

# A Confidence Voting Process for Ranking Problems based on Support Vector Machines

Tianshi Jiao<sup>\*</sup> · Jiming Peng<sup>†</sup> · Tamás Terlaky

## Abstract

In this paper, we deal with ranking problems arising from various data mining applications where the major task is to train a rank-prediction model to assign every instance a rank. We first discuss the merits and potential disadvantages of two existing popular approaches for ranking problems: the ‘Max-Wins’ voting process based on multi-class support vector machines (SVMs) and the model based on multi-criteria decision making. We then propose a confidence voting process for ranking problems based on SVMs, which can be viewed as a combination of the SVM approach and the multi-criteria decision making model. Promising numerical experiments based on the new model are reported.

**Keywords:** Multi-class Classification, Ranking, “Max-Win” Voting, Fuzzy Voting

## 1 Introduction

Given a training data set with  $m$  training instances:  $(x^c, y^c)$ ,  $c = 1, \dots, m$ , where  $x^c \in \mathbb{R}^n$  and  $y^c \in C_1, C_2, \dots, C_q$  is the associate class label, the generic classification problem refers to constructing a model to assign any unlabeled instance a class label. The ranking problem, as a special case of the multi-class classification problem, is to learn a rank-prediction model that assigns an instance a discrete rank “as close as possible” to its actual rank. Ranking problems arise from various disciplines such as biology, business and engineering [7, 8].

Various classification methods based on statistical approaches and mathematical programming have been proposed for ranking problems. The paper [21] gives a survey on

---

<sup>\*</sup>T. Jiao · T. Terlaky. Department of Computing and Software, McMaster University, Hamilton, Ontario, Canada. The research of the last author was supported by the grant #R.PG 0048923 of NESERC, the MITACS project “New Interior Point Methods and Software for Convex Conic-Linear Optimization and Their Application to Solve VLSI Circuit Layout Problems” and the Canada Researcher Chair Program.

<sup>†</sup>J. Peng. The corresponding author. Department of Industrial & Enterprise Systems Engineering, University of Illinois at Urbana-Champaign, 117 Transportation Building 104 S. Mathews Ave. Urbana, IL 61801 USA. Email: pengj@uiuc.edu

various methods for classification problems focused on applications in country risk analysis, which is a special case of ranking. Among others, Multicriteria decision-aid (MCDA), also known as preference disaggregation approach, is particularly interesting and has been widely used in numerous fields [8, 9, 21].

Since the ranking problem is a special case of the multi-class classification problem for which many methods have been proposed, we can apply these classification methods to the ranking problem directly. One popular method to solve generic classification problems is the so-called support vector machines (SVMs), originally proposed for binary classification problems [6]. Many results on extending binary SVMs to multi-class cases have been reported in the classification community [12, 13, 17]. A complete survey on SVMs can be consulted in Moguerza and Moñoz [14].

However, as we shall see in our later discussion, both the UTADIS model and the SVM-based approach suffer from some weak points when applied to ranking problems. We note there have been several works in the machine learning community on SVMs for ranking problems or ordinal regression. For example, Herbrich et' al [10, 11] proposed a method to find the decision boundaries based on support vector methods for ordinal regression. The authors of [4] discussed the applications of SVMs to ordinal regression problems arising from cancer treatment.

Different from the above-mentioned works, in this paper we propose a hybrid algorithm that can be viewed as a combination of the UTADIS model and M-SVMs. The proposed algorithm uses a two-stage strategy. In the first stage we perform the standard M-SVMs training. However, by using the ranking information for the underlying ranking problem, we are able to reduce the number of necessary binary classifiers from  $\frac{1}{2}q(q-1)$  to  $q-1$ , and thus we need to solve fewer subproblems. In the second stage we use the decision functions obtained from the SVM training in the first stage as the input data to a new optimization model, which is similar to the UTADIS model, to find the optimal decision function and decision boundaries. Like in the UTADIS model, each binary classifier (or decision function) contributes a value between 0 and 1 (called **Confidence Vote**). Finally, an instance is assigned to the class which receives the maximal number of confidence votes. For convenience, we call such a process a confidence-based voting process and the corresponding algorithm a confidence voting-based support vector machine (CVSVM).

The paper is organized as follows. In Section 2, we first give some background knowledge on the UTADIS model and M-SVMs. Then we discuss some potential limits and weak points of these two approaches and propose some remedies. In Section 3, we describe the CVSVM method for ranking problems and generic classification problems. Numerical experiments on the CVSVM method are reported in Section 4 and concluding remarks and discussions are given in Section 5.

## 2 Background Knowledge

### 2.1 The UTADIS Model

We start with a brief introduction to one of the most efficient MCDA models, the so-called utility additive discrimination model (UTADIS) [8]. In the standard UTADIS model, a utility function  $U(x)$ , represented as the additive combination of a set of criterion functions

$U_i(x_i)$ ,  $i = 1, \dots, n$ , is constructed to indicate the rank of an instance. Each such a criteria function  $U_i(x_i)$  is further assumed to be piecewise-linear and monotone. For a given instance  $x$ , if the utility function has been obtained and a set of decision boundaries  $\mu_1, \dots, \mu_{q-1}$  have been found, then we can decide the rank of  $x$  based on the following process

$$\begin{cases} \text{if } U(x) \geq \mu_1, \text{ then } x \in C_1, \\ \text{if } \mu_k \leq U(x) < \mu_{k-1}, \forall k \ 2 \leq k \leq q-1, \text{ then } x \in C_k, \\ \text{if } U(x) < \mu_{q-1}, \text{ then } x \in C_q. \end{cases} \quad (1)$$

