

A Novel Algorithm Model for Multi-class Classification¹

YANG Zhixia^{2,3} DENG Naiyang^{3,4}

Abstract: Multi-class classification is an important and on-going research subject in machine learning. In this paper, we propose an algorithm model for k-class multi-class classification problem based on p-class ($2 \leq p \leq k$) support vector ordinal regression machine (SVORM). A series of algorithms can be generated by selecting the different parameters p, L and the code matrix. When $p = 2$, they reduce to the popular algorithms based on 2-class SVMs. When $p = 3$, they improve K-SVCR in [1] and v-K-SVCR in [19]. The algorithms based on p-class SVORM in this algorithm model are more interesting because our preliminary numerical experiments show that then are promising. At last, some problems for further study are suggested.

Key words: Multi-class classification problem; decomposition-reconstruction; support vector ordinal regression machine; error-correcting output code

1. INTRODUCTION

Multi-class classification is an important problem in data mining and machine learning. A k -class classification problem is described as follows: Given a training

$$T = \{(x_1, y_1), \dots, (x_l, y_l)\} \in (X \times Y)^l, \quad (1)$$

where $x_i \in X = R^n$ is the input, $y_i \in Y = \{1, 2, \dots, k\}$ is the output or the class label, l is the number of training points, our task is to find a decision function $f : R^n \rightarrow Y = \{1, 2, \dots, k\}$, by which

¹ This work is supported by the Key Project of National Natural Science Foundation of China (No.10631070), the National Natural Science Foundation of China (No.10801112, No.70601033) and the China Postdoctoral Science Foundation funded project (No.20080430573)

² College of Mathematics and System Science, Xinjiang University, 830046, Urumuqi, P.R.China.

³ Academy of Mathematics and Systems Science, Chinese Academy of Sciences, 100190 Beijing, P.R.China

⁴ Academy of Mathematics and Systems Science, Chinese Academy of Sciences, 100190 Beijing, P.R.China

* Received 1 October 2008; accepted 10 November 2008

any input $\bar{x} \in R^n$ is assigned an output or a class label $f(\bar{x})$. Currently there are roughly two types of approaches to solve this problem. One is the “all-together” approach [4, 7, 13, 16, 17] that solves the k -class problem by considering all training points from all classes in one optimization formulation, while the other is the “decomposition-reconstruction” architecture approach [1-3,5,8-12,16,18,19]. The “decomposition-reconstruction” architecture approach first uses a decomposition scheme to transform a k -class problem into a series of 2-class subproblems or 3-class subproblems and get a series of classifiers, and then uses a reconstruction scheme to fuse the outputs of all classifiers for a particular input and assign it to one of the k classes. Most of the decomposition schemes are based on 2-class subproblems including “one-versus-one” method[11,12], “one-versus-rest” method [5,16] and the more general “error-correcting output code (ECOC)” method [3,8-10]. But recently, there are also some decomposition schemes based on 3-class subproblems, such as K-SVCR[1], ν -K-SVCR[19].

The algorithm proposed in this paper belongs to the “decomposition-reconstruction” architecture approach, but instead of 2-class or 3-class subproblems, p -class subproblems are constructed, where p is any integer between 2 and k . Furthermore, the p -class subproblems are solved by support vector ordinal regression machine (SVORM). Note that using SVORM to solve 3-class subproblem was proposed first time in our work [18] and latter in [2]. This idea is developed and studied in detail here.

The main contribution of this paper is to establish a very general algorithm model. It consists of the following steps: (1) For a k -class classification problem, construct several, say L , p -class classification subproblems, where $p \in [2, k]$ is an integer parameter; (2) Solve the above L subproblems by SVORM and get L classifiers $f^1(x), \Lambda, f^L(x)$; (3) For any input \bar{x} , assign its class label.

The rest of the paper is organized as follows: In section 2, SVORM is briefly introduced. Section 3 and section 4 give the multi-class algorithm model and three concrete algorithms respectively. Preliminary experimental results are presented in Section 5. Finally, in section 6 some conclusions are drawn and further research is suggested.

2. SUPPORT VECTOR ORDINAL REGRESSION MACHINE (SVORM)

2.1 SVORM Algorithm

Support Vector Ordinal Regression Machine is studied in [15]. It solves a special class of multi-class classification problems with order. For convenience, the training set here is expressed by the following way:

$$T = \{x_i^j\}_{i=1, \Lambda, j^j}^{j=1, \Lambda, p} \quad (2)$$

where $x_i^j \in R^n$ is the input, j is the output or the class label and l^j is the number of the training points in each class j . Our task is to find a real value function $g(x)$ and an orderly real sequence $b_1 \leq \Lambda \leq b_{p-1}$, and, then, construct a decision function

$$f(x) = \min_{r \in \{1, \Lambda, p\}} \{r : g(x) - b_r < 0\}, \quad (3)$$

where $b_p = +\infty$.

