MULTICLASS RIDGE-ADJUSTED SLACK VARIABLE OPTIMIZATION USING SELECTED BASIS FOR FAST CLASSIFICATION

Yinan Yu[a,c], *Konstantinos I. Diamantaras*[b], *Tomas McKelvey*[a], *S.Y. Kung*[c]

Chalmers University of Technology [a]
Gothenburg, Sweden
{yinan,tomas.mckelvey}@chalmers.se

TEI of Thessaloniki [b]
Thessaloniki, Greece
kdiamant@it.teithe.gr

Princeton University [c]
Princeton, USA
{yinany, kung}@princeton.edu

ABSTRACT

Kernel techniques for classification is especially challenging in terms of computation and memory requirement when data fall into more than two categories. In this paper, we extend a binary classification technique called Ridge-adjusted Slack Variable Optimization (RiSVO) to its multiclass counterpart where the label information encoding scheme allows the computational complexity to remain the same to the binary case. The main features of this technique are summarized as follows: (1) Only a subset of data are pre-selected to construct the basis for kernel computation; (2) Simultaneous active training set selection for all classes helps reduce complexity meanwhile improving robustness; (3) With the proposed active set selection criteria, inclusion property is verified empirically. Inclusion property means that once a pattern is excluded, it will no longer return to the active training set and therefore can be permanently removed from the training procedure. This property greatly reduce the complexity. The proposed techniques are evaluated on standard multiclass datasets MNIST, USPS, pendigits and letter which could be easily compared with existing results.

Index Terms— RiSVO, kernel, multiclass classification, large scale data, RKHS basis construction

1. INTRODUCTION

Multi-class classification problems are of growing importance in a vast range of domains, such as handwriting identification, medical applications, posture recognition, etc, where data fall into C categories for $C > 2$. In the literature, there are mainly two ways of solving such problems:

(a) Dividing the task into several binary problems using binary classifiers such as SVM [17]. Typical approaches for training and combining the binary classifiers are the One-versus-All (OvA) [12] and All-versus-All (AvA) [16] techniques.
(b) Direct extensions of binary classifiers using a global objective function. Existing techniques can be found in the literature, such as extensions of SVM [19], Neural Networks [4], Decision Trees [5], K-Nearest Neighbors [3], Naive Bayes [11], etc.

Techniques in category (a) enjoy the benefits of being intuitive and straightforward. However, more training time is required since more than one binary classifiers are needed, i.e. C for OvA and $\frac{C(C-1)}{2}$ for AvA. Nevertheless, classifiers of AvA are in general of much smaller scales than OvA and possibly result in less computations.

On the other hand, (b) has the following features:

- Compared to OvA, intrinsic structure within each class is preserved and results in more balanced classifiers.
- A global optimization problem is formulated and training is carried out simultaneously for all the classes. However, the potential drawback is that this typically results in a huge number of parameters and constraints.
- Outputs for all classes usually have a fair comparison and no normalization is needed.

Due to these properties, we are interested in finding a simultaneous technique without major increase in the complexity compared to its binary version. To this end, we propose an iterative technique extending [20] to handle multiclass case. The reduction of computational complexity is summarized in Table 1.

At each iteration k, given training size N, for $m \ll N_{S,k} \ll N_{S,k-1} \leq N$ and $N_{S,0} = N$,

Procedure	Complexity reduction
Active set select. (Sec. 2.2.1)	Kernel eval: $\mathcal{O}(N_{S,k-1}^2) \rightarrow \mathcal{O}(N_{S,k}^2)$
Inclusion prop. (Sec. 2.2.2)	Training: $\mathcal{O}(N_{S,k-1}^3) \rightarrow \mathcal{O}(N_{S,k}^3)$
RKHS basis select. (Sec. 3)	Kernel eval: $\mathcal{O}(N_{S,k}^2) \rightarrow \mathcal{O}(N_{S,k}m)$ Training: $\mathcal{O}(N_{S,k}^3) \rightarrow \mathcal{O}(m^3)$

The paper is organized as follows. First of all, the multiclassification formulation is presented in Section 2. In Section 3, we present a technique to construct the basis matrix for kernel computations by using a subset of the training data. Numerical results are shown in Section 4.

