Cost-Sensitive Learning of SVM for Ranking

Jun Xu[1], Yunbo Cao[2], Hang Li[2], and Yalou Huang[1]

[1] College of Software, Nankai University, Weijin Road 94, Tianjin, China
nkxj@hotmail.com, yellow@nankai.edu.cn
[2] Microsoft Research Asia, Zhichun Road 49, Beijing, China
{yucao, hangli}@microsoft.com

Abstract. In this paper, we propose a new method for learning to rank. 'Ranking SVM' is a method for performing the task. It formulizes the problem as that of binary classification on instance pairs and performs the classification by means of Support Vector Machines (SVM). In Ranking SVM, the losses for incorrect classifications of instance pairs between different rank pairs are defined as the same. We note that in many applications such as information retrieval the negative effects of making errors between higher ranks and lower ranks are larger than making errors among lower ranks. Therefore, it is natural to bring in the idea of cost-sensitive learning to learning to rank, or more precisely, to set up different losses for misclassification of instance pairs between different rank pairs. Given a cost-sensitive loss function we can construct a Ranking SVM model on the basis of the loss function. Simulation results show that our method works better than Ranking SVM in practical settings of ranking. Experimental results also indicate that our method can outperform existing methods including Ranking SVM on real information retrieval tasks such as document search and definition search.

1 Introduction

Learning to rank is an important research topic in machine learning, because many issues in information retrieval and data mining can be formalized as ranking problems. For example, in information retrieval, the user types a query, and the system calculates the relevance scores of documents with respect to the query and returns the documents in descending order of the relevance scores. The relevance scores can be calculated by a ranking function constructed with machine learning.

Machine learning approaches to ranking have been proposed. Ranking SVM [14] is such a method. It formulizes the learning to rank as that of learning for classifying instance pairs into two categories and trains a SVM model to perform the task.

We note that in many applications such as information retrieval, the negative effects of making errors between higher ranks and lower ranks are larger than making errors among lower ranks[5]. In Ranking SVM, however, the losses for incorrect classifications of instance pairs between different rank pairs are defined as the same (i.e., only correctly ranked and incorrectly ranked are considered). Therefore, to make Ranking SVM more useful in practice, it is necessary to change the formation of the loss function.

J. Fürnkranz, T. Scheffer, and M. Spiliopoulou (Eds.): ECML 2006, LNAI 4212, pp. 833 – 840, 2006.

In the paper, we propose a new method for learning 'Ranking SVM' based on cost-sensitive learning. Specifically, we set different losses for misclassification of instance pairs between different rank pairs. We find it is feasible to make a generalization of the learning algorithm of Ranking SVM such that given a cost-sensitive loss function we construct a Ranking SVM model on the basis of the loss function.

2 Related Work

The problem of learning to rank can be formulized as classification [10],[12], regression [17], or ordinal regression. The ordinal regression methods can be further classified into two groups: referred to, in this paper, as 'point-wise training' (c.f. [6], [7]) and 'pair-wise training' (c.f. [4], [14]) respectively. We consider pair-wise training in this paper. Ranking SVM [14] is a typical method of ordinal regression based on pair-wise training. It formulizes the learning to rank problem as that of learning for classifying instance pairs into two categories (correctly ranked and incorrectly ranked) and trains a SVM model to perform the task.

Cost-sensitive learning is a sub-field of supervised learning in which classifiers are constructed when different penalties (losses) are needed for different types of classification errors [9]. Recent research on cost-sensitive learning can be grouped into three categories: 1) creating particular classifiers in cost-sensitive learning (e.g., [3], [11]), 2) assigning each example to its lowest expected cost class (e.g., [8], [24]), and 3) modifying the distribution of training examples prior to performing learning (e.g., [1], [18]).