Using the map $\Phi(x) : R^n \rightarrow H$, where H is a Hilbert space, the maximum margin principle leads

to the following optimization problem

$$\min_{w, b, \xi^{(*)}} \frac{1}{2} \|w\|^2 + C \sum_{j=1}^p \sum_{i=1}^{l^j} (\xi_i^j + \xi_i^{*j}), \quad (4)$$

$$\text{s.t. } (w \cdot \Phi(x_i^j)) - b_j \leq -1 + \xi_i^j, \quad j=1, L, p, i=1, L, l^j, \quad (5)$$

$$(w \cdot \Phi(x_i^j)) - b_{j-1} \geq 1 - \xi_i^{*j}, \quad j=1, L, p, i=1, L, l^j, \quad (6)$$

$$\xi_i^j \geq 0, \xi_i^{*j} \geq 0, \quad j=1, L, p, i=1, L, l^j \quad (7)$$

where $w \in H$, $b = (b_1, \Lambda, b_{p-1})^T$, $b_0 = -\infty$, $b_p = +\infty$, $\xi^{(*)} = (\xi_1^1, \Lambda, \xi_{l^1}^1, \Lambda, \xi_1^p, \Lambda, \xi_{l^p}^p)$,

$\Lambda, \xi_{l^p}^p, \xi_1^{*1}, \Lambda, \xi_{l^1}^{*1}, \Lambda, \xi_1^{*p}, \Lambda, \xi_{l^p}^{*p}$, the penalty parameter $C > 0$. The dual of the problem (4)--(7) can be expressed as

$$\min_{\alpha^{(*)}} \frac{1}{2} \sum_{j,i} \sum_{j',i'} (\alpha_i^{*j} - \alpha_i^j)(\alpha_{i'}^{*j'} - \alpha_{i'}^{j'}) K(x_i^j, x_{i'}^{j'}) - \sum_{j,i} (\alpha_i^j + \alpha_i^{*j}), \quad (8)$$

$$\text{s.t. } \sum_{i=1}^{l^j} \alpha_i^j = \sum_{i=1}^{l^{j+1}} \alpha_i^{*j+1}, \quad j=1, L, p-1, \quad (9)$$

$$0 \leq \alpha_i^j, \alpha_i^{*j} \leq C, \quad j=1, L, p, i=1, L, l^j, \quad (10)$$

where $K(x, x') = (\Phi(x) \cdot \Phi(x'))$ is the kernel, $\alpha^{(*)} = (\alpha_1^1, \Lambda, \alpha_{l^1}^1, \Lambda, \alpha_1^p, \Lambda, \alpha_{l^p}^p, \alpha_1^{*1}, \Lambda, \alpha_{l^1}^{*1}, \Lambda, \alpha_1^{*p}, \Lambda, \alpha_{l^p}^{*p})^T$,

$\Lambda, \alpha_{l^1}^{*1}, \Lambda, \alpha_1^{*p}, \Lambda, \alpha_{l^p}^{*p})^T$, $\alpha_i^{*1} = 0, i=1, \Lambda, l^1, \alpha_i^p = 0, i=1, \Lambda, l^p$.

Now we are in a position to describe the SVORM algorithm:

Algorithm 1. (SVORM)

1st. Given a training set (2);

2nd. Select $C > 0$ and a kernel $K(x, x')$, solve the dual problem (8)—(10) and get its solution $\alpha^{(*)} = (\alpha_1^1, \Lambda, \alpha_{l^1}^1, \Lambda, \alpha_1^p, \Lambda, \alpha_{l^p}^p, \alpha_1^{*1}, \Lambda, \alpha_{l^1}^{*1}, \Lambda, \alpha_1^{*p}, \Lambda, \alpha_{l^p}^{*p})^T$;

3rd. Compute

$$g(x) = \sum_{j=1}^p \sum_{i=1}^{l^j} (\alpha_i^{*j} - \alpha_i^j) K(x_i^j, x); \quad (11)$$

4th. For $j=1, \Lambda, p-1$, execute the following steps:

4th 1.1 Try to choose a component $\alpha_i^j \in (0, C)$ in $\alpha^{(*)}$. If we get such i , let

$$b_j = 1 + \sum_{j'=1}^p \sum_{i'=1}^{l^{j'}} (\alpha_{i'}^{*j'} - \alpha_{i'}^{j'}) K(x_{i'}^{j'}, x_i^j), \quad (12)$$

