PAPER

Learning Subspace Classification Using Subset Approximated Kernel Principal Component Analysis

Yoshikazu WASHIZAWA[†,††a)], ***Member***

SUMMARY We propose a kernel-based quadratic classification method based on kernel principal component analysis (KPCA). Subspace methods have been widely used for multiclass classification problems, and they have been extended by the kernel trick. However, there are large computational complexities for the subspace methods that use the kernel trick because the problems are defined in the space spanned by all of the training samples. To reduce the computational complexity of the subspace methods for multiclass classification problems, we extend Oja's averaged learning subspace method and apply a subset approximation of KPCA. We also propose an efficient method for selecting the basis vectors for this. Due to these extensions, for many problems, our classification method exhibits a higher classification accuracy with fewer basis vectors than does the support vector machine (SVM) or conventional subspace methods.

key words: *kernel PCA, learning subspace methods, support vector machines*

1. Introduction

Subspace classification methods have been widely used, especially for multiclass or many-class classification problems. Class-feature information compression (CLAFIC) and the multiple similarity method are the original subspace methods [1], [2], and they have been extended in various ways, such as to create the orthogonal subspace methods [3] and the mutual subspace methods [4]–[7] and their kernelizations.

CLAFIC extracts features from the training samples belonging to the target class and does not consider the differences between classes. On the other hand, learning subspace methods (LSMs) obtain subspaces using samples from both the target and nontarget class, and the features are extracted with consideration of the differences between classes [3]. Therefore, in general, the classification performance of the LSMs is higher than that of the simpler CLAFIC.

Subspace methods have also been extended by using the kernel trick, which is based on kernel principal component analysis (KPCA) [8]. CLAFIC has been extended to kernel CLAFIC methods [9]–[11], and the mutual subspace methods have also been extended [12]–[14]. These extensions exhibited higher performance than original methods. The kernel extension of LSM is expected to show higher performance than kernel CLAFIC because kernel LSM utilizes samples from both the target and nontarget class, and the features are extracted with consideration of the differences between classes as well as LSM. We focus on the kernelization of LSM in this paper in order to improve the classification accuracy, yet it inherits the advantages of the subspace methods. Figure 1 shows the relation of subspace methods.

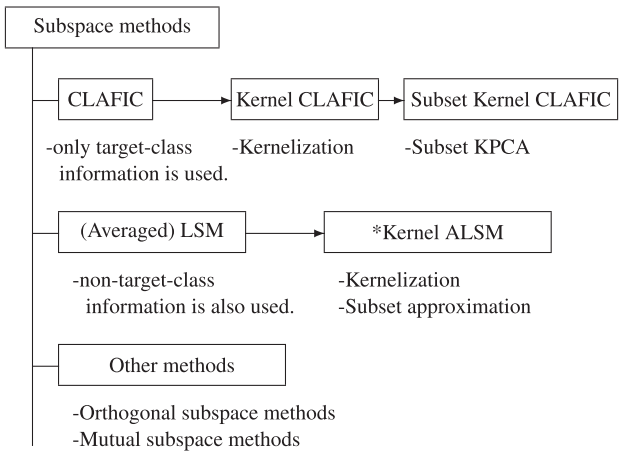

Fig. 1 Subspace methods. '*Kernel ALSM' is the scope of the paper.

In KPCA, all of the training samples are used for the basis vectors. Therefore, the size of the eigenvalue decomposition (EVD) will be equal to the number of the training samples. This computational cost can be a drawback for the subspace method, especially in big-data problems. For example, it is difficult to perform EVD for a matrix on the order of tens of thousands. Moreover, if the number of basis vectors is large, the computational cost in the test stage is also high. Therefore, it is difficult to apply the kernel trick to LSM, since LSM uses samples both in the target and the nontarget class.

Subset KPCA was proposed as a way to reduce the number of basis vectors in KPCA [15]. Subset KPCA uses a subset of the samples as the basis vectors, and all of the training vectors are used for the optimization problem to obtain the projection operator. In this case, the size of the EVD is equal to the size of the subset, and this is tunable. However, the selection of the basis vectors is based on the approximation error of the KPCA, and conventional clustering techniques are used. It is thus not suitable for classification problems as we discuss in Sect. 3.2.

