambda^p \lambda^p \gamma\right)^2 < \|\mathbf{w}^* \cdot \mathbf{w}^t\|^2 = \|\mathbf{w}^t\|^2 \leq \sum_{p=1}^{t} \lambda^p \lambda^p (2\tau + 4R^2)$$

$$\left(\sum_{p=1}^{t} \lambda^p \lambda^p\right) \leq \frac{\sum_{p=1}^{t} \lambda^p \lambda^p (2\tau + 4R^2)}{\sum_{p=1}^{t} \lambda^p \lambda^p \gamma^2} = \frac{2\tau + 4R^2}{\gamma^2} \tag{14}$$

Let $\lambda_{\min} = \min_r \lambda_r$. With (14), we have

$$t \leq \frac{\sum_{p=1}^{t} \lambda^p \lambda^p}{\lambda_{\min}^2} \leq \frac{2\tau + 4R^2}{\lambda_{\min}^2 \gamma^2} \tag{15}$$

Therefore claim *a* of Theorem 1 holds. Now we show claim *b*. With (12) and (14) we have,

$$\|\mathbf{w}^t\|^2 \leq \sum_{p=1}^{t} \lambda^p \lambda^p (2\tau + 4R^2) \leq \frac{2\tau + 4R^2}{\gamma^2}(2\tau + 4R^2) = (\frac{2\tau + 4R^2}{\gamma})^2, \text{ so}$$

$$\|\mathbf{w}^t\| \leq \frac{2\tau + 4R^2}{\gamma} \tag{16}$$

Then the resulting margin for $\mathbf{x}_r$ given by $\mathbf{w}^t$ is bounded from below.

$$\gamma_r^t = \frac{v_r \mathbf{w}^t \cdot \mathbf{u}_r}{\|\mathbf{w}^t\|} \geq \frac{\lambda_r \tau}{\|\mathbf{w}^t\|} \geq \lambda_r \gamma \frac{\tau}{2\tau + 4R^2} \tag{17}$$

Therefore claim b of Theorem 1 holds. □

Proof of Theorem 2

Proof. We show it by bounding $\|\mathbf{w}^t\|^2$ from above and below. Suppose $\{(\mathbf{x}_{i,j}, y_{i,j})\}$ is *separable* by $\mathbf{w}^*$ with margin γ, so

$$\mathbf{w}^* \cdot \mathbf{x}_{i,j} - \mathbf{w}^* \cdot \mathbf{x}_{i,l} > \gamma, \tag{18}$$

where $1 \leq i \leq m$, $1 \leq j, l \leq k$. Let us consider each updating $\mathbf{w}^t = \mathbf{w}^{t-1} + \sum_s u_s \mathbf{x}_{i,s}$ for some fixed i. According to the algorithm, we have

$$1 \leq \frac{1}{2} \sum_s \mid u_s \mid \leq \frac{k(k-1)/2}{2} < \frac{k^2}{4} \tag{19}$$

$$\sum_s u_s \mathbf{x}_{i,s} = \sum_{(j,l) \in S_t} \mathbf{x}_{i,j} - \mathbf{x}_{i,l}, \tag{20}$$

where $S_t = \{(j,l) \mid 1 \leq j, l \leq k, , \mathbf{w}^t \cdot (\mathbf{x}_{i,j} - \mathbf{x}_{i,l}) \leq \tau\} | S_t |= \frac{1}{2} \sum_s \mid u_s \mid$ for each iteration. Here $\frac{1}{2} \sum_s \mid u_s \mid$ represents the number of mistakes made by $\mathbf{w}^t$ on sentence i at the t^{th} updating. We define e_t as $e_t \equiv \frac{1}{2} \sum_s \mid u_s \mid$.

We first bound $\|\mathbf{w}^t\|^2$ from below. We have

$$\begin{aligned}
\mathbf{w}^* \cdot \mathbf{w}^t &= \mathbf{w}^* \cdot \mathbf{w}^{t-1} + \sum_s u_s \mathbf{w}^* \cdot \mathbf{x}_s \\
&= \mathbf{w}^* \cdot \mathbf{w}^{t-1} + \sum_{(j,l) \in S_{t-1}} \mathbf{w}^*(\mathbf{x}_{i,j} - \mathbf{x}_{i,l}) \\
&\geq \mathbf{w}^* \cdot \mathbf{w}^{t-1} + \sum_{(j,l) \in S_{t-1}} \gamma \\
&= \mathbf{w}^* \cdot \mathbf{w}^{t-1} + e_{t-1}\gamma \\
&\geq \sum_{p=1}^{t-1} e_p \gamma
\end{aligned}$$

$$\|\mathbf{w}^t\|^2 = \|\mathbf{w}^*\|^2 \|\mathbf{w}^t\|^2 \geq (\mathbf{w}^* \cdot \mathbf{w}^t)^2 \geq \left(\sum_{p=1}^{t-1} e_p \right)^2 \gamma^2 \tag{21}$$