Otherwise go to 4th 1.2 ;

4th 1.2 Try to choose a component $\alpha_i^{*j+1} \in (0, C)$ in $\alpha^{(*)}$. If we get such i , let

$$b_j = \sum_{j'=1}^p \sum_{i'=1}^{l^{j'}} (\alpha_{i'}^{*j'} - \alpha_{i'}^{j'}) K(x_{i'}^{j'}, x_{i'}^{j+1}) - 1, \quad (13)$$

Otherwise go to 4th 1.3;

4th 1.3 Let

$$b_j = \frac{1}{2}(b_j^{dn} + b_j^{up}), \quad (14)$$

where

$$b_j^{dn} = \max \{ \max_{i \in I_1^j} (g(x_i^j) + 1), \max_{i \in I_4^j} (g(x_i^{j+1}) - 1) \},$$

$$b_j^{up} = \min \{ \min_{i \in I_3^j} (g(x_i^j) + 1), \min_{i \in I_2^j} (g(x_i^{j+1}) - 1) \},$$

and

$$I_1^j = \{i \in \{1, \Lambda, l^j\} \mid \alpha_i^j = 0\}, I_2^j = \{i \in \{1, \Lambda, l^{j+1}\} \mid \alpha_i^{*j+1} = 0\},$$

$$I_3^j = \{i \in \{1, \Lambda, l^j\} \mid \alpha_i^j = C\}, I_4^j = \{i \in \{1, \Lambda, l^{j+1}\} \mid \alpha_i^{*j+1} = C\};$$

5th. If there exists $j \in \{1, \Lambda, p\}$ such that $b_j \leq b_{j-1}$, stop or go to 2;

6th. Define $b_p = +\infty$, construct the decision function

$$f(x) = \min_{r \in \{1, \Lambda, p\}} \{r : g(x) - b_r < 0\}. \quad (15)$$

2.2 The order in the training set

Intuitively and geometrically, SVORM with kernel divides the P -class inputs by $P-1$ parallel hyperplanes with fixed order in the feature space H . Note that, there must be adjacency among the P -class: class j adjoins both class $j-1$ and class $j+1$, but class $j-1$ is not adjacent to class $j+1$. It is clear that the final partition hyperplanes in the feature space H , and therefore the partition hyperplanes in the original input space R^n obtained by SVORM depend on the order in the set Y ; when this order is changed, the classifier obtained is also changed too. Thus, for a given P -class training set, many different classifiers can be obtained when different order in Y is assigned. However, can we find a reasonable classifier for any order in Y ? Generally speaking, the answer is positive if the kernel in SVORM corresponds a map $\Phi(x)$ with high dimension space H . The reason is that in high dimension space a training set with any order in Y is usually possible to be partitioned by parallel hyperplanes suitably although it is not true in low dimension space. Imaging the case with three inputs x_1, x_2 and x_3 , and by putting superscripts on them with different orders, three training sets can be constructed following the expression of (1):

$$T_1 = \{x_1^1, x_2^2, x_3^3\}, T_2 = \{x_1^2, x_2^3, x_3^1\}, T_3 = \{x_1^3, x_2^1, x_3^2\}. \quad (6)$$

Let us compare the cases between $x \in R^1$ and $x \in R^2$. For the first case $x \in R^1$, without loss of generality, assume that $x_1 < x_2 < x_3$. Obviously, corresponding to both the training sets T_2 and T_3

there is no suitable parallel hyperplanes (points) with the order required. However, for the second case $x \in R^2$. Most probably a triangle is formed by the three points x_1, x_2 and x_3 . Now corresponding to the training sets T_1, T_2 and T_3 , the suitable parallel hyperplanes (lines) with the order required does exist. This observation supports the following conclusion: Most probably for any order in Y , SVORM works well when its kernel corresponds to a map $\Phi(x)$ with high dimension space H .

In order to confirm the above conclusion, we consider the artificial example appeared in [1] and [19], which can be visualized. There are 3 classes, and each one consists of 50 inputs generated by a Gaussian distribution in R^2 :

Class I: x_1, Λ, x_{50} ; Class II: x_{51}, Λ, x_{100} ; Class III: $x_{101}, \Lambda, x_{150}$.