To find the optimal utility function, we first define a distance-based loss function as demonstrated by Figure 7 in the Appendix of this paper. For any instance  $x^c$  in class  $i$ , if  $U(x^c)$  is between  $\mu_i$  and  $\mu_{i-1}$ , then the instance is correctly classified according to (1) and thus the loss is zero. Otherwise, the loss function has a positive value. For each quantitative feature (or attribute)  $x_i$ , let  $[x_{i*}, x_i^*]$  be its domain. We can discretize this domain by using a set of gridding points  $x_i^j, j = 1, \dots, r_i$  with  $x_{i*} = x_i^1$  and  $x_i^* = x_i^{r_i}$ . Let us denote that the marginal utility function values at these points by  $U_i^j$  respectively. In the UTADIS model, we further assume that the *marginal utility function*  $U_i(\cdot)$  is piecewise linear given by

$$U_i(x_i) = \frac{x_i^{j+1} - x_i}{x_i^{j+1} - x_i^j} U_i^j + \frac{x_i - x_i^j}{x_i^{j+1} - x_i^j} U_i^{j+1},$$

where  $U_i^j$  and  $U_i^{j+1}$  are to be defined. At the last step of the UTADIS model, we solve the following linear programming (LP) problem

$$\min_{\sigma, \mu, U} \sum_{c \in C_k: k < q} \sigma^+(c) + \sum_{c \in C_k: k > 1} \sigma^-(c) \quad (2a)$$

$$\text{s.t. } \sum_{i=1}^n U_i(x_i^c) - \mu_k + \sigma^+(c) \geq 0, \forall 1 \leq k \leq q-1, \forall c \in C_k, \quad (2b)$$

$$\sum_{i=1}^n U_i(x_i^c) - \mu_{k-1} - \sigma^-(c) \leq -\delta, \forall 2 \leq k \leq q, \forall c \in C_k, \quad (2c)$$

$$\mu_{k-1} - \mu_k \geq \epsilon, \quad k = 2, \dots, q-1, \quad (2d)$$

$$U_i^{j+1} - U_i^j \geq 0, \quad \forall i = 1, \dots, n, \quad \forall j = 1, \dots, r_i - 1, \quad (2e)$$

$$U_i^1 = 0, \quad \sum_{i=1}^n U_i^{r_i} = 1 \quad \forall i = 1, \dots, n, \quad (2f)$$

$$\sigma^+(c) \geq 0, \quad \sigma^-(c) \geq 0, \quad \forall c, \quad (2g)$$

for the utility function and decision boundaries. Here  $q$  is the number of classes,  $C_k$  represents the set of training instances in the  $k$ th class, and  $\epsilon$  is a pre-specified parameter to separate the decision boundaries. The first two constraints define the distance-based loss captured by  $\sigma^+$  and  $\sigma^-$  and linearly penalized in the objective function. Constraint (2e) guarantees that the normalization functions  $U_i(t)$  are monotone. By solving this LP problem, we find the optimal utility function  $U(x)$  and the decision boundaries that minimize the distance-based misclassification loss.

Note that in the above model we assume that all the preferential criteria are independent and all the criteria functions are monotone with respect to the corresponding feature. In practice, these assumptions can hardly be satisfied. This partially explains why the standard UTADIS model can not provide very satisfactory performance in many applications. As remedies to this point, several approaches based on more sophisticated optimization models have been proposed [8, 9, 16]. These models typically involve intensive computation.

## 2.2 M-SVMs

SVM is a popular method to solve generic classification problems. The algorithm was originally proposed for binary classification problems [6]. Extending the binary SVMs to multi-class cases is an active research topic in the classification community. One typical way to construct a multi-class SVM classifier is by combining several binary SVM classifiers. Three different strategies to combine the binary classifiers are: “one-against-one” [13], “one-against-all” [2] and Directed Acyclic Graph Support Vector Machines (DAGSVM) [17]. Experimental comparison of these strategies [12] indicates the performances are very close. Thus in this paper, we focus only on the “one-against-one” strategy, though the results can be extended to other strategies as well.

The “one-against-one” strategy constructs  $q(q-1)/2$  binary classifiers where each one is trained on data from two classes. To train the classifier  $C_{ij}$  separating the  $i$ -th and  $j$ -th classes, we solve the following optimization problem [19]

$$\min_{w^{ij}, b^{ij}, \xi_c^{ij}} \frac{1}{2} \|w^{ij}\|^2 + C \sum_c \xi_c^{ij}, \quad (3a)$$

$$\text{s.t. } (w^{ij})^T \phi(x^c) + b^{ij} \geq 1 - \xi_c^{ij}, \text{ if } y^c = i, \quad (3b)$$

$$(w^{ij})^T \phi(x^c) + b^{ij} \leq -1 + \xi_c^{ij}, \text{ if } y^c = j, \quad (3c)$$

$$\xi_c^{ij} \geq 0. \quad (3d)$$

Let us denote the corresponding decision functions by

$$f_{ij}(z) = (w^{ij*})^T \phi(z) + b^{ij*}. \quad (4)$$

After obtaining the decision functions, the class label for any new instance  $x$  is decided by the “Max-Wins” voting method [12] described as follows. We first calculate:

$$V_i(x) = \sum_{j=1, \dots, q, j \neq i} s(f_{ij}(x)), \quad (5)$$

where

$$s(t) = \begin{cases} 1, & \text{if } t > 0; \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

The function  $s(t)$  maps the decision function value  $f_{ij}(x)$  to  $\{0, 1\}$ . An intuitive understanding of (5) is that when classifier  $C_{ij}$  decides that a instance  $x^c$  is in class  $i$  rather

than in class  $j$  ( $f_{ij}(x^c) > 0$ ), it contributes 1 as its vote to  $V_i(x^c)$ , otherwise it contributes 0. For every class  $i$  and every instance  $x$ ,  $V_i(x)$  is the sum of all the votes collected from the binary classifiers supporting the decision that the instance  $x^c$  is in class  $i$ . Such a process is known as “voting”.