2. MULTICLASS RISVO FORMULATION

2.1. Formulation

Given training data $\boldsymbol{x}_i$, a nonlinear mapping $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^p$ ($d \ll p$) maps training data $\boldsymbol{x}_i$ to a high dimensional feature space. Kernel tricks [15] are usually used when computations

in such high dimensional space are prohibitive. That is, instead of explicitly constructing $\varphi(\boldsymbol{x}_i)$, we replace all inner products by $k(\boldsymbol{x}_i, \boldsymbol{x}_i) = \varphi(\boldsymbol{x}_i)^T\varphi(\boldsymbol{x}_i)$, where $k(\cdot,\cdot)$ is a predefined kernel function. Without ambiguity, we refer φ_i as the training data for convenience. Given training data φ_i and corresponding class label $t_i \in \{1, ..., C\}$, from the representer theorem, the class prediction $\hat{y}$ of an unseen data vector φ is defined as:

$$\hat{y} = \arg\max_c \{ \sum_{\varphi_i \in G} A(i,c)\varphi_i^T\varphi + b(c)\}, \tag{1}$$

where $A(i,c)$ is the $(i,c)^{th}$ entry of matrix $\boldsymbol{A}$ and $b(c)$ is the c^{th} element of the bias vector $\boldsymbol{b}$. Matrix $\begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{b}^T\end{bmatrix}^*$ is referred to as the "solution matrix" learned from the training process. Set G is a subset of the whole training data to construct the basis for kernel computations. It is defined in Sec. 3.

Similar to RiSVO, the multiclass version is an iterative method works as follows. Let matrix $\boldsymbol{\Phi}_M = [\varphi_{m_1}, \cdots, \varphi_{m_M}]$, $\forall \varphi_{m_i} \in M \subseteq D$, where D is the whole training set. At each iteration $k \geq 1$, the optimization problem is cast as

$$\begin{bmatrix}\boldsymbol{A}^k\\ \boldsymbol{b}^{kT}\end{bmatrix}^* = \arg\max_{\boldsymbol{A},\,\boldsymbol{b}} \frac{1}{2}\left(E(\boldsymbol{\xi}^T\boldsymbol{\xi}|\varphi_i \in S^k) + \rho\|\boldsymbol{\Phi}_G\boldsymbol{A}\|^2\right), \tag{2}$$

where $S^k \subseteq S^{k-1}$ is the active set defined in Section 2.2.1 with $S^0 = D$, ρ the ridge parameter [9] $E(\cdot)$ the expectation operator and $\|\cdot\|$ denotes the Frobenius norm. Vector $\boldsymbol{\xi}$ is called the slack vector. It is closely related to the coding scheme of the label information. Note that although $\boldsymbol{\xi}$ is k dependent, to simplify the notation, we only specify the step number explicitly for S^k, but everything else follows.

Now let us define $\boldsymbol{\xi}$. For each given training pattern $\varphi_i \in S^k$, let $\boldsymbol{T}_i$ be a diagonal matrix and the entry $\boldsymbol{T}_i(j,j) = \begin{cases}+1 & \text{if } t_i = j\\ -1 & \text{otherwise}\end{cases}$. and

$$\boldsymbol{\lambda}_i^k = \boldsymbol{T}_i(\boldsymbol{A}^{(k-1)T}\boldsymbol{\Phi}_G^T\varphi_i + \boldsymbol{b}^{(k-1)}) \tag{3}$$

The multiclass slack vector $\boldsymbol{\xi}_i$ is thus defined as:

$$\boldsymbol{\xi}_i = \boldsymbol{e} - \boldsymbol{\lambda}_i^k \tag{4}$$

where $\boldsymbol{e}$ is a all-one vector.

Let us define the kernel matrix $\boldsymbol{K}_{AB} = \boldsymbol{\Phi}_A^T\boldsymbol{\Phi}_B$ and $\boldsymbol{K}_A = \boldsymbol{\Phi}_A^T\boldsymbol{\Phi}_A$, $\boldsymbol{E}$ an all-one matrix, and N_M the cardinality of set M. Without loss of generality, we drop the iteration number k, the solution of (2) can be found by taking the derivative to zero, which results in

$$\begin{bmatrix}\boldsymbol{A}^k\\ \boldsymbol{b}^{kT}\end{bmatrix}^* = \begin{cases}\begin{bmatrix}\boldsymbol{K}_S + \rho\boldsymbol{I} & , & \boldsymbol{e}\end{bmatrix}^+\boldsymbol{\Pi}, & \text{for } G = S\\ \boldsymbol{\Omega}^+(\boldsymbol{K}_{GS} + \boldsymbol{E})\boldsymbol{\Pi}, & \text{otherwise}\end{cases} \tag{5}$$

where

$$\boldsymbol{\Omega} = [(\boldsymbol{K}_{GS} + \boldsymbol{E})\boldsymbol{K}_{SG} + \rho\boldsymbol{K}_G, \quad \boldsymbol{K}_{GS}\boldsymbol{e} + N_S\boldsymbol{e},] \tag{6}$$

and $\boldsymbol{\Pi} = [\boldsymbol{T}_i\boldsymbol{e}]_{\varphi_i \in S}$.