In this paper, we use the subset approximation of

Manuscript received August 21, 2015.
Manuscript revised December 14, 2015.
Manuscript publicized January 25, 2016.
[†]The author is with The Univeristy of Electro-Communications, Chofu-shi, 182–8585 Japan.
[††]The author is with Brain Science Institute, RIKEN, Wako-shi, 351–0198 Japan.
a) E-mail: washizawa@uec.ac.jp
DOI: 10.1587/transinf.2015EDP7334

KPCA [15] to extend the average learning subspace method (ALSM) [3]. By combining these two approaches, we are able to obtain a classification method that has a limited number of basis vectors, yet it inherits the advantages of the subspace methods. Furthermore, we propose a new method for selecting the basis vector that are used for the classification, and we also propose an extension of the reweighting rule. We compare their accuracy in an experiment and present the results in Sect. 5.

The remainder of the paper is organized as follows. We review the ALSM, KPCA, and the subset approximation of KPCA in Sect. 2. We describe the proposed methods for classification and the selection of basis vectors selection method in Sect. 3. We compare the proposed method with the other classification methods in Sect. 4. We explain the experimental results of a handwritten-digit classification problem and multiclass classification problems using open benchmarks, in Sect. 5, and we present our conclusion in Sect. 6.

2. Existing Methods

2.1 Average Learning Subspace Method

Let $\mathcal{X}_c = \{\boldsymbol{x}_1^c, \boldsymbol{x}_2^c, \ldots, \boldsymbol{x}_{N_c}^c\}$ be a set of training samples of a class c ($c = 1, \ldots, C$), where N_c is the number of samples belonging to class c. Subspace methods including the ALSM measure the similarity between a vector $\boldsymbol{x}$ and the class c by using the projection norm onto a subspace of the class c,

$$g_c(\boldsymbol{x}) = \|\boldsymbol{U}_c^\top \boldsymbol{x}\|^2, \tag{1}$$

where the column vectors of $\boldsymbol{U}_c$ form an orthonormal basis of the subspace. The sample $\boldsymbol{x}$ is classified into that class whose projection norm $g_c(\boldsymbol{x})$ is maximum. Therefore, finding the optimal subspaces, i.e., the optimal $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_C$, for classification is the objective of the learning. Note that $\boldsymbol{U}_c$ is not a unique representation of the subspace; however, the projection norm Eq. (1) does not depend on the selection of $\boldsymbol{U}_c$.

The ALSM iteratively obtains the subspace, and CLAFIC, which is PCA without the centering for each class is used as an initial subspace. PCA is obtained from the EVD of the correlation matrix of the class c, $\boldsymbol{R}_c = \sum_{i=1}^{N_c} \boldsymbol{x}_i^c (\boldsymbol{x}_i^c)^\top$. Once the initial subspaces have been obtained, we find the misclassified samples from the training sets,

$$\begin{aligned}\mathcal{E}_c = \{\boldsymbol{x} | (&\boldsymbol{x} \text{ belongs the class } c) \\ &\cap (\boldsymbol{x} \text{ is classified to the other class})\}\end{aligned} \tag{2}$$

$$\begin{aligned}\mathcal{S}_c = \{\boldsymbol{x} | (&\boldsymbol{x} \text{ does not belongs the class } c) \\ &\cap (\boldsymbol{x} \text{ is classified to the class } c)\}.\end{aligned} \tag{3}$$

We call $\mathcal{E}_c$ the enhancement set and $\mathcal{S}_c$ the suppression set, since the samples in $\mathcal{E}_c$ should be enhanced and the samples in $\mathcal{S}_c$ should be suppressed. Thus, the subspace is obtained from the modified correlation matrix,

Algorithm 1 Average learning subspace method (ALSM)

Require: Training set $\mathcal{X}_1, \ldots, \mathcal{X}_C$, Hyper-parameters, r: dimension of subspaces, α, β: strength of enhancement and suppression, T: number of maximum iterations