By putting superscripts on them, three training sets are obtained:

$$T_1 = \{x_1^1, \Lambda, x_{50}^1, x_{51}^2, \Lambda, x_{100}^2, x_{101}^3, \Lambda, x_{150}^3\}, \quad (17)$$

$$T_2 = \{x_1^2, \Lambda, x_{50}^2, x_{51}^3, \Lambda, x_{100}^3, x_{101}^1, \Lambda, x_{150}^1\}, \quad (18)$$

$$T_3 = \{x_1^3, \Lambda, x_{50}^3, x_{51}^1, \Lambda, x_{100}^1, x_{101}^2, \Lambda, x_{150}^2\}, \quad (19)$$

The partition curves obtained by SVORM are shown in Fig.1, where the parameter $C=1000$, and the kernel is the polynomial kernel $K(x, x') = ((x \cdot x') + 1)^d$ with degree $d=2,3,4$.

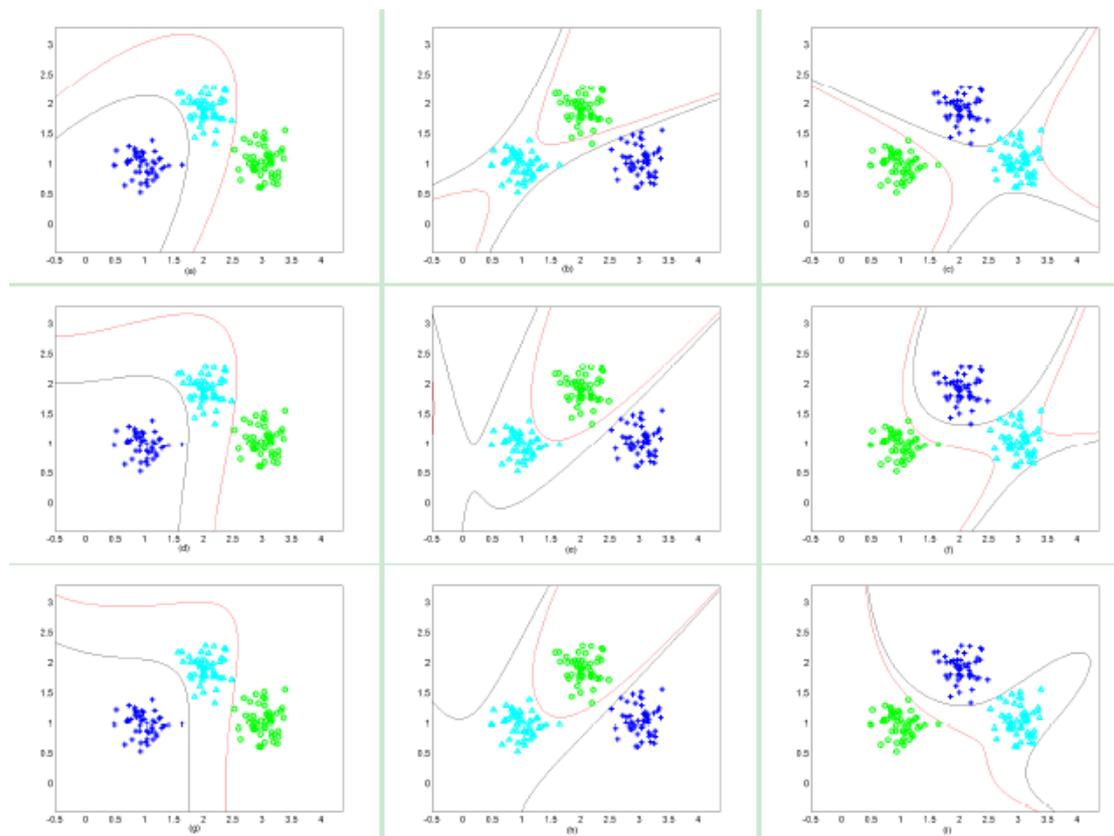


Fig.1. Polynomial kernel, $C=1000$.(a) $T_1, d = 2$; (b) $T_2, d = 2$; (c) $T_3, d = 2$; (d) $T_1, d = 3$;
(e) $T_2, d = 3$; (f) $T_3, d = 3$; (g) $T_1, d = 4$; (h) $T_2, d = 4$; (i) $T_3, d = 4$;

From Fig 1, it can be observed that for any of the degree $d=2,3,4$, the training data are separated nicely when arbitrary order of classes is arranged. This implies the reasonability to use SVORM with kernel in our Algorithm 2 (Algorithm Model) established in next section.

3. MULTI-CLASS ALGORITHM MODEL

Let us establish an algorithm model for a k -class classification problem by solving some p -class classification subproblems using SVORM. Suppose that the number of the p -class classification subproblems is L . In order to construct these subproblems, we extend and study the code matrix in [3,8-10]. A code matrix here is a matrix $M = (m_{ij})_{k \times L}$ with $m_{ij} \in \{1, \Lambda, p\}$. According to its columns, the k -class classification problem is decomposed into L p -class classification subproblems. Thus Algorithm 1 can be used to yield L classifiers $f^1(x), \Lambda, f^L(x)$.