Therefore, the computational complexity scales as $\mathcal{O}(m^3)$, where m is the size of the basis $\boldsymbol{\Phi}_G$ in the Reproducing Kernel Hilbert Space (RKHS) (cf. Sec. 3).

2.2. Active set selection and inclusion property

Two key ideas of multiclass RiSVO algorithm are **active set selection** and the **inclusion property**, where the former improves the classification performance and the latter reduces computational complexity. Both properties are verified empirically in Sec. 4.

2.2.1. Rules for Active Set Selection

Starting from $S^0 = D$, at each iteration $k \geq 1$, we select the active set $S^k \subset S^{k-1}$ and the iterations are terminated when $S^k = S^{k-1}$. The underlying idea is to improve robustness by discarding (i) **the well classified** patterns and (ii) the "**outliers**", i.e. the selected patterns need to fall between two marginal hyperplanes. This idea is used by LASVM [2] in binary classification, whereas we extend it to a multiclass version, where the inclusion property is preserved. More precisely, for all $\varphi_i \in S^{k-1}$ and $j \in \{1, \cdots, C\}$, the selection criteria at step k are:

- $C_1^k = \{\varphi_i : \gamma_l < \boldsymbol{\lambda}_i^k(t_i) < \gamma_u\}$
- $C_2^k = \{\varphi_i : \zeta_l < E(\boldsymbol{\lambda}_i^k(j)|j \neq t_i) < \zeta_u\}$

where $\boldsymbol{\lambda}_i^k$ is defined in Equation (3) and $\boldsymbol{\lambda}_i^k(l)$ denotes the l^{th} entry. Parameters γ_l, γ_u, ζ_l and ζ_u are user defined marginal boundaries, where $\zeta_l > \gamma_l$, $\zeta_u < \gamma_u$.

Set C_1^k means that for a training pattern φ_i from class c to be selected, it has to fall between two hyperplanes $\boldsymbol{\lambda}_i^k(t_i) = \gamma_l$ and $\boldsymbol{\lambda}_i^k(t_i) = \gamma_u$ for dimension t_i. Similarly, C_2^k indicates that the averaged position of φ_i on the other $C-1$ dimensions should fall between $E(\boldsymbol{\lambda}_i^k(j)|j \neq t_i) = \zeta_l$ and $E(\boldsymbol{\lambda}_i^k(j)|j \neq t_i) = \zeta_u$. Since t_i is the true label of data φ_i, the condition $\zeta_l > \gamma_l$, $\zeta_u < \gamma_u$ indicates that we care more about the correct dimension t_i.

The active set at iteration k for multiclass RiSVO is hence defined as

$$S^k = C_1^k \cup C_2^k. \tag{7}$$

2.2.2. Inclusion Property

To guarantee and speed up convergence, we always search among $\varphi_i \in S^{k-1}$ to identify S^k. This requires the validity of the Inclusion Property. That is, even we search among all training data $\varphi_i \in D$ at every iteration k, the following property holds.

$$S^K \subseteq S^{K-1} \subsetneq \cdots \subsetneq S^1 \subsetneq S^0 \tag{8}$$

where maximum iteration K is defined by user or the minimum integer when $S^K = S^{K-1}$.

Equation (8) is called the *Inclusion Property*. **Inclusion property means that at each iteration k, the active set S^k can be identified by searching within the previous active set S^{k-1}.** It means that once a pattern is removed from the active set, it will no longer satisfy Eq. (7) at future iterations. The sufficient condition for this property to hold will be presented in the full paper due to the limited scope. In this paper, it is only verified empirically.

A summary of the implementation for Multiclass RiSVO can be found in ***Algorithm Multiclass RiSVO*** and the code will be available on the authors' website shortly.