3 Learning of SVM for Ranking

Assume that a training set of labeled data is available. Each instance $(\vec{x}_i, y_i) \in R^n \times Y$ has been generated independently from an unknown distribution, where $\vec{x}$ denotes a feature vector and y denotes a label of rank. In the space of ranks $Y = \{r_1, r_2, \cdots, r_q\}$, there exists a total order between the ranks $r_q \succ r_{q-1} \succ \cdots \succ r_1$, where '$\succ$' denotes preference relationship. The goal of ranking learning is to induce a ranking function $f \in F$, which can determine preference relation between instances:

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow f(\vec{x}_i) > f(\vec{x}_j) \tag{1}$$

Herbrich et al propose transforming the learning task into that of learning for classification *on pairs of instances* (Ranking SVM)[14]. First, we assume that f is a linear function:

$$f_{\vec{w}}(\vec{x}) = \langle \vec{w}, \vec{x} \rangle, \tag{2}$$

where $\vec{w}$ denotes a vector of weights and $< \cdot, \cdot >$ stands for an inner product. Plugging (2) into (1) we obtain

$$\vec{x}_i \succ \vec{x}_j \Leftrightarrow \langle \vec{w}, \vec{x}_i - \vec{x}_j \rangle > 0 \tag{3}$$

Note that the relation $\vec{x}_i \succ \vec{x}_j$ between instance pairs $\vec{x}_i$ and $\vec{x}_j$ is expressed by a new vector $\vec{x}_i - \vec{x}_j$. Next, we take any instance pair and their relation to create a new vector and a new label. Let $\vec{x}^{(1)}$ and $\vec{x}^{(2)}$ denote the first and second instances, and let $y^{(1)}$ and $y^{(2)}$ denote their ranks, then we have

$$\left(\vec{x}^{(1)} - \vec{x}^{(2)}, z = \begin{cases} +1 & y^{(1)} \succ y^{(2)} \\ -1 & y^{(2)} \succ y^{(1)} \end{cases}\right) \tag{4}$$

From the given training data set *S,* we create a new training data set *S′* containing ℓ labeled vectors:

$$S' = \left\{\left(\vec{x}_i^{(1)} - \vec{x}_i^{(2)}, z_i\right)\right\}_{i=1}^{\ell} \tag{5}$$

Next, we take *S′* as classification data and construct a SVM model that can assign either positive label $z = +1$ or negative label $z = -1$ to any vector $\vec{x}^{(1)} - \vec{x}^{(2)}$.

Constructing the SVM model is equivalent to solving the following quadratic optimization problem:

$$\begin{aligned} &\min_{\vec{w}} M(\vec{w}) = \frac{1}{2}\|\vec{w}\|^2 + C\sum_{i=1}^{\ell}\xi_i \\ &\text{subject to } \xi_i \geq 0, \quad z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)}\right\rangle \geq 1 - \xi_i \quad i = 1, \cdots, \ell. \end{aligned} \tag{6}$$

Note that the optimization in (6) is equivalent to that in (7), when $\lambda = \frac{1}{2C}$ [13].

$$\min_{\vec{w}} \sum_{i=1}^{\ell}\left[1 - z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)}\right\rangle\right]_+ + \lambda\|\vec{w}\|^2, \tag{7}$$

where subscript '+' indicates positive part. The first term is the so-called 'empirical hinge loss' and the second term is a regularizer.

Suppose that $\vec{w}^*$ is the weights in the SVM solution. We utilize $\vec{w}^*$ to form a ranking function $f_{\vec{w}^*}(\vec{x}) = \left\langle \vec{w}^*, \vec{x}\right\rangle$.

In many applications such as information retrieval the negative effects of making errors between higher ranks and lower ranks are larger than making errors among lower ranks[5]; for example, usability studies show that search users usually only look at the top ranked results [22].

Let us consider an example in information retrieval. Suppose that there are seven documents to rank and there are three possible ranks: definitely relevant, possibly relevant, and not relevant, denoted as 'd', 'p', and 'n' respectively. The perfect ranking is given and two rankings 1 and 2 are supposed to be made.

Perfect ranking: d p p n n n n
Ranking 1: *p d* p n n n n
Ranking 2: d p *n p* n n n

Both ranking 1 and ranking 2 are not perfect. From practical viewpoint, the 'loss' in ranking 1 should be larger than that in ranking 2, because the error in ranking 1 is between the highest rank d and the middle rank p, while the error in ranking 2 is between the middle rank p and the lowest rank n. That is to say, a larger penalty should be added to the former. However, this is not reflected in the conventional Ranking SVM or any other existing ranking methods.