After the voting process, the class label of the instance  $x$  is decided by:

$$\text{Class Label of } i^*(x) = \operatorname{argmax}_{i=1,\dots,q} V_i(x), \quad (7)$$

i.e. instance  $x$  belongs to class  $i^*$  that collects most votes from the binary classifiers. It is worthwhile mentioning that in the above-described M-SVM model, we did not use some restrictive assumptions on the decision functions as in the UTADIS model. This allows the use of the model to various scenarios.

Unfortunately, directly applying the M-SVM to ranking problems can not provide satisfactory results. This is because, although the ranking problem can be cast as a generic multi-class classification problem, it has a specific property, i.e., the class labels (or ranks) are represented by natural numbers from 1 to  $q$ , which indicate an ordinal relation of the classes. The ranking information among classes implies that the cost of misclassification errors for ranking problems should be distance-dependent. For example, in many applications such as evaluating the financial credit of a person and ranking the risks of investments [9], a misclassification error from class  $q$  to class 1 might cause a much bigger loss than what caused by a misclassification error from class  $q$  to class  $q - 1$ . It is essential for the corresponding ranking model to be cost-sensitive. However, most existing M-SVMs for generic classification problems are not cost-sensitive since they usually treat all the misclassification errors equivalently.

## 3 Confidence-Voting-Based Support Vector Machines

### 3.1 CVSVM for Ranking Problems

In the present section, we propose a two-stage process in our confidence-voting-based method for ranking problems. In the first stage, we construct  $q - 1$  binary classifiers where the  $i$ -th classifier is trained by taking the instances from the first  $i$  classes as positive instances, and taking the rest instances as negative ones. In other words, in the  $i$ -th SVM training, we solve the following optimization problem:

$$\min_{w^i, b^i, \xi^i} \frac{1}{2} \|w^i\|^2 + C \sum_{j=1}^n \xi_j^i \quad (8a)$$

$$\text{s.t. } (w^i)^T \phi(x_j) + b^i \geq 1 - \xi_j, \text{ if } y_j \leq i, \quad (8b)$$

$$(w^i)^T \phi(x_j) + b^i \leq -1 + \xi_j, \text{ if } y_j > i, \quad (8c)$$

$$\xi_j^i \geq 0. \quad (8d)$$

Here we use the ranking information to group the classes so that the number of the necessary binary classifiers is reduced from  $q(q - 1)/2$  to  $q - 1$ . After solving problem (8) for all  $i = 1, \dots, q - 1$ , we get  $q - 1$  decision functions:

$$f_i(x) = (w^i)^T \phi(x) + b_i; \quad i = 1, \dots, q - 1. \quad (9)$$

Based on these decision functions, we prepare the input data for the next stage and store them in a matrix. For each instance  $x^c$  in the training set, we calculate the decision function values  $f_i(x^c)$ ,  $i = 1, \dots, q - 1$  of the  $q - 1$  binary classifiers, and store them into a matrix. For each decision function  $f_i(x)$ , let us define  $f_{i*} = \min\{f_i(x^c), c = 1, \dots, n\}$ ,  $f_i^* = \max\{f_i(x^c), c = 1, \dots, n\}$ . We then discretize the domain  $[f_{i*}, f_i^*]$  by a set of gridding points defined by

$$f_{i*} = f_i^1 < f_i^2 < \dots < f_i^{r_i} = f_i^*.$$

The second stage of our new algorithm is a decision process. Recall that in the so called ‘‘Max-Win’’ voting process of the M-SVM, the 0/1 voting function  $s(t)$  is used in (5b) to map the decision function value  $f_{ij}(x^c)$  into the 0/1 vote. In our new algorithm, we extend the discrete voting function  $s(t)$  to a piecewise linear monotone functions  $U_i(t)$ ,  $i = 1, \dots, q - 1$ , which maps the decision function value  $f_i(x)$  to a fuzzy vote in the interval  $[0, 1]$ . For an instance  $x$ , the summation of the fuzzy votes from the binary classifiers indicates its rank. The piecewise linear voting function  $U_i(t)$ , and the summation of the fuzzy votes are defined as follows:

$$U(x) = \sum_{i=1, \dots, q-1} U_i(f_i(x)), \quad (10a)$$

$$U_i(f) = \frac{f_i^{j+1} - f}{f_i^{j+1} - f_i^j} U_i^j + \frac{f - f_i^j}{f_i^{j+1} - f_i^j} U_i^{j+1}, \quad (10b)$$

Here  $U(x)$  corresponds to the utility function value in UTADIS model, which is the summation of all the fuzzy voting functions  $U_i(f)$ ,  $i = 1, \dots, q - 1$ , and  $U_i^j = U_i(f_i^j)$ ,  $i = 1, \dots, q - 1$ ,  $j = 1, \dots, r_i$ . For any instance  $x$ , we can use both  $U(x)$  and a set of decision boundaries to decide the rank of  $x$ , according to (1).