Algorithm Multiclass RiSVO

- Initialization: let $S^0 = D$.
  Define maximum iteration K.
- **Repeat**: iteration k
  - Update $\begin{bmatrix} \boldsymbol{A}^k \\ \boldsymbol{b}^{kT} \end{bmatrix}^*$ according to Equation (5)
  - Compute $\boldsymbol{\xi}_i$ and $\boldsymbol{\lambda}_i^k$ from Equations (4) and (3)
  - Identify S^k according to Equations (7) and (8).
  - Stop if $k = K$ or $S^k = S^{k-1}$

3. RKHS BASIS SELECTION FOR LARGE SCALE DATA

In a linear classifier, we typically have the label estimation

$$\hat{y} = \arg\max_c \{\boldsymbol{w}_c^T \varphi + b(c)\}, \tag{9}$$

where $\boldsymbol{w}_c$ is the c^{th} column of the decision matrix $\boldsymbol{W}$. The relation between Equation (9) and (1) can be explained by the representer theorem [15], where the solution vector can be written as a linear combination of the training data in the Reproducing Kernel Hilbert Space (RKHS), i.e. $\boldsymbol{W} = \sum_{\varphi_i \in G} \varphi_i \boldsymbol{A}(i,:)$. In another word, the set of vectors $G = \{\varphi_1, \varphi_2, , \varphi_m\}$ form a *basis*. Ideally speaking, all the training data could be used to construct this basis. However, from Equation (5), we know that the order of the computational complexity depends on the size of the basis m. Therefore, the computational complexity can be reduced by constructing the basis using a subset of the training data. In the literature, this is often referred to as the kernel approximation problem. The review of such techniques is out of the scope. Related topics can be found in [14, 1, 7, 18, 10, 13]. In the context of finding subspace basis, we develop a selection technique based on subspace projection. By removing data with respect to small (normalized) projection distance, the subspace directions not "informative" enough are therefore rejeccted. These directions are sometimes considered dominated by noice. To fully explore the subspace concept, the technique is analyzed from three perspectives: (1) numerical, (2) basis reduction and (3) computation. Note that the "redundant" samples are only discarded for the basis construction, but still kept for training due to the rich label information.

Let us fix some notations before we proceed. Some might be different from previous sections due to different purposes. Let the basis matrix $\boldsymbol{\Phi}_m = [\varphi(\boldsymbol{x}_1), \varphi(\boldsymbol{x}_2), \cdots, \varphi(\boldsymbol{x}_m)]$. Define $\boldsymbol{K}_m = \boldsymbol{\Phi}_m^T \boldsymbol{\Phi}_m$ and $K_m(x_i, x_j) = \varphi(\boldsymbol{x}_i)^T \varphi(\boldsymbol{x}_j)$ the $(i,j)^{th}$ entry of the $m \times m$ matrix. In the empirical space, when a new pattern $\varphi(\boldsymbol{x}_r)$ is included as a part of the basis, the matrix $\boldsymbol{\Phi}_m$ becomes $\boldsymbol{\Phi}_{m+1}$ and the corresponding kernel matrix is denoted as $\boldsymbol{K}_{m+1}$. Finally, let vector $\boldsymbol{k}_m = [K_m(\boldsymbol{x}_1, \boldsymbol{x}_r) \quad \cdots \quad K_m(\boldsymbol{x}_m, \boldsymbol{x}_r)]^T$.

3.1. Numerical perspective

First of all, numerical issues are discussed. Let $\tilde{\boldsymbol{\Phi}}_m = [\varphi(\boldsymbol{x}_{g_1}), \quad \varphi(\boldsymbol{x}_{g_1}), \quad \cdots, \quad \varphi(\boldsymbol{x}_{g_m})], \quad g_i \in \mathbb{N}$ be an orthogonal basis in the intrinsic space and $\angle(\boldsymbol{A}, \boldsymbol{B})$ denote the principal angle [8] between matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

Definition 1 (Highly Redundant Pattern)**.** *Given a new pattern $\boldsymbol{x}_r$, let $\boldsymbol{\Phi}_{m+1} = \begin{bmatrix} \tilde{\boldsymbol{\Phi}}_m & \varphi(\boldsymbol{x}_r) \end{bmatrix}$. Pattern $\boldsymbol{x}_r$ is called Highly Redundant (w.r.t. $\tilde{\boldsymbol{\Phi}}_m$), if and only if $|\boldsymbol{K}_{m+1}| = 0$, i.e. $\angle(\boldsymbol{\Phi}_{m+1}, \tilde{\boldsymbol{\Phi}}_m) = 0$.* □

The definition tells us that by including a highly redundant pattern, the kernel matrix becomes rank deficient which causes computational issues. Therefore, highly redundant patterns should be discarded during basis construction. However, $|\boldsymbol{K}_{m+1}|$ in Definition 1 is generally non-zero. In order to have a meaningful numerical definition, relaxation needs to be introduced, i.e. practically speaking, $\boldsymbol{x}_r$ is considered Highly Redundant if $\boldsymbol{K}_{m+1}$ is numerically rank deficient.