Next question is, for a given input \bar{x} , how to assign the class it belongs to. Using the L classifiers obtained above, we get a row vector formed by the outputs of these classifiers: $F(\bar{x}) = (f^1(\bar{x}), \Lambda, f^L(\bar{x}))$. Obviously, if it happens that this row vector is the same with the i -th row in the code matrix M , the input \bar{x} should be assigned to the class i . So, in general, the input \bar{x} should be assigned to the class whose corresponding row of the matrix M is closest to the row vector $F(\bar{x})$. Here the closeness is measured by the Hamming distance defined as follows:

Definition 1. Given $a = (a_1, \Lambda, a_L)$, $b = (b_1, \Lambda, b_L) \in R^L$, the Hamming distance $d(a, b)$ between a and b is defined as the number of their different components

$$d(a, b) = |\{s \mid a_s \neq b_s\}|, \quad (20)$$

where $|\cdot|$ is the number of the elements in the set.

Now we are in a position to describe our algorithm model:

Algorithm 2. (Multi-class algorithm model)

1st. Given a training set

$$T = \{(x_1, y_1), \Lambda, (x_l, y_l)\} \in (X \times Y)^l, \quad (21)$$

where $x_i \in X = R^n$ is the input, $y_i \in Y = \{1, 2, \Lambda, k\}$ is the output or the class label, l is the number of training points;

2nd. Select the positive integers $p \in \{2, \Lambda, k\}$ and L . Construct the code matrix $M = (m_{st})_{k \times L}$, where $m_{st} \in \{1, \Lambda, p\}$;

3rd. Construct the L , p -class classification subproblems: For $t = 1, \Lambda, L$, according to the t -th column $m_t = (m_{1t}, \Lambda, m_{kt})^T$ of the code matrix M , divide the inputs x_i , $i = 1, \Lambda, L$, in the training

set (21) into p classes by assigning x_i to the class $m_{y,t}$, and the t -th subproblems is constructed;

4th. Executing Algorithm 1 to the above subproblems, we get L , p -class classifiers $f^1(x), \Lambda, f^L(x)$;

5th. Construct a row vector $F(\bar{x}) = (f^1(\bar{x}), \Lambda, f^L(\bar{x}))$. Assign a test point \bar{x} to the class

$$J = \arg \min_r d(F(\bar{x}), m_r), \quad \text{where } d(\cdot, \cdot) \text{ is Hamming distance defined by (20), } m_r \text{ is the } r\text{-th row of } M. \quad \square$$

A main advantage of the above algorithm model is its generality; It yields many algorithms by selecting the parameters p and L , and constructing the code matrix.

4. SOME ALGORITHMS

In order to evaluate the above algorithm model, concrete algorithms will be derived from it. For this purpose, it is necessary to construct some code matrices. First, let us list some properties the code matrix should have:

Property 1. Any element of code matrix is one among the integers $1, 2, \Lambda, p$. Furthermore, in each column of the code matrix, $1, 2, \Lambda, p$ should come forth at least once respectively because p -class classification subproblem will be solved.

Property 2. In the code matrix, there does not exist any two columns which lead to the same partition supersurfaces. Note that, for any subproblem corresponding to a column, a classifier is obtained by executing Algorithm 1 to it. If the i -th column and the j -th column of the code matrix is identical, the same classifier, and therefore the same partition supersurfaces, will be obtained. This case should be excluded. Furthermore, similar case happens when the i -th column and the j -th column are complementary: they divide the k classes into p groups in the same way and label these p groups with the complete inverse order. For example, when $k=p=3$, the column $(1,2,3)^T$ and $(3,2,1)^T$ are complementary. Obviously, they yield the same partition supersurfaces and therefore one of them should be deleted.