To find the optimal set of fuzzy voting functions  $U_i(f)$ ,  $i = 1, \dots, q - 1$  and boundary values  $\mu_1, \dots, \mu_{q-1}$ , we solve the following linear optimization problem

$$\min_{\sigma, \mu, U} \sum_{c \in C_k: k < q} \sigma^+(c) + \sum_{c \in C_k: k > 1} \sigma^-(c) \quad (11a)$$

$$\text{s.t.} \quad \sum_{i=1}^{q-1} U_i(f_i(x^c)) - \mu_k + \sigma^+(c) \geq 0, \quad \forall 1 \leq k \leq q - 1, \quad \forall c \in C_k, \quad (11b)$$

$$\sum_{i=1}^{q-1} U_i(f_i(x^c)) - \mu_{k-1} - \sigma^-(c) \leq 0, \quad \forall 2 \leq k \leq q, \quad \forall c \in C_k, \quad (11c)$$

$$\mu_{k-1} - \mu_k \geq \epsilon, \quad \forall k = 2, \dots, q - 1, \quad (11d)$$

$$U_i^{j+1} - U_i^j \geq 0, \quad \forall i = 1, \dots, q - 1, \quad \forall j = 1, \dots, r_i - 1, \quad (11e)$$

$$U_i^1 = 0, U_i^{r_i} = 1, \quad \forall i = 1, \dots, q - 1, \quad (11f)$$

$$\sigma^+(c) \geq 0, \quad \sigma^-(c) \geq 0, \quad \forall c. \quad (11g)$$

We mention that  $\epsilon$  is a prefixed parameter to separate the decision boundaries. Note that the above LP problem is different from the LP problem in (2) in two perspectives. First

we replace  $x_i^c$  in (2) by  $f_i(x^c)$ , the decision function value calculated in the first stage. Secondly the normalization constraints (2f) has been changed to (11f).

The whole process of the CVSVM algorithm can be described as follows:

### Confidence-Voting-Based Support Vector Machine

- S.1** Solve problem (8) for  $i = 1, \dots, q - 1$  to get the binary decision functions  $f_i(x)$ ;
- S.2** Solve problem (11) for the final decision function  $U(x)$  and decision boundaries  $\mu_1, \dots, \mu_{q-1}$ ;
- S.3** For any instance, use the process (1) to decide its rank.

We remark that in the first stage of the CVSVM model, the binary decision functions are built upon the general SVM training. Thus these decision functions do not depend on some restrictive assumptions such as these used in the UTADIS model. This gives rise to a great flexibility of the new model. It should be mentioned that in the second stage, we still assume that all the decision functions ( $f_i$ ) are independent and each utility function  $U_i$  is monotone. At a first glance, one might think this is a bit restrictive. However, recalling the ordinal information from all the classes in the input data set, under the assumption of disjoint classes, it is reasonable to expect that the interactions among the decision functions for different classes are very minimal and each utility function should follow some sort of ordering. This justifies the assumptions in the second stage. Another interpretation of the two-stage strategy is that we first use the binary SVM training to map the data from the input space to the decision (or kernel) space. Then we construct the decision functions and decision boundaries in the decision space based on some conditions on the decision functions.

We also point out that since the objective function in the optimization model of the second stage is distance-based, the resulting prediction model is cost-sensitive. The usage of the distance-based loss function in the decision model helps to avoid serious misclassification errors as confirmed by our numerical experiments in the next section. Finally, we note that the complexity of the algorithm is dominated by the SVM training in the first stage (which is polynomial in terms of the input data size), since the LP problem (1) in the second stage is very easy to solve. Therefore, the overall complexity of the CVSVM method is polynomial.

## 3.2 CVSVM for Generic Classification Problems

In this subsection we describe the CVSVM algorithm for generic classification problems. Recall that in the classical M-SVM approach for multi-class classification problems, we first construct a binary classifiers  $C_{ij}(x)$  associated with a decision function  $f_{ij}(x)$  to separate instances from class  $i$  and class  $j$  for all  $i \neq j$ . Then we employ the ‘‘Max-Win’’ voting process to decide to which class an instance belongs. In the CVSVM algorithm, we replace the discrete voting function  $s(t)$  in (5) by a piecewise-linear monotone voting function  $U_{ij}(f)$  that maps the decision function value  $f_{ij}(x)$  to a real value in  $[0, 1]$ . Then we estimate the confidence votes of each class by

$$V_i(x) = \sum_{j=1, \dots, q, j \neq i} U_{ij}(f_{ij}(x)). \quad (12)$$

Note that for an instance  $x$ , any binary classifier  $C_{ij}$  contributes two votes:  $v_1$  for class  $i$  and  $v_2$  for class  $j$ . In a discrete voting process, we have  $v_1 + v_2 = 1$ . It will be desirable to keep such a relation in our new fuzzy voting process, which can be characterized by the following relation

$$U_{ij}(f_{ij}(x)) + U_{ji}(f_{ji}(x)) = 1, \forall i \neq j. \quad (13)$$

Since we know in prior  $f_{ij}(x) = -f_{ji}(x)$ , the above relation holds if

$$U_{ij}(t) = 1 - U_{ji}(-t), \forall i \neq j. \quad (14)$$

Analogue to the CVSVM algorithm for ranking problems in the previous subsection, we discretize the domain that contains the values  $f_{ij}(x^c)$  by a set of gridding points satisfying

$$f_{ij}^1 = \min_{c=1, \dots, m} \{f_{ij}(x^c)\} \leq f_{ij}^2 \leq \dots \leq f_{ij}^r = \max_{c=1, \dots, m} \{f_{ij}(x^c)\}.$$