3.2. Basis reduction perspective

For large data set, the purpose of basis construction is not only to avoid numerical issue, but also to reduce the size of the basis in order to scale down the computations with preserved information. To this end, Empirical Redundancy is introduced.

Definition 2. *Given a pattern $\boldsymbol{x}_r$, let $\boldsymbol{\Phi}_{m+1} = \begin{bmatrix} \tilde{\boldsymbol{\Phi}}_m | \varphi(\boldsymbol{x}_r) \end{bmatrix}$. Pattern $\boldsymbol{x}_r$ is called empirically non-redundant (w.r.t $\tilde{\boldsymbol{\Phi}}_m$), if and only if $\frac{\pi}{2} \geq \angle(\boldsymbol{\Phi}_{m+1}, \tilde{\boldsymbol{\Phi}}_m) > \frac{\pi}{2} - \theta_\epsilon$ for small $\theta_\epsilon \geq 0$.* □

This leads us to Theorem 1.

Theorem 1. *Given $\tilde{\boldsymbol{\Phi}}_m$ and pattern $\boldsymbol{x}_r$, $\boldsymbol{x}_r$ is empirically non-redundant w.r.t. $\tilde{\boldsymbol{\Phi}}_m$ iff*

$$\left| \frac{\boldsymbol{k}_{m+1}^T \tilde{\boldsymbol{K}}_m^{-1} \boldsymbol{k}_{m+1}}{K_{m+1}(\boldsymbol{x}_r, \boldsymbol{x}_r)} \right| < \epsilon. \tag{10}$$

for some small number $\epsilon \geq 0$. □

Proof. We denote $\tilde{\varphi}(\boldsymbol{x}_r)$ the projection of $\varphi(\boldsymbol{x}_r)$ onto subspace $\tilde{\boldsymbol{\Phi}}_m$, and $(\tilde{\boldsymbol{\Phi}}_m^T \tilde{\boldsymbol{\Phi}}_m)^{-1}$. Thus the angle between $\varphi(\boldsymbol{x}_r)$ and $\tilde{\varphi}(\boldsymbol{x}_r)$ is computed as

$$\theta = \arg\cos \frac{\varphi(\boldsymbol{x}_r)^T \tilde{\varphi}(\boldsymbol{x}_r)}{\|\varphi(\boldsymbol{x}_r)\| \|\tilde{\varphi}(\boldsymbol{x}_r)\|}$$

where

$$\begin{aligned} \varphi(\boldsymbol{x}_r)^T \tilde{\varphi}(\boldsymbol{x}_r) &= \boldsymbol{k}_{m+1}^T \tilde{\boldsymbol{K}}_m^{-1} \boldsymbol{k}_{m+1} \\ \|\varphi(\boldsymbol{x}_r)\| &= \sqrt{K_{m+1}(\boldsymbol{x}_r, \boldsymbol{x}_r)} \\ \|\tilde{\varphi}(\boldsymbol{x}_r)\| &= \sqrt{\boldsymbol{k}_{m+1}^T \tilde{\boldsymbol{K}}_m^{-1} \boldsymbol{k}_{m+1}} \end{aligned}$$

This gives us $\cos^2\theta = \left| \frac{\boldsymbol{k}_{m+1}^T \tilde{\boldsymbol{K}}_m^{-1} \boldsymbol{k}_{m+1}}{K_{m+1}(\boldsymbol{x}_r, \boldsymbol{x}_r)} \right|$. Since $\boldsymbol{x}_r$ empirically non-redundant, i.e. $\frac{\pi}{2} \geq \angle(\boldsymbol{\Phi}_{m+1}, \tilde{\boldsymbol{\Phi}}_m) \geq \frac{\pi}{2} - \theta_\epsilon$, we have $\cos^2\theta < \epsilon$ for some small $\epsilon > 0$. Thus it follows the conclusion. □

Due to the lack of space, the proof will be given in the full paper. The theorem tells us that as a data selection rule for basis construction, update $\tilde{\boldsymbol{\Phi}}_{m+1} = \boldsymbol{\Phi}_{m+1}$ is applicable if Equation (10) holds true; otherwise, $\tilde{\boldsymbol{\Phi}}_{m+1} = \tilde{\boldsymbol{\Phi}}_m$.