Remind that when 2-class or 3-class classifier are used, a popular strategy to construct the subproblems is "one-verse-rest" or "one-verse-rest-verse-one". When p -class classifiers are used, it is easy to see that the above Property 1 and 2 will be satisfied if the code matrix is constructed by extending the above strategy ----- keeping "the rest" to be the second class in the p -class subproblems. This leads to the following two algorithms for k -class classification problem:

Algorithm 3. (Multi-class algorithm based on 3-class SVORM)

The same with algorithm model except that $p=3$ and $L=k(k-1)/2$ are selected, and the code matrix $M = (m_{st})_{k \times L}$ is constructed by "one-verse-rest-verse-one" decomposition architecture. For example, for a 4-class problem ($k=4$), the code matrix M should be a 4×6 matrix:

$$M = \begin{pmatrix} 1 & 1 & 1 & 2 & 2 & 2 \\ 3 & 2 & 2 & 1 & 1 & 2 \\ 2 & 3 & 2 & 3 & 2 & 1 \\ 2 & 2 & 3 & 2 & 2 & 2 \end{pmatrix}.$$

Algorithm 4. (Multi-class algorithm based on 4-class SVORM)

The same with algorithm model except that $p=4$ and $L=k(k-1)(k-2)/3!$ are selected, and the code matrix $M = (m_{st})_{k \times L}$ is constructed by "one-verse-rest-verse-one-verse-one" decomposition architecture. For example, for a 4-class classification ($k=4$), the code matrix

M should be a 4×4 matrix:

$$M = \begin{pmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 3 & 3 \\ 3 & 3 & 2 & 4 \\ 4 & 4 & 4 & 2 \end{pmatrix}.$$

Surely, the above approach is rather restrictive. We are able to construct a code matrix with much more columns, which satisfies both the Property 1 and Property 2. The code matrix with the largest number L_{\max} of the columns is called the exhaustive code matrix. In order to compute L_{\max} for given k and p , consider all of the possible subproblems. Obviously all subproblems can be obtained by the following steps: (i) Divide k class into p groups; (ii) Put the p groups in order. Denoting the number of the ways to execute step (i) as $N(k,p)$ and noticing that there are $p!$ ways to execute step (ii), the number of all possible subproblems should be $N(k,p) \cdot p!$. So recalling Property 2, we have

$$L_{\max} = N(k,p) \cdot p! / 2, \quad (22)$$

where $N(k,p)$ can be computed by recursive formula:

$$N(m,1) = 1, m = 1, L, k,$$

$$N(m,2) = [C_m^1 N(m-1,2) + C_m^2 N(m-2,2) + L + C_m^{m-2} N(2,2)] / 2!,$$

$$m = 2, L, k,$$

L

$$N(m, j+1) = [C_m^1 N(m-1, j) + C_m^2 N(m-2, j) + L + C_m^{m-j} N(j, j)] / (j+1)!,$$

$$m = j, L, k, j = 0, 1, L, p-1.$$

For example, for the case $k=4, p=3$, we have $L_{\max}=18$, and the exhaustive code matrix is:

$$M = \begin{pmatrix} 2 & 2 & 2 & 1 & 3 & 3 & 1 & 3 & 3 & 1 & 3 & 3 & 2 & 2 & 2 & 1 & 1 & 1 \\ 1 & 2 & 3 & 2 & 2 & 2 & 3 & 1 & 3 & 3 & 1 & 3 & 2 & 1 & 1 & 2 & 2 & 3 \\ 3 & 1 & 3 & 3 & 1 & 3 & 2 & 2 & 2 & 3 & 3 & 1 & 1 & 2 & 3 & 2 & 3 & 2 \\ 3 & 3 & 1 & 3 & 3 & 1 & 3 & 3 & 1 & 2 & 2 & 2 & 3 & 3 & 2 & 3 & 2 & 2 \end{pmatrix}.$$

With increasing k and p , the consuming time by SVORM using the exhaustive code matrix usually becomes unacceptable. So a practical way is to use a smaller code matrix by selecting some of its columns, say L columns, from the exhaustive code matrix. In order to select the best or better L columns,

we consider the power of the code matrix to correct errors appeared in the classifiers $f^1(x), \Lambda, f^L(x)$.

Definition 2. Suppose the classifier $f^1(x), \Lambda, f^L(x)$ are obtained by Algorithm 2 when the $M = (m_{ij})_{k \times L}$ is used, we say that the code matrix is able to correct at least r errors if, for any input \bar{x} , the Algorithm Model assigns its class correctly when the number of errors in

$$(f^1(\bar{x}), \Lambda, f^L(\bar{x})). \text{ is not greater than } r.$$

Definition 3. For a code matrix $M = (m_{ij})_{k \times L}$, the minimum Hamming distance between any pair of rows of the code matrix M is called *the minimum Hamming distance of the code matrix M*.

Similar to the result about the power of a code matrix to correct errors, we have the following theorem:

Theorem 1. For a code matrix $M = (m_{ij})_{k \times L}$, where $m_{ij} \in \{1, \Lambda, p\}$, if its the minimum Hamming distance is d_{\min} , the code matrix can correct at least $[(d_{\min} - 1)/2]$ errors ($[\cdot]$ is the integral part of \cdot).