Then we solve the following LP problem

$$\min_{\xi, U} \sum_{c, j} \xi_j^c \quad (15a)$$

$$\text{s.t. } V_i(x^c) - V_j(x^c) + \xi_j^c \geq 0, \forall c \in C_i, \forall i \neq j, \quad (15b)$$

$$V_i(x^c) = \sum_{j=1, \dots, q, j \neq i} U_{ij}(f_{ij}(x^c)), \forall i = 1, \dots, q, \quad (15c)$$

$$U_{ij}^l \geq U_{ij}^{l-1}, \forall i \neq j, \forall l = 2, \dots, r, \quad (15d)$$

$$U_{ij}^l = 1 - U_{ji}^{r-l}, \forall i \neq j, \forall l = 1, \dots, r-1, \quad (15e)$$

$$U_{ij}^1 = 0, U_{ij}^r = 1, \forall i \neq j, \quad (15f)$$

$$\xi_j^c \geq 0, \quad (15g)$$

for the decision functions  $U_{ij}(f)$  and thus  $V_i(x)$ . Here we assume that the decision function value  $f_{ij}(x^c)$  falls into the interval  $[f_{ij}^l, f_{ij}^{l+1}]$ , and  $U_{ij}(f_{ij}(x))$  is calculated by:

$$U_{ij}(f_{ij}(x)) = \frac{f_{ij}^{l+1} - f_{ij}(x)}{f_{ij}^{l+1} - f_{ij}^l} U_{ij}^l + \frac{f_{ij}(x) - f_{ij}^l}{f_{ij}^{l+1} - f_{ij}^l} U_{ij}^{l+1}, \quad (16)$$

Similar to the process of ‘‘Max-Win’’ voting (7), for any instance  $x^c \in C_i$ , if  $V_i(x^c) \neq \max_{j=1, \dots, q} V_j(x^c)$ , a misclassification error occurs. We use  $\xi_j^c$  to capture these misclassification errors, which are linearly penalized in the objective function. Constraint (15 d) guarantees that the sequence  $U_{ij}^l$ ,  $l = 1, \dots, r$  is monotone. We impose the constraint (15 e) so that the voting functions satisfy the relation (14). The optimal voting functions  $U_{ij}(t)$ ,  $i \neq j$  obtained by solving problem (15) and correspondingly  $V_i(x)$  can be used to predict the class labels of unlabeled instances.

## 4 Numerical Experiments

### 4.1 Numerical Results on Ranking Problems

In this section, we first report our experiments on the CVSVM algorithm for ranking problems and compare it with the described M-SVM, the UTADIS model, the Support

Vector Regression (SVR1) [18] and the support vector based methods (denoted by SVR2) for ordinal regression [10, 11]. It should be pointed out that the algorithm proposed in [10, 11] is a slight modification of the standard SVR with a different heuristics to select the decision boundaries. The candidate methods are tested on three ranking problems including the Country Risk Classification problem from [9, 16], and two other problems from the UCI Machine Learning Repository [15]. All the experiments in this paper are done on an IBM PC with an P-IV 1.7G CPU and 256M memory using Matlab 7.0. The involved LP problems are solved by SeDuMi (<http://sedumi.mcmaster.ca/>) and we use the LIBSVM [5] for the SVM training.

Since how to select a good kernel is not the focus of the present work, we simply use the ‘‘RBF’’ kernel function [3]  $K(x^i, x^j) = \exp^{-\gamma\|x^i - x^j\|^2}$  with  $\gamma = 0.2$  as the kernel function in all the binary SVM training. For each candidate method, we use the popular 10-fold Cross Validation to test its performance and repeat the process 11 times. Each time, we try different penalty factors:  $C = 2^{-i}$ ,  $i = -2, \dots, 8$  and pick out the best prediction performance from the 11 trials as the final output.

For each test problem, we give three tables to summarize the numerical results. The first two tables are the error rate matrices of the M-SVM method and the CVSVM method, respectively, which demonstrate the effect of the distance-base loss function. In each of these two tables, the diagonal elements represent the percentages of the correctly classified instances, and the off-diagonal elements represent the percentages of the misclassified instances from class  $i$  to class  $j$ . In our experiments, the accuracy of the algorithms is calculated as follows

$$AveAccuracy = \frac{\sum_i m_i}{m}, \quad (17)$$

where  $m_i$  is the number of correctly classified instances at the  $i$ -th Validation, and  $m$  is the total number of instances. In the third table, we compare the overall accuracy of the five selected algorithms. To justify the comparison based on the overall accuracy, we also calculated the so-called Wilcoxon rank tests for the M-SVM and CVSVM methods on three test problems (see Table 4), which indicates the difference between these two algorithms can not be ignored.

The first test problem is the so-called Country Risk Classification Problem (CRCP for short). The database of this problem has two sources: the economic and political data come from the World Development Indicators (WDI) [20] database created by The World Bank. This database records 575 attributes for 207 countries. In [16], the authors selected 40 attributes of 69 countries from the 1998 and 1999 database of WDI as the instances of the Country Risk Classification problem. The country risk ranks (the class labels) of these 69 countries are provided by Standard and Poors [1]. The results are summarized in Tables 1-3. As we see from Table 3, the CVSVM method achieves the highest accuracy among all the four tested algorithms. Further, Tables 1 and 2 illustrate that, although the accuracies of both the M-SVM algorithm and the CVSVM algorithm are very close, the CVSVM algorithm was able to avoid some expensive misclassification errors. For example, the misclassification rate from class 8 to class 5 is as high as 66.60% for the M-SVM algorithm, while the CVSVM algorithm did not make such serious misclassification errors. This verifies our analysis in Section 2 that the usage of the distance-based loss function in the CVSVM ensures the model is cost-sensitive.

The second test problem is known as the Computer Hardware Database Problem (CHD for short) and consists of 209 instances from 5 classes and each instance has 6 numeric attributes. The numerical results are presented Tables 5-7. From these three tables one can easily see that the CVSVM algorithm achieves the highest accuracy (66.03%) among all the five algorithms and is more cost-sensitive.

The last ranking problem is the Auto-mpg estimation problem (AUTO) that consists of 398 instances from 6 classes and each instance has 7 attributes. The experimental results are summarized by Tables 8-10, which show that the CVSVM algorithm achieves the highest accuracy.