3.3. Computational perspective

However, with the increasing size of the basis, the computation of $\boldsymbol{K}_m^{-1}$ in Equation (10) becomes costly. By using *matrix inversion lemma* [6], $\boldsymbol{K}_m^{-1}$ is updated in an iterative fashion and there are only vector operations involved in the computations.

$$\boldsymbol{K}_m^{-1} = \begin{bmatrix} \boldsymbol{K}_{m-1}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\mu} \begin{bmatrix} -\boldsymbol{K}_{m-1}^{-1}\boldsymbol{k}_m \\ 1 \end{bmatrix} \left[-\boldsymbol{k}_m^T \boldsymbol{K}_{m-1}^{-1}, 1 \right]$$

where $\mu = K_m(\boldsymbol{x}_r, \boldsymbol{x}_r) - \boldsymbol{k}_m^T \tilde{\boldsymbol{K}}_{m-1}^{-1} \boldsymbol{k}_m$. The implementation of updating the basis is summarized in ***Algorithm Basis Construction*** **(B.C.)**.

Algorithm *Basis Construction* (B.C.)

- Initialization:
  Select the basis size m and a small number $0 < \epsilon \ll 1$.
  Randomly select m_0 patterns to initialize the basis matrix $\boldsymbol{\Phi}_0 = \left[\varphi(\boldsymbol{x}_{g_1}), \cdots, \varphi(\boldsymbol{x}_{g_{m_0}})\right]$.
  Compute $\boldsymbol{K}_0$, where $K_0(i,j) = K(\boldsymbol{x}_{g_i}, \boldsymbol{x}_{g_j})$.
  Let $q = 0$, $r = 1$, $\boldsymbol{P}_0 = \boldsymbol{K}_0^{-1}$.
- Step 1: For every new pattern $\boldsymbol{x}_r$, compute:

$$\boldsymbol{p}_r = \boldsymbol{P}_q \boldsymbol{k}, \quad \delta_r = \frac{\boldsymbol{k}^T \boldsymbol{p}_r}{K(\boldsymbol{x}_r, \boldsymbol{x}_r)}$$

  where $\boldsymbol{k} \in \mathbb{R}^{m_q \times 1}$ and $\boldsymbol{k}(i) = K(\boldsymbol{x}_{g_i}, \boldsymbol{x}_r)$.
- Step 2: If $|\delta_r| < \epsilon$, select the pattern $\boldsymbol{x}_r$ and update:

$$\boldsymbol{\Phi}_{q+1} = \left[\boldsymbol{\Phi}_q \varphi(\boldsymbol{x}_{g_r})\right], \; q = q+1, \; m_q = m_0 + q.$$

$$\boldsymbol{P}_q = \begin{bmatrix} \boldsymbol{P}_{q-1} & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{\mu} \begin{bmatrix} -\boldsymbol{p}_r \\ 1 \end{bmatrix} \left[-\boldsymbol{p}_r^T \quad 1 \right]$$

  where $\mu = K(\boldsymbol{x}_r, \boldsymbol{x}_r) - \boldsymbol{k}^T \boldsymbol{p}_r$ (11)

  Otherwise ignore the pattern.
- Step 3: if $m_q < m$, set $r \leftarrow r+1$ and go to Step 1; otherwise, stop.

4. EXPERIMENTAL RESULTS

4.1. Multiclass RiSVO

In this section, multiclass RiSVO is evaluated on the data sets MNIST, USPS, UCI letters and UCI pendigits. The parameters are chosen by cross-validation using the training patterns and testing error rates are presented in this section. The results and chosen parameters are summarized in Table 1 which could be compared with other existing techniques [21, 22, 23]. For example, SVM gives 2.03% testing error rate on dataset pendigits and 5.75% on letters, whereas our method achieve 0.6% and 2.56%, respectively. If we compare the number of support vectors, SVM has 1200 on pendigits and 8400 on letters, where multiclass RiSVO uses 1500 and 4000 basis vectors for representing the solution matrix. On the other hand, multiclass RiSVO does not overperform SVM on dataset MNIST and pendigits. Our guess is that RiSVO is essentially a subspace approach, which is not designed for very spase images without any preprocessing. However, if we take into consideration the computational complexity, for C classes, multiclass RiSVO still scales as $\mathcal{O}(m^3)$. The convergence of three data sets are shown in Fig. 1. The advantages with respect to the computational efficiency are summarized.