Then we bound $\|\mathbf{w}^t\|^2$ from above.

$$\begin{aligned}
\|\mathbf{w}^t\|^2 &= \|\mathbf{w}^{t-1}\|^2 + 2\mathbf{w}^{t-1}\sum_s u_s\mathbf{x}_{i,s} + \left\|\sum_s u_s\mathbf{x}_{i,s}\right\|^2 \\
&= \|\mathbf{w}^{t-1}\|^2 + 2\sum_{(j,l)\in S_{t-1}} \mathbf{w}^{t-1}\cdot(\mathbf{x}_{i,j} - \mathbf{x}_{i,l}) + \left\|\sum_s u_s\mathbf{x}_{i,s}\right\|^2 \\
&\le \|\mathbf{w}^{t-1}\|^2 + 2e_{t-1}\tau + \left\|\sum_s u_s\mathbf{x}_{i,s}\right\|^2 \\
&\le \|\mathbf{w}^{t-1}\|^2 + 2e_{t-1}\tau + 4e_{t-1}^2R^2 \\
&\le 2\sum_{p=1}^{t-1} e_p\tau + 4\sum_{p=1}^{t-1} e_p^2R^2 \qquad (22)
\end{aligned}$$

Combining (21) and (22), we have

$$\left(\sum_{p=1}^{t} e_p\right)^2 \gamma^2 \le 2\sum_{p=1}^{t} e_p\tau + 4\sum_{p=1}^{t} e_p^2R^2 \qquad (23)$$

Therefore, the number of mistakes made in the first t updates is

$$\begin{aligned}
\sum_{p=1}^{t} e_p &\le \frac{2\sum_{p=1}^t e_p\tau}{\sum_{p=1}^t e_p\gamma^2} + \frac{4\sum_{p=1}^t e_p^2R^2}{\sum_{p=1}^t e_p\gamma^2} \\
&\le \frac{2\tau}{\gamma^2} + \frac{4(k^2/4)\sum_{p=1}^t e_pR^2}{\sum_{p=1}^t e_p\gamma^2} \\
&= \frac{k^2R^2 + 2\tau}{\gamma^2} \qquad (24)
\end{aligned}$$

Since $\forall p$, $1 \le e_p$, thus $t \le \sum_{p=1}^t e_p$. Therefore Algorithm 2 stops after t updates, where

$$t \le \sum_{p=1}^{t} e_p \le \frac{k^2R^2 + 2\tau}{\gamma^2}$$

Therefore claim a and b of Theorem 2 hold. For claim c, we use the same technique as in the proof of Theorem 1, and we have

$$\gamma^t \ge \gamma\frac{\tau}{2\tau + k^2R^2}$$

□

Acknowledgments

We thank Michael Collins for help on the parse reranking data, and Anoop Sarkar, Franz Josef Och, and other members of the SMT team of the JHU Workshop 2003 for providing the MT data set. We also thank anonymous reviewers for useful comments.

Notes

1. By best we mean the parse that is the most close to the gold standard.
2. In Collins & Duffy (2002), the tree kernel is used as features on all tree segments. Here we use the feature set used in Collins (2000), and Shen and Joshi (2003).
3. f-score = 2 × precision × recall/(precision+recall).
4. The result of the algorithm depends on the order in which the training samples are used.

References

Charniak, E. (2000). A maximum-entropy-inspired parser. In J. Wiebe (Ed.), *Proceedings of the 1st meeting of the north American chapter of the association for computational linguistics* (pp. 132–139). Washington, USA: Seattle.

Collins, M., (1999). Head-Driven statistical models for natural language parsing. Ph.D. thesis, University of Pennsylvania.

Collins, M. (2000). Discriminative reranking for natural language parsing. In P. Langley (Ed.), *Proceedings of the 17th International Conference on Machine Learning* (pp. 175–182). Standord, CA, USA: Morgan Kaufmann.

Collins, M. (2004). Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In H. Bunt, J. Carroll, & G. Satta (Eds.), *New developments in parsing technology*. Kluwer Academic Publishers.