As we can see from the above tables, among the five selected algorithms, the M-SVM is the fastest, followed by the two SVR-based algorithms (SVR1 and SVR2). The CVSVM is always between the M-SVM and the SVR-based algorithms. Sometimes the CVSVM is faster, and sometimes it might be slower than the SVR-based algorithms based on the data structure of the underlying problem and the number of classes. The UTADIS model is the slowest. This is possibly due to the fact the discretization process might lead to a large number of variables and thus result in a large scale LP. On the other hand, since we simply use SeDuMi (a general solver for conic optimization problems) as the LP solver in our experiments, thus might not be able to explore the sparse structure of the underlying LP problem in UTADIS.

## 4.2 Numerical Results on Generic Classification Problems

We have also tested the performance of the CVSVM method for generic multi-class classification problems and compared it with the performance of M-SVM. The selected benchmark problems are from the UCI Machine Learning Repository [15] that have been used in [12] to compare the performance of various M-SVM approaches. Table 11 lists the numbers of training and testing instances, the numbers of attributes, and classes for all the test problems.

In our experiments, we still use the “RBF” kernel function [3, 12] with fixed parameter  $\gamma = 0.2$  for all binary SVM classifiers. For the data sets where both the training and testing sets are available, we simply measure the performance by using the testing data set. We then test both candidate algorithms with different penalty parameters to see the influence of the penalty parameter. For the test problems where the test data are not available, we conduct the popular 10-fold Cross Validation to evaluate its performance. We repeat such a process 11 times and each time we carry out the SVM training with different penalty factors:  $C = 2^{-i}$ ,  $i = -2, \dots, 8$ . The comparison results with the M-SVM approach with the “one-against-one” strategy are summarized by Figures 1-6.

As we can see from the above figures, the best performance of both candidate algorithms are comparable. Another interesting phenomenon, we would like to point out, is that the performance of the CVSVM method is much less sensitive to the choice of the penalty parameters. In particular, for the vowel, DNA, wine, satimage problems, the performance of the CVSVM method is quite stable with respect to various choices of the penalty parameter.

Class	1	2	3	4	5	6	7	8
1	100.00	0	0	0	0	0	0	0
2	0	100.00	0	0	0	0	0	0
3	0	35.29	58.82	5.88	0	0	0	0
4	0	0	3.57	82.14	14.29	0	0	0
5	0	0	3.03	9.09	81.82	6.06	0	0
6	0	0	0	8.33	33.33	58.33	0	0
7	0	0	0	0	33.33	0	66.67	0
8	0	0	0	0	66.67	0	33.33	0

Table 1: **M-SVM on CRCP**

Class	1	2	3	4	5	6	7	8
1	85.00	15.00	0	0	0	0	0	0
2	0	100.00	0	0	0	0	0	0
3	0	29.41	58.82	11.76	0	0	0	0
4	0	0	3.57	89.29	7.14	0	0	0
5	0	0	0	12.12	87.88	0	0	0
6	0	0	0	0	41.67	58.33	0	0
7	0	0	0	0	0	66.67	33.33	0
8	0	0	0	0	0	33.33	33.33	33.33

Table 2: **CVSVM on CRCP**

Algorithms	M-SVM	CVSVM	UTADIS	SVR1	SVR2
Accuracy	80.43	81.16	61.59	75.36	77.54
Training Time (s)	1.8572	2.6314	3.9743	3.5632	3.750

Table 3: **Comparisons on CRCP**

Problems	CRCP	CHD	AUTO
z	0.3501	0.2697	0

Table 4: **Wilcoxon Rank test: M-SVM VS CVSVM**

Class	1	2	3	4	5
1	51.61	48.39	0	0	0
2	14.75	68.85	6.56	3.28	6.56
3	0	41.38	31.03	13.79	13.79
4	0	16.00	44.00	24.00	16.00
5	0	3.17	4.76	3.17	88.89

Table 5: **M-SVM on CHD**

Class	1	2	3	4	5
1	51.61	45.16	3.23	0	0
2	9.84	63.93	19.67	4.92	1.64
3	0	17.24	51.72	24.14	6.90
4	0	16.00	28.00	52.00	4.00
5	0	0	4.76	7.94	87.30

Table 6: **CVSVM on CHD**

Algorithms	M-SVM	CVSVM	UTADIS	SVR1	SVR2
Accuracy	61.72	66.03	62.20	60.29	63.64
Training Time (s)	0.2166	0.9409	1.9737	0.5001	0.8125

Table 7: **Comparisons on CHD**

Class	1	2	3	4	5	6
1	83.02	16.98	0	0	0	0
2	13.27	70.41	14.29	2.04	0	0
3	1.28	17.95	51.28	25.64	2.56	1.28
4	0	2.60	15.58	70.13	9.09	2.60
5	0	0	1.79	16.07	55.36	26.79
6	0	0	2.78	5.56	41.67	50.00

Table 8: **M-SVM on AUTO**

Class	1	2	3	4	5	6
1	81.13	18.87	0	0	0	0
2	11.22	71.43	17.35	0	0	0
3	1.28	12.82	61.54	21.79	2.56	0
4	0	0	22.08	64.94	11.69	1.30
5	0	0	1.79	19.64	57.14	21.43
6	0	0	2.78	0	41.67	55.56

Table 9: **CVSVM on AUTO**

Algorithms	M-SVM	CVSVM	UTADIS	SVR1	SVR2
Accuracy	64.32	66.08	52.76	64.82	62.56
Training Time (s)	0.4024	2.1097	3.9421	2.1101	2.3750

Table 10: **Comparisons on AUTO**

Problem	#Training Data	#Testing Data	#Attributes	#Classes
DNA	2000	1186	180	3
Vowel	528	0	10	11
Vehicle	846	0	18	4
Wine	178	0	13	3
Segment	2310	0	19	7
Satimage	4435	2000	36	6

Table 11: Test Problems

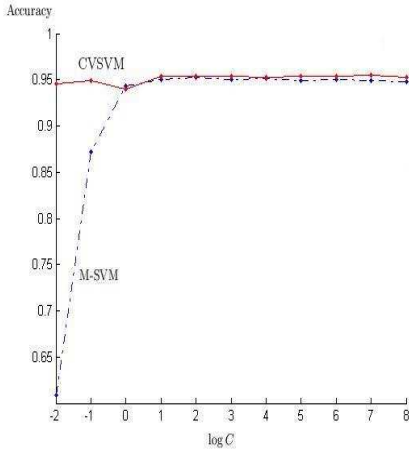


Figure 1: DNA.