Ensure: Basis of subspaces: $\boldsymbol{U}_1, \ldots, \boldsymbol{U}_C$, and functions $g_c(\boldsymbol{x})$, $c = 1, \ldots, C$

```
# Obtain the initial subspace by CLAFIC
for c = 1 to C do
    Obtain the correlation matrix R_c = Σ_{i=1}^{N_c} x_i^c (x_i^c)^⊤
    Perform EVD for R_c and obtain r-major eigenvectors u_1^c, ..., u_r^c.
    Let U_c = [u_1^c, ..., u_r^c].
end for
# main loop
for iter = 1 to T do
    Classify all training samples, and obtain E_c and S_c for c = 1, ..., C
    for c = 1 to C do
        Update the correlation matrix R_c by using Eq. (4).
        Perform EVD for R_c and update U_c.
    end for
end for
```

$$\boldsymbol{R}_c \leftarrow \boldsymbol{R}_c + \alpha \sum_{\boldsymbol{x} \in \mathcal{E}_c} \boldsymbol{x}\boldsymbol{x}^\top - \beta \sum_{\boldsymbol{x} \in \mathcal{S}_c} \boldsymbol{x}\boldsymbol{x}^\top, \tag{4}$$

where α and β control the strength of the enhancement and the suppression, respectively. After we update the subspaces, we again update $\mathcal{E}_c$ and $\mathcal{S}_c$. Finally, the optimal subspaces are obtained. We summarize this procedure as Algorithm 1.

It should be noted that the modified correlation matrix $\boldsymbol{R}_c$ in Eq. (4) is obtained from all of the samples in class c, with the addition of a small subset of samples that do not belong to class c. This reduces the computational complexity when ALSM is extended to a kernel version. An extension of SVM uses a similar approach to explicitly mine samples that are difficult to classify [7]. On the other hand, a binary classifier with the one-vs-all approach uses all of the training samples and has a higher computational complexity.

2.2 KPCA and Subset Approximation

PCA finds the r-dimensional subspace that maximizes the power of the transformed vectors;

$$\begin{aligned}\max_{U} \quad & \sum_{i=1}^{N} \|\boldsymbol{U}^\top \boldsymbol{x}_i\|^2 = \mathrm{Trace}[\boldsymbol{U}^\top \boldsymbol{R} \boldsymbol{U}], \\ \text{subject to} \quad & \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_r,\end{aligned} \tag{5}$$

where $\boldsymbol{I}_r$ is the identity matrix of size r. In this section, we omit the subscript c indicating the class label.

KPCA maximizes the power of the mapped vectors;

$$\begin{aligned}\max_{U} \quad & \sum_{i=1}^{N} \|U^* \phi(\boldsymbol{x}_i)\|^2, \\ \text{subject to} \quad & U^* U = \boldsymbol{I}_r,\end{aligned} \tag{6}$$

where $\phi(\cdot)$ is a mapping function from the input space to the feature space. The asterisk $\cdot^*$ denotes the operator defined by $g^* f = \langle f, g \rangle$, or $\langle Af, g \rangle = \langle f, A^* g \rangle$ for all f, g in the feature space, where $\langle \cdot, \cdot \rangle$ denotes the inner product. Note that since U can be infinite, the transpose $\cdot^\top$ is replaced by $\cdot^*$. Since

the columns of U can be expressed by a linear combination of $\phi(\boldsymbol{x}_i)$ $(i = 1, \ldots, N)$, U can be parameterized as

$$U = \Phi \boldsymbol{A}, \tag{7}$$

where $\Phi = [\phi(\boldsymbol{x}_1), \ldots, \phi(\boldsymbol{x}_N)]$ is a basis operator, $\boldsymbol{A}$ is an $N \times r$ matrix, and $\boldsymbol{A}$ is obtained by the EVD of the Gram matrix $\boldsymbol{K} = \Phi^* \Phi \in \mathbb{R}^{N \times N}$. The i, j-th element of $\boldsymbol{K}$ is given by $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $k(\cdot, \cdot)$ is