To deal with the problem, we propose a new type of SVM model for ranking, which retains different losses for different rank pairs. Specifically, we introduce penalty weights τ's into different rank pairs in the loss function in (7). That is to say, we re-formalize SVM learning problem as that of minimizing the following loss function.

$$\min_{\vec{w}} L(\vec{w}) = \sum_{i=1}^{\ell}\tau_{k(i)}\left[1 - z_i\left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)}\right\rangle\right]_+ + \lambda\|\vec{w}\|^2, \tag{8}$$

where $k(i)$ represents the type of rank pairs $y_i^{(1)}$ and $y_i^{(2)}$ with regard to $\vec{x}_i^{(1)} - \vec{x}_i^{(2)}$; $\tau_{k(i)}$ represents penalty weight for $k(i)$.

If it is to penalize errors between a rank pair, then we can assign a larger weight (larger than 1.0). Note that our method contains Ranking SVM as a special case in which all τ's equal 1.0.

In our method, instead of directly solving (8), we solve the equivalent quadratic optimization problem as described below.

$$\begin{aligned} &\min_{\vec{w}} M(\vec{w}) = \frac{1}{2}\left\|\vec{w}\right\|^2 + \sum_{j=1}^{\ell} C_{k(i)} \cdot \xi_i \\ &\text{subject to } \xi_i \geq 0, \quad z_i \left\langle \vec{w}, \vec{x}_i^{(1)} - \vec{x}_i^{(2)} \right\rangle \geq 1 - \xi_i \quad i = 1, \cdots, \ell \end{aligned} \tag{9}$$

This is because the following proposition 1 holds. Note that in (9) we use different C's for different rank pairs.

Proposition 1: *The problems in* (8) *and* (9) *are equivalent, when* $C_{k(i)} = \tau_{k(i)} / 2\lambda$.

The Lagrange dual function of problem (9) is

$$L_D = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{i'=1}^{\ell} \alpha_i \alpha_{i'} z_i z_{i'} \left\langle \vec{x}_i^{(1)} - \vec{x}_i^{(2)}, \vec{x}_{i'}^{(1)} - \vec{x}_{i'}^{(2)} \right\rangle \cdot \tag{10}$$

We maximize L_D subject to the constraints $0 \leq \alpha_i \leq C_{k(i)} \quad i = 1, \cdots, \ell$.

4 Experimental Results

We conducted three experiments: simulation, document search, and definition search.

As measures for evaluating the results of ranking methods, we used Normalized Discounted Cumulative Gain (NDCG)[16] and Mean Average Precision (MAP)[2]. They are both widely used in information retrieval. NDCG is based on the assumption that there are more than two ranks for relevance ranking while MAP is based on the assumption that there are two categories: relevant and irrelevant.

As baselines, we used Ranking SVM in all the experiments. In the document search experiment, we also compared our method with BM25 [21] and Language Model for Information Retrieval (LMIR) [20]. In the definition search experiment, we also used SVM classifier as a baseline.

One important issue for our method is to determine the values of the penalty parameters τ's. It appears to be hard to derive the ideal values of τ's analytically. In this paper we propose a heuristic method (c.f. Fig. 1) for estimating the parameter values. In experiments we used training data sets for the estimation.

4.1 Simulation Experiments

We conducted a simulation to verify the effectiveness of our method. First, we assumed that there are three ranks: r_1, r_2, and r_3, and instances in the two dimensional Euclidean space are generated according to Gaussian distributions $N(m_k, \boldsymbol{I})$. We set the centers as m_1=(0, -0.5), m_2=(0, 2), and m_3=(2, 2.5) for r_1, r_2, and r_3, respectively.

We assume that the standard deviations $\boldsymbol{I}$*'s* are represented by the 2×2 identity matrix. Next, we randomly generated n_k instances for each rank r_k (k =1, 2, 3) according to its distribution. We set n_1 = 1000, n_2 = 200, and n_3 = 100 (c.f., Fig. 2).