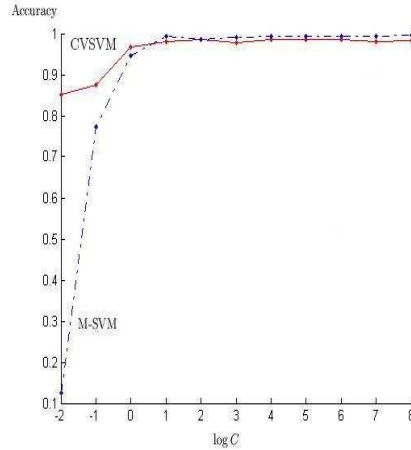


Figure 2: Vowel.

## 5 Contribution and Future Work

In this paper we have proposed a new ranking algorithm which combine the so-called UTADIS model with the M-SVM approach elegantly. The new model does not depend on such unrealistic assumptions as in the UTADIS model, and it is cost sensitive. The complexity of the whole process is polynomial in term of the input data size. Empirical results on several ranking problems show that the proposed CVSVM method achieves the highest accuracy and it is more cost-sensitive than several well-known algorithms in the literature.

The proposed algorithm has been adapted to generic multi-class classification problems. Experimental results show that the new algorithm is more robust with respect to various choices of the penalty parameter in the SVM training compared with the classical M-SVM method. Possibly, the UTADIS model helps to correct the suboptimal performance induced by a wrong selection of the penalty parameter.

There are several ways to extend the results of the present work. One possible way is to study the theoretical properties of the CVSVM method including its generalization capacity. Another extension is to consider the use of other objective functions (such as the quadratic objective functions) in the optimization problem in the second stage of the algorithm, which might lead to more cost-sensitive prediction models.

As pointed out by an anonymous referee, in the special scenario when the data set contains several small classes, our model might not work well. This phenomena has been

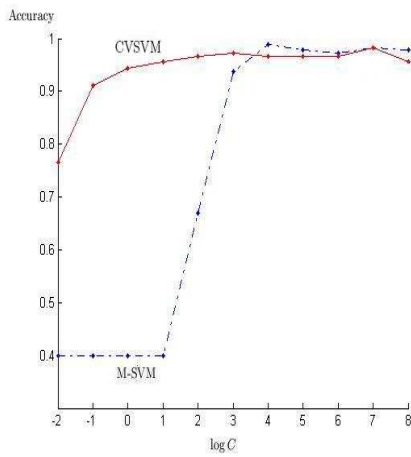


Figure 3: Wine.

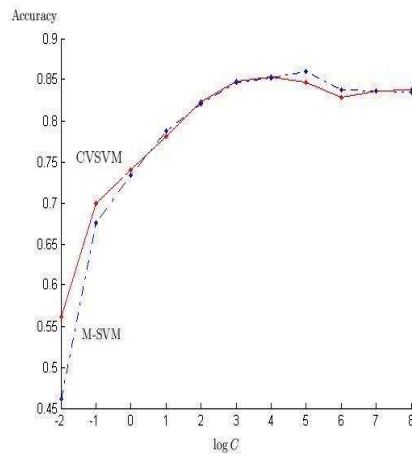


Figure 4: Vehicle.

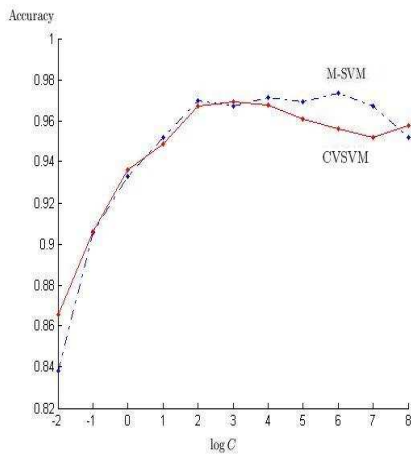


Figure 5: Segment.

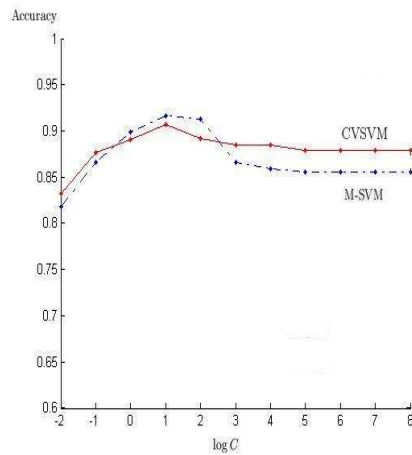


Figure 6: Satimage.

actually observed in our experiments for the CRCP that has a few very small classes. Therefore, when we applied the 10-fold cross-validation method, the training data set might not contain any instances from these small classes. This also explains why the misclassification error ratio is pretty high for these small classes as indicated in the misclassification error matrix. We also noted that in [10, 11], the training set was carefully constructed so that it includes examples from all the classes. We believe to deal with ranking problems with many small classes, extra effort is needed. This will be one of our future research topics.

**Acknowledgement** The authors would like to thank the two anonymous referees for their valuable suggestions that lead to substantial improvement of the present paper.