a. From computational perspective, multiclass RiSVO scales as $\mathcal{O}(m^3)$, which remains the same as binary RiSVO, i.e. the order of complexity does not depend on the number of classes.
b. Alg. *Basis Construction* (B.C.) offers a fast and numerically robust tool of reducing the basis size with preserved information.
c. By using the proposed active set selection criteria, the inclusion property is empirically verified on all datasets. That is, when a data set is excluded at step k_0, it does not come back to the active set for $k > k_0$.
d. Parallelizablility for big data computations. It fits the framework of Map-Reduce as presented in [20], where the data set is divided into M chunks and computations are carried out independently. The final result of multiclass RiSVO is obtained by various methods, such as averaging over all M processors.

Notes on selecting hyperparameters $\gamma_l, \gamma_u, \zeta_l, \zeta_u$: These parameter are chosen from $\gamma_l, \gamma_u \in \{\pm 1, \pm 1.1, \pm 1.2\}$ and $\zeta_l, \zeta_u \in \{\pm 0.5, \pm 0.6, \pm 0.7\}$. For every pattern φ, by setting $\gamma_l < \zeta_l$ and $\gamma_u > \zeta_u$, we are assignning more weights to the "correct" position t_i.

m	100	700	1000	1300	1600	1900
Random	10.12	4.20	3.57	3.38	3.0	2.99
Alg.B.C.	9.78	3.98	3.51	3.33	2.95	2.92

Table 2. Testing error comparison between random basis selection and *Alg.* Basis Construction *(B.C.)*.

4.2. RKHS basis construction

By reducing the size of the basis, m, the computations of Multiclass RiSVO is reduced, since the complexity mainly depends on m. However, a random data selection might degrade the performance. A breif comparison between random selection and *Alg.* Basis Construction *(B.C.)* in terms of classification error on data set MNIST is shown in Table 2. Note that *Alg.* Basis Construction *(B.C.)* scales as $\mathcal{O}(m^2 N_D)$.

Notes on selecting ϵ: according to Section 3.1, **ϵ shouldn't be chosen too large, i.e. $\epsilon \rightarrow 1$**. Otherwise at the next iteration, $\delta_r \rightarrow 0$ and $\frac{1}{\mu} \rightarrow +\infty$. The problem will thus become ill-conditioned. Therefore, $0 < \epsilon \ll 1$ (c.f. Algorithm *Basis Construction* (B.C.) and Sec.3.1).

Data			Parameters						P_{err}
Name	C	Training data	Kernel	Basis $\mathbf{\Phi}(G)$		RiSVO (Eq.2, 7)			
				m	ϵ	(γ_l, γ_u)	(ζ_l, ζ_u)	ρ	
MNIST[21]	10	784×60000	poly, $d = 2$	3500	0.5	$(-1.1, 1.1)$	$(-0.5, 0.5)$	0.001	2.33%
USPS[22]	10	256×3035	poly, $d = 2$	3035	-	$(-1.1, 1.1)$	$(-0.5, 0.5)$	0.001	4.81%
UCI Letters[23]	26	16×20000	rbf, $\sigma = 5$	4000	0.3	$(-1.2, 1.2)$	$(-0.7, 0.7)$	0.001	2.56%
UCI pendigits [23]	10	16×1500	rbf, $\sigma = 1$	1500	-	$(-1, 1)$	$(-0.5, 0.5)$	0.001	0.60%

Table 1. The data sets are from standard multiclass databases, which can be found in the corresponding references.

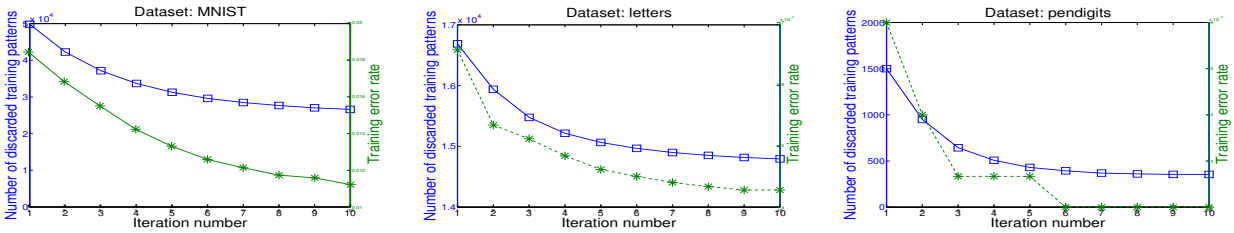

Fig. 1. Convergence of number of discarded patterns and classification error on the training set for three data bases.