Method: Given a rank pair (r_1, r_2), estimate value of the penalty parameter τ for the rank pair.
$Drop = 0$, T = number of iterations
Query set $Q = \{q_1, q_2, \ldots, q_k\}$,
for $i = 1$ **to** k **do**
  Get document set $D_i = \{d_{i1}, d_{i2}, \ldots, d_{in}\}$ retrieved by q_i
  Get corresponding rank labels $L_i = \{l_{i1}, l_{i2}, \ldots, l_{in}\}$
  Create a perfect ranking for q_i and calculate NDCG@1: $M_{perfect}$
  $Drop_i = 0$
  for $j = 1$ **to** T **do**
    Randomly pick up two documents d_1 and d_2 whose labels are r_1 and r_2 respectively
    Swap d_1 and d_2 and calculate NDCG@1 for the new ranking: M_{ij}
    $Drop_i = Drop_i + (M_{perfect} - M_{ij})$
  End for
  $Drop = Drop + (Drop_i / T)$
End for
Return $Drop / k$

Fig. 1. Heuristic Method for setting the penalty parameters τ's

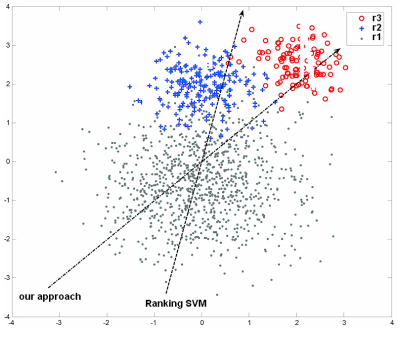

Fig. 2. Two ranking functions in simulation

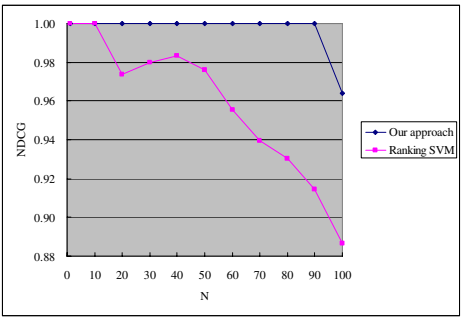

Fig. 3. NDCG curves in simulation

We applied our method and Ranking SVM to learn ranking functions from the data. In our method we set large penalty values for rank pairs r_3-r_1 and r_3-r_2, and set a small penalty value for rank pair r_2-r_1. The ranking function $f(\vec{x}) = 2.85x_1 + 3.01x_2$ was created with our method. The ranking function obtained by conventional Ranking SVM was $f(\vec{x}) = 0.53x_1 + 2.04x_2$. Fig. 2 shows the ranking functions. We calculated the NDCG values at the positions of 1, 10, 20, …, 100 and the results are given in Fig. 3. We can see that the NDCG scores of our method stay at 1.0 until when position N reaches 90. The results demonstrate that our method works better than Ranking SVM for enhancing ranking accuracy.

4.2 Experiment on Document Search

In the experiment, we tested whether our proposed methods can work well for document search. We made use of the OHSUMED collection [15]. The relevance judgments of OHSUMED are either 'd' (*definitely relevant*), 'p' (*possibly relevant*), or 'n' (*not relevant*). Rank 'n' has the largest number of documents, followed by 'p' and 'd'.

Each instance consists of a vector of features, determined by a query and a document. We adopted the feature set of [19]. Table 1 shows all the features. For

example, *tf* (term frequency), *idf* (inverse document frequency), document length, and their combinations are features. BM25 score is another feature, which is calculated using the ranking method of BM25[21]. For the baseline methods of BM25 and LMIR, we used the tool Lemur (http://www.lemurproject.org/).

We compared the performances of our method and Ranking SVM with the data through 4-fold cross-validation. The result reported in Fig. 4 are those averaged over four trials. From the figure, we see that our method outperforms baselines in terms of all the measures. The result indicates that our method is effective for the task of document retrieval.

We conducted Sign Test on the improvements of our method over BM25, LMIR, and Ranking SVM in terms of NDCG@1. The results indicate that the improvements are statistically significant (p-value < 0.05).

We analyzed the results and found that our method can indeed make better rankings than Ranking SVM. For example, for query 9, the top five documents returned by ranking SVM and our method are listed in Table 2 (the scores of NDCG@1 and NDCG@5 are also given). We note that both of the two ranking methods have two document pairs incorrectly ranked. Two (d, p) pairs reversed in the ranking by Ranking SVM and one (d, p) and one (p, n) pairs reversed by our method. However, the errors in our method are less hurtful, as they are not between the ranks d and p That is to say, our method can meet the requirement in practical ranking problems better.