Let us return to the problem of selecting L columns from the exhaustive code matrix. The above theorem shows that the best way is to select such L columns that the minimum Hamming distance d_{\min} of the $k \times L$ matrix obtained is the largest. Following this idea but with approximation, we propose a strategy as follows:

Construction of the code matrix randomly: Select an integer L and construct several, say h , $k \times L$ matrixes M_1, Λ, M_h by selecting L columns randomly from the exhaustive code matrix. Compute their minimum Hamming distances $d_{\min}(M_1), \Lambda, d_{\min}(M_h)$. Take the code matrix M to be the one among M_1, Λ, M_h whose minimum Hamming distance is the largest.

The above strategy leads to the following algorithm:

Algorithm 5 (Multi-class algorithm based on 3-class SVORM with random)

The same with algorithm model except that $p=3$, L are selected property, and the code matrix $M = (m_{ij})_{k \times L}$ is constructed as follows: firstly construct an exhaustive code and then construct h matrixes M_1, Λ, M_h by choosing randomly L columns from the exhaustive code matrix. Select the code matrix M as

$$M = \arg \min \{d_{\min}(M_1), \Lambda, d_{\min}(M_h)\}. \quad (23)$$

5. NUMERICAL EXPERIMENT

For three concrete algorithms (Algorithm 3--5) established in Section 4, the numerical experiments results are presented for several problems from the usual UCI Repository of machine learning databases [6]. A summary of the characteristics of the selected datasets (*Iris, Wine, Glass,*

Vowel, Vehicle and Segment) is described in Table 1. Since no test data sets are provided in these six benchmark datasets, we use K -fold cross validation to evaluate the performance of the algorithms. That is, each dataset is split randomly into K subsets and one of these sets is reserved as a test set. This

process is repeated K times, where $K=5$ for the dataset *Vehicle*, while $K=10$ for the rest. In addition, following the approach in [14], the kernel matrix is reduced for *Vechile*, where 30% data of datasets were randomly chosen. The numerical experiments are implemented by using Matlab 7.0 on Intel Pentium IV 2.60GHz PC with 256MB of RAM.

Table 1. Characteristics of the selected datasets from the UCI repository

Dataset	Patterns	Features	Classes
Iris	150	4	3
Wine	178	13	3
Glass	214	4	6
Vehicle	846	18	4
Segment	210	17	7
Vowel	528	10	11

In our algorithms, the kernels and parameters are selected as follows: For data sets 'Iris' and 'Wine', the parameter $C=10$, and the polynomial kernels $K(x, x') = ((x \cdot x') + 1)^d$ with degree $d=2$ and $d=3$ are employed respectively. In all of our three algorithms, for data sets 'Glass', 'Vehicle', 'Segment' and 'Vowel', the parameter $C=1000, 2^{11}, 100, 1000$ respectively, and the Gaussian kernels $K(x, x') = \exp(-\frac{\|x - x'\|^2}{2\sigma^2})$ with $\sigma = 2, 2^{3.5}, 100, 0.5$ are employed respectively. In addition, in Algorithm 5, the parameter $L=k(k-1)/2$ is selected.

Let us make some observation from Table 2. Comparing the algorithms based on 2-class subproblems (the first 2 columns) with the ones based on 3-class subproblems (the columns 3,4,5 and 7), the latter is superior to the former. Among the algorithms based on 3-class subproblems, our algorithm (Algorithm 3 and 5) are a little better than both K-SVCR [1] and ν -K-SVCR [19]. Note that among all of the algorithms Algorithm 4 is best, where 4-class subproblems are solved. So the experiments show that it is promising to establish multi-class algorithms with higher

precision from our algorithm model by selecting proper integer p and code matrix M .

Table 2. Percentage of error on the validation set on Algorithm 3—5

Dataset	1-v-r	1-v-1	K-SVCR	ν -K-SVCR	Algorithm 3	Algorithm 4	Algorithm 5
Iris	1.33	1.33	[1.93,3.0]	1.33	1.33	1.33	--
Wine	5.6	5.6	[2.29,4,29]	3.3	2.81	--	--
Glass	35.2	36.4	[30.47, 36.35]	[32.38,36.35]	28.50	10.28	27.10
Vehicle	--	17.72	19.29	--	18.11	5.12	16.14
Segment	--	12.86	13.33	--	15.71	5.24	13.33
Vowel	39.8	38.7	--	--	7.85	--	--

1-v-r: "one versus rest", 1-v-1: "one versus one". For Iris, Wine and Glass, the results of the first four algorithms come from [19], while for Vowel, the results of the first two algorithms come from [17]. For Vehicle and Segment, the showing results are computed by ourself.