## References

- [1] S. Alexe, P.L. Hammer, A. Kogan, and M. A. Lejeune. (2003). “A Nonrecursive Regression Model for Country Risk Rating”, *Technical Report*, Rutgers University,

Center for Operations Research, Piscataway, New Jersey.

- [2] L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, L.D. Jackel, Y. LeCun, U.A. Müller, E. Säckinger, P. Simard, and V. Vapnik. (1994). "Comparison of Classifier Methods: A Case Study in Handwritten Digit Recognition". *ICPR*, pp. 77-87.
- [3] C. Burges. (1998). "A Tutorial on Support Vector Machines for Pattern Recognition". *Data Mining and Knowledge Discovery*, 2, pp. 121-167.
- [4] J.S. Cardoso, J.F. Pinto da Costa and M.J. Cardoso. (2005). "Modelling ordinal relations with SVMs: An application to objective aesthetic evaluation of breast cancer conservative treatment", *Neural Networks*, 18(5-6), 808-817.
- [5] C. Chang and C. Lin. (2004). "User Manual for LIBSVM", <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [6] C. Cortes and V. Vapnik. (1995). "Support-Vector Networks", *Machine Learning*, 20, pp. 273-297.
- [7] K. Crammer and Y. Singer. (2001). "Pranking with Ranking". In *Proceedings of the Conference on Neural Information Processing Systems (NIPS2001)*.
- [8] M. Doumpos and C. Zopounidis. (2002). "Multicriteria Decision Aid Classification Methods". *Vol. 73 of Applied Optimization*, Kluwer Academic Publishers, 101 Philip Drive, Norwell, MA 02061, U.S.A.
- [9] M. Doumpos, S.H. Zanakakis and C. Zopounidis.(2001). "Multicriteria preference disaggregation for classification problems with an application to global investing risk". *Decision Sciences*, 32, 333-385.
- [10] R. Herbrich, T. Graepel and K. Obermayer, (2000). "Large margin rank boundaries for ordinal regression", In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans Eds., *Advances in Large Margin Classifiers*, MIT Press, Cambridge, MA, 115-132.
- [11] R. Herbrich, T. Graepel and Obermayer, K. (1999), "Support vector learning for ordinal regression", In: *9th International Conference on Artificial Neural Networks (ICANN 99)*, 97-102.
- [12] C. Hsu and C. Lin.(2002). "A comparison of Methods for Multi-class Support Vector Machines" *IEEE Transactions on Neural Networks*, 13, 415-425.
- [13] U. Kressel. (1999). "Pairwise Classification and Support Vector machines". In B. Scholkopf, C.J.C. Burges Ed., *Advances in Kernel Methods: Support Vector Learning*, pp. 255-268, MIT Press.
- [14] J.M.Moguerza and A. Muñoz. 2006. Support Vector Machines with Applications. *Statistical Science* Vol(21), No 3, 322-336.

- [15] D. J. Newman, S. Hettich, C.L. Blake and C.J. Merz. (1998). UCI Repository of Machine Learning Databases, Irvine, CA, University of California, Department of Information and Computer Science. See also <http://www.ics.uci.edu/~mlearn/MLRepository.html>
- [16] J.M. Peng and X.J. Wang. (2005). *Generalized Utility Additive Discrimination and its Application to Country Risk Classification*, Technical Report, Department of CAS, McMaster University, Hamilton, Ontario, Canada.
- [17] J.C. Platt, N. Cristianini and J. Shawe-Taylor.(2000). “Large Margin DAGs for Multiclass Classification”. *Advances in Neural Information Processing Systems 12*, 547-553, MIT Press.
- [18] A.J. Smola and B. Schölkopf. (2004). “A Tutorial on Support Vector Regression”, *Statistics and Computing*, Volume 14, 199-222.
- [19] V. Vapnik. (1998). *Statistical Learning Theory*, Wiley, New York.
- [20] The World Bank. (2002). *World Development Indicators on CD-ROM, IBRD World Bank*, Washington, D.C.
- [21] C. Zopounidis and M. Doumpos. (2002). Multi-criteria decision aid in financial decision making: methodologies and literature review. *J. Multi-Criteria Decision Analysis*, 11, 167-186.

# Appendix

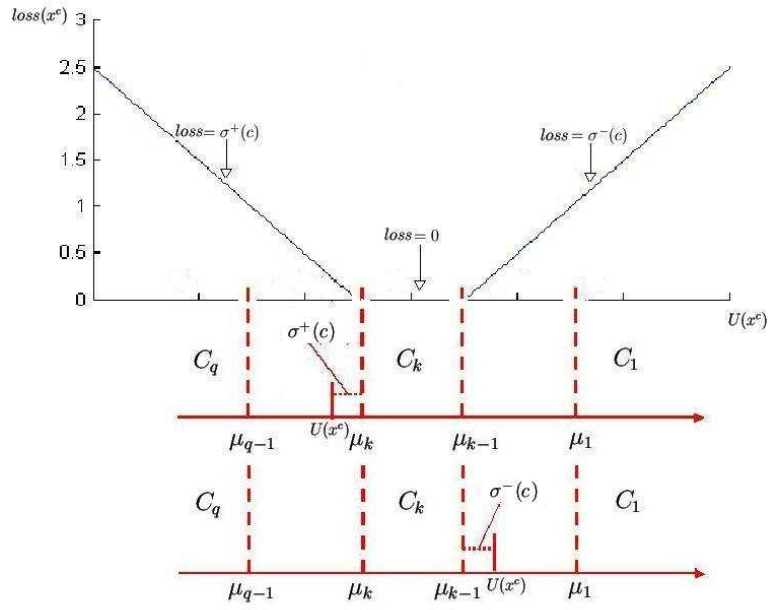


Figure 7: The loss function of the UTADIS model.