5. CONCLUSION

In this paper, we present a multi-classification technique aiming for high efficiency and robustness. The technique is based on three improtant features. (1) Active set seleciton for improving robustness; (2) The empirical validity of Inclusion property for reducing complexity; and (3) Selection $G \subsetneq D$ to construct the basis for representing the solution matrix. This can be viewed as a preselection of the support vectors from a subspace projection viewpoint. As a future direction, we will provide further empirical and theoretical analysis on multiclass RiSVO.

6. REFERENCES

[1] Bach F. and Jordan M. I, Predictive low-rank decomposition for kernel methods, *in Proceeding of ICML*, 2005.

[2] Bordes A., Ertekin S., Weston J., and Bottou L., Fast Kernel Classifiers with Online and Active Learning, *Journal of Machine Learning Research*, 6:15791619, 2005.

[3] Bay S. D., *Combining Nearest Neighbor Classifiers Through Multiple Feature Subsets*, Proc. of the 15th Intl Conf on Machine Learning, pp. 37-45, Morgan Kaufmann Publishers Inc., USA, 1998

[4] Bishop C. M., *Neural Networks for Pattern Recognition* Oxford University Press, 1995.

[5] Breiman L., Friedman J., Olshen R. A., and Stone C. J., *Classification and Regression Trees*, 1984.

[6] Diamantaras K. I. and Kung SY, *Principal Component Neural Networks: Theory and Applications*, 1996.

[7] Drineas P., and Mahoney M. W., On the Nyström method for approximating a Gram matrix for improved kernel-based learning. *JMLR*, 6:2153 – 2175, 2005.

[8] Golub G.H., Van Loan, C. F., *Matrix Computation*, 3rd edition, The Johns Hopkins University Press, 1996.

[9] Hoerl A. E. and Kennard R. W. *Ridge regression: Biased estimation for nonorthogonal problems*, Technometrics, 42(1):80-86, 1970.

[10] Rahimi, A. and Recht B., Random features for large-scale kernel machines. *In Proceeding of NIPS*, 2007.

[11] Rennie J.D.M., *Improving multi-class text classification with naive Bayes*, Tech. Report, MIT, 2001.

[12] Rifkin R. and Klautau A., *In Defense of One-Vs-All Classification*, JMLR, pp. 101-141, 2004.

[13] B. Savas and I.S. Dhillon, Clustered low rank approximation of graphs in information science applications. *In SDM*, 2011.

[14] Schoölkopf B., Knirsch P, Smola A. and Burges C., *Fast Approximation of Support Vector Kernel Expansions, and an Interpretation of Clustering as Approximation in Feature Spaces*, Mustererkennung, Informatik aktuell, pp 125-132, 1998.

[15] Schölkopf, B. and Smola, A. J., *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* 1st Edition, The MIT Press, Dec. 2001.

[16] Hastie T. and Tibshirani R. *Classification by pairwise coupling*, Editors: Jordan M. I., Kearns M. J., and Solla S. A., Advances in Neural Information Processing Systems, vol. 10. The MIT Press, 1998

[17] Vapnik, V., *Statistical Learning Theory*, 1st Edition, Wiley-Interscience, September, 1998.

[18] Wang H., Hu Z. and Zhao Y., *Kernel Principal Component Analysis for Large Scale Data Set*, Lecture Notes in Computer Science, vol. 4113, pp 745-756, 2006.

[19] Weston J. and Watkins C., *Support vector machines for multiclass pattern recognition*, Proc. of the 7th European Symposium on Artificial Neural Networks, 1999.

[20] Yu Y., Diamantaras K. I., McKelvey T., and Kung S.Y., *Ridge-adjusted Slack Variable Optimization for Supervised Classification*, IEEE Intl Workshop on Machine Learning for Signal Processing, UK, 2013.

[21] *http://yann.lecun.com/exdb/mnist*

[22] *http://www.kernel-machines.org/data.html*

[23] *http://www.ics.uci.edu/mlearn/MLRepository.html*