6. CONCLUSIONS

In this paper, an algorithm model is proposed which includes a series of algorithms for the multi-class classification based on p -class ($2 \leq p \leq k$) SVORM. When $p=2$, our algorithms are reduced to the multi-class algorithms based on 2-class SVMs. And when $p=3$, our algorithms are related with [1] and

ν -K-SVCR [19]. But our algorithms are not only more general but also are easier to be implemented because there is no any parameter corresponding to δ in [1] and ν_2 in [19] which require tuning carefully. Surely, more important are the algorithms based on p -class subproblems with $p \geq 4$ (maybe including $p=3$); which can be expected to improve the precision. This opinion is supported by our preliminary numerical experiment.

For the algorithm model proposed in this paper, there are still some problems deserve to be studied further. They include:

- 1st. Compare the precision and efficiency of the algorithms with different p , L and code matrix;
- 2nd. Construct a proper code matrix efficiently;
- 3rd. Improve the algorithm model further. For example, introduce a weight vector to the classifiers $f^1(x), \Lambda, f^L(x)$ because they usually have different precision. Due to the precision can be measured by the leave-one-out error, it maybe used to decide the weight vector.

REFERENCES

- [1] C Angulo, X Parra and A Catala. K-SVCR. A support vector machine for multi-class classification. *Neurocomputing*, 55, 57--77, 2003
- [2] C Angulo, F J Ruiz, L Gonzalez and J A Ortega. Multi-classification by using tri-class SVM. *Neural Procession Letters* (2006) 23:89-101
- [3] E L Allwein, R E Schapire and Y Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1, 2001, pp.113--141
- [4] K P Bennett. *Combining support vector and mathematical programming methods for classification*. in: B Scholkopf, C J C Burges and A J Smola (Eds.), *Advances in Kernel Methods: Support Vector Learning*, 1999, pp.307--326. MIT Press, Cambridge, MA
- [5] L Bottou, C Cortes, J S Denker, H Drucher, I Guyon, L D Jackel, Y LeCun, U A M\$\ddot{u}\$ller, E Sackinger, P Simard and V Vapnik. *Comparison of classifier methods: a case study in handwriting digit recognition*. in: IAPR (Ed.), *Proceedings of the International Conference on Pattern Recognition*. IEEE Computer Society Press, 1994, PP.77--82.
- [6] C Balake, C Merz. *UCI Repository of Machine Learning Database*, 1998
- [7] K Crammer, Y Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2, 2002, pp.265--292
- [8] K Crammer, Y Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47, 2002, pp.201--233
- [9] T G Dietterich, G Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research* 2, 1995, pp.263--286
- [10] J Furnkranz. *Round robin classification*. *Journal of Machine Learning Research*, 2, 2002, pp.721--747
- [11] T J Hastie, R J Ribshirani. *Classification by pairwise coupling*. in: M I Jordan, M J Kearns and S A Solla (Eds.), *Advances in Neural Information Processing Systems*, 10, 1998, pp.507--513. MIT Press, Cambridge, MA

- [12] U Kre β . *Pairwise classification and support vector machines*. in: B Scholkopf, C J C Burges and A J Smola(Eds.). *Advances in Kernel Methods: Support Vector Learning*, 1999,pp.255--268. MIT Press, Cambridge, MA
- [13] Y Lee, Y Lin and G Wahba. *Multicategory support vector machines*. *Computing Science and Statistics*, 33, 2001,pp.498--512
- [14] Y L Lee, O L Mangasarian. *RSVM: Reduced support vector machines*. *Proceedings of the First SIAM International Conference on Data Mining*, Chicago.[<ftp://ftp.cs.wisc.edu/pub/dimi/tech-reports/00-07.ps>.]
- [15] A Shashua, A Levin. Taxonomy of large margin principle algorithms for ordinal regression problems. Technical Report 2002-39. *Leivniz Center for Research School of Computer Science and Eng.*. The hebrew University of Jerusaalem
- [16] V Vapnik. *Statistical Learning Theory*. Wiley, New York,1998
- [17] J Weston, CWatkins. *Multi-class support vector machines*. CSD-TR-98-04 Royal Holloway, University of London, Egham,UK,1998
- [18] Z X Yang, N Y Deng. and Y J Tian. A multi-class classification algorithm based on ordinal regression machine. International Conference on CIMCA2005 \& IAWTIC 2005, Vienna, Austria, 2005, Vol 2, pp.810--814
- [19] P Zhong, M Fukushima. A new multi-class support vector algorithm. *Optimization Methods and Software*,21,pp.359--372,2006