4.3 Experiment on Definition Search

In this experiment, we verify whether our method can also be used in other information retrieval tasks such as search of definitions [23]. The problem is defined as retrieving and ranking definitions found from documents for a given query term. The key issue here is to rank definitions (or definition candidates) extracted from documents. The definitions can be in paragraphs or sentences and are extracted by patterns and rules. The definition candidates are categorized into three levels: 'good', 'indifferent', and 'bad', according to their goodness as definitions. This is another multiple ranks ranking problem. SVM and Ranking SVM are used for performing the task and it is empirically proved that both of the methods can work well. We tried to see whether it is possible to improve the results by using the cost sensitive learning method proposed in this paper.

We conducted 5-fold cross validation and Fig. 5 shows the results averaged over the five trials. In the experiment, we also used SVM classifier as baseline. We did not

Table 1. Features for building ranking model. $C(w, d)$ represents the raw count of word w in document d; C represents the collection; n is the number of terms in the query; $|.|$ is the size of a set; and $idf(.)$ is the inverse document frequency.

1	$\sum_{q_i \in q \cap d} \log(c(q_i, d)+1)$	2	$\sum_{q_i \in q \cap d} \log(\frac{\|C\|}{c(q_i, C)}+1)$	3	$\sum_{q_i \in q \cap d} \log(idf(q_i))$
4	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i, d)}{\|d\|}+1)$	5	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i, d)}{\|d\|} \cdot idf(q_i)+1)$	6	$\sum_{q_i \in q \cap d} \log(\frac{c(q_i, d)}{\|d\|} \cdot \frac{\|C\|}{c(q_i, C)}+1)$
7	$\log(BM25\ score)$				

Table 2. Top 5 documents ranked by Ranking SVM and our method with respect to query 9

	Ranking SVM	Our Method
Top 5 ranked docs	*p* ***d*** ***d*** *p n*	***d*** *p* ***d*** *n p*
NDCG@1	0.3333	1.0
NDCG@5	0.5453	0.6238

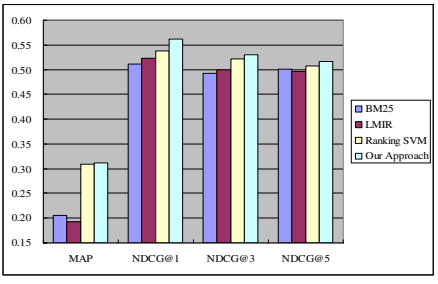

Fig. 4. Ranking accuracies in document search

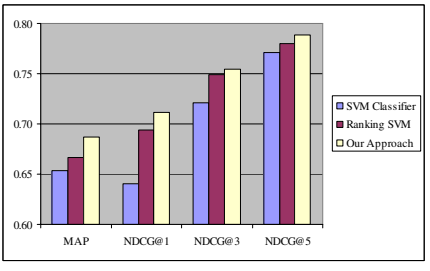

Fig. 5. Ranking accuracies in definition search

use BM25 and LMIR, because they are not suitable for conducting definition search. From Fig. 5, we conclude that our method outperforms the baseline methods of using SVM classifier and Ranking SVM. This indicates again that our method is effective for improving real ranking problems.

5 Conclusion

In the paper, we have proposed a novel cost-sensitive method to learn Support Vector Machines for ranking. We note that in many applications such as information retrieval the negative effects of making errors between higher ranks and lower ranks are much larger than making errors among lower ranks. Therefore, in learning methods for ranking, it is necessary to set up different losses for incorrectly ranking instances between different ranks. All the existing methods did not take the issue into consideration. In this paper, we take Ranking SVM as an example and have developed a new method to deal with the problem. We find that it is possible to make a generalization of the learning algorithm of Ranking SVM with a new cost-sensitive loss function. Simulation results show that our method can indeed reduce errors between higher ranks and lower ranks and thus perform better than Ranking SVM in practical settings of ranking. Experimental results verify that our method significantly outperforms Ranking SVM and other baseline methods for performing real Information Retrieval tasks.