Collins, M., & Duffy N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In E. Charniak, & D. Lin (Eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*(pp. 263–270). Philadelphia, PA, USA.

Crammer, K. & Singer, Y. (2003). A family of additive online algorithms for category ranking. *Journal of Machine Learning Research, 3:Feb*, 1025–1058.

Crammer, K. & Singer, Y. (2001). PRanking with ranking. In Z. G. Thomas, G. Dietterich, & Suzanna Becker (Eds.), *Proceedings of the 15th Annual Conference Neural Information Processing Systems* (pp. 641–647). Vancouver, British Columbia, Canada: The MIT Press.

Crammer, K. & Singer, Y. (2002). On the learnability and design of output codes for multiclass problems. *Machine Learning, 47:(2/3)*, 201–233.

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

Har-Peled, S., Roth, D., & Zimak, D. (2002). Constraint classification: A new approach to multiclass classification. In N. Cesa-Bianchi, M. Numao, & R. Reischuk (Eds.), *Proceedings of the 13th Conference on Algorithmic Learning Theory* (pp. 365–379). Lubeck, Germany: Springer-Verlag.

Harrington, E. F. (2003). Online ranking/collaborative filtering using the perceptronalgorithm. In T. Fawcett & N. Mishra (Eds.), *Proceedings of the 20th International Conference on Machine Learning* (pp. 250–257). Washington, DC, USA: AAAI Press.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large Margin Rank Boundaries for Ordinal Regression. In *Advances in Large Margin Classifiers*. (pp. 115–132). The MIT Press.

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

Li, Y., Zaragoza, H., Herbrich, R., Shawe-Taylor, J., & Kandola, J. (2002). The perceptron algorithm with uneven margins. In A. G. H. Claude Sammut (Ed.), *Proceedings of the 19th International Conference on Machine Learning* (pp. 379–386). Sydney, Australia: Morgan Kaufmann.

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

Novikoff, A. B. J. (1962). On convergence proofs on perceptrons. *The Symposium on the Mathematical Theory of Automata, 12*, 615–622.

Och, F. J. (2003). Minimum error rate training for statistical machine translation. In E. W. Hinrichs, & D. Roth (Eds.), *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)* (pp. 160–167). Sapporo, Japan.

Och, F. J., Gildea, D., Khudanpur, S., Sarkar, A., Yamada, K., Fraser, A., Kumar, S., Shen, L., Smith, D., Eng, K., Jain, V., Jin, Z., & Radev, D. (2004). A smorgasbord of features for statistical machine translation. In S. Dumais, D. Marcu, & S. Roukos (Eds.), *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 161–168). Boston, MA, USA.

Papineni, K., Roukos, S., & Ward, T. (2001). Bleu: a method for automatic evaluation of machine translation. IBM Research Report, RC22176.

Ratnaparkhi, A. (1997). A Linear Observed Time Statistical Parser based on Maximum Entropy Models. In C. Cardie, & R. Weischedel (Eds.), *Proceedings of the 2nd Conference of Empirical Methods in Natural Language Processing* (pp. 1–10). Providence, Rhode Island, USA.

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

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1997). Boosting the margin: A new explanation for the effectiveness of voting methods. In D. H. Fisher (Ed.), *Proceedings of the 14th International Conference on Machine Learning* (pp. 322–330). Nashville, Tennessee, USA: Morgan Kaufmann.

Shen, L. & Joshi, A. K. (2003). An SVM based voting algorithm with application to parse reranking. In W. Daelemans & M. Osborne (Eds.), *Proceedings of the 7th Conference on Computational Natural Language Learning* (pp. 9–16). Edmonton, Canada: Morgan Kaufmann.

Shen, L., & Joshi, A. K. (2004). Flexible margin selection for reranking with full pairwise samples. In K. Su, & J. Tsujii (Eds.), *Proceedings of the 1st International Joint Conference of Natural Language Processing* (pp. 467–474). Sanya, Hainan Island, China.

Shen, L., Sarkar, A., & Och, F. J. (2004). Discriminative Reranking for Machine Translation. In S. Dumais, D. Marcu, & S. Roukos (Eds.), *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics* (pp. 177–184). Boston, MA, USA.

Vapnik, V. N. (1998). *Statistical Learning Theory*. John Wiley.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *IRE WESCON Convention Record, part 4*.

Zhang, T. (2000). Large margin winnow methods for text categorization. In *KDD-2000 Workshop on Text Mining*(pp. 81–87). Boston, MA, USA.

Received October 8, 2003
Revised May 14, 2004
Accepted May 24, 2004