References

1. Abe, N., Zadrozny, B., Langford, J.: An Iterative Method for Multi-class Cost-sensitive Learning. In: Proc. of 10th Inter. Conf. on KDD, Seattle, Washington, USA. (2004) 3–11
2. Baeza-Yates, R. A., Ribeiro-Neto B.: Modern Information Retrieval, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1999

3. Bradford, J., Kunz, C., Kohavi, R., Brunk, C., Brodley, C.: Pruning decision trees with misclassification costs. In: Proc. of ECML. Chemnitz, Germany. (1998) 131-136
4. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to Rank using Gradient Descent. In: Proc. of 22nd ICML. Bonn, Germany. (2005)
5. Cao, Y., Xu, J., Liu, T., Li, H., Huang, Y., Hon, H. W.: An Ordinal Regression Method for Document Retrieval. Proc. of 29th Inter. ACM SIGIR Conf., (2006), to appear
6. Chu, W., Keerthi, S.: New Approaches to Support Vector Ordinal Regression. In: Proc. of ICML, Bonn, Germany. (2005) 145–152
7. Crammer, K., Singer, Y.: PRanking with Ranking. Advances in NIPS 14, Cambridge, MA: MIT Press. (2002) 641–647
8. Domingos, P.: MetaCost : A general method for making classifiers cost sensitive. In: Proc. of the 5th KDD, San Diego, CA, USA. (1999) 155–164
9. Elkan, C.: The Foundations of Cost-Sensitive Learning. In: Proc. of the 17th Inter. Joint Conf. on Artificial Intelligence, (2001) 973–978
10. Frank, E., Hall, M.: A Simple Approach to Ordinal Classification. In: Proc. of ECML, Freiburg, Germany. (2001) 145-165
11. Geibel, P., Wysotzki, F.: Perceptron based learning with example dependent and noisy costs. In: Proc. of 12th ICML, Washington DC, USA. (2003)
12. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification: A new approach to multiclass classification and ranking. Advances in NIPS 15. (2002)
13. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: data mining, inference and prediction. Springer-Verlag. (2001)
14. Herbrich, R., Graepel, T., Obermayer, K.: Large Margin Rank Boundaries for Ordinal Regression. Advances in Large Margin Classifiers. (2000) 115-132.
15. Hersh, W. R., Buckley, C., Leone, T. J., Hickam, D. H.: OHSUMED: An interactive retrieval evaluation and new large test collection for research. In: Proc. of the 17th Inter. ACM SIGIR Conf. (1994) 192-201
16. Jarvelin, K., Kekalainen, J.: IR evaluation methods for retrieving highly relevant documents. In: Proc. of the 23rd Inter. ACM SIGIR Conf. (2000) 41-48
17. Kramer, S., Widmer, G., Pfahringer, B., & Degroeve, M. (2001). Prediction of ordinal classes using regression trees. Fundamenta Informaticae, vol. 47, 1-13.
18. Monard, M. C. & Batista, G. E. A. P. A.: Learning with Skewed Class Distribution. Advances in Logic, Artificial Intelligence and Robotics, Sao Paulo, SP. (2002) 173-180
19. Nallapati, R.: Discriminative models for information retrieval. Proc. of the 27th Inter. ACM SIGIR Conf., Sheffield, United Kingdom. (2004) 64-71
20. Ponte J. and Croft W. B.: A language model approach to information retrieval. Proc. of the Inter. ACM SIGIR Conf. (1998) 275-281.
21. Robertson, S., Hull, D. A.: The TREC-9 Filtering Track Final Report. Proc. of the 9th TREC, (2000) 25-40.
22. Spink, A., Jansen B.J., Wolfram, D., Saracevic, T.: From e-sex to e-commerce: web search changes. IEEE Computer,35(3), (2002) 107-109
23. Xu, J., Cao, Y., Li, H., Zhao, M.: Ranking Definitions with Supervised Learning Methods. Proc. of the 14th Inter. Conf. on World Wide Web, (2005) 811-819
24. Zadrozny, B., Elkan, C.: Learning and making decisions when costs and probabilities are both unknown. In: Proc. of the 7th Inter. Conf. on KDD, (2001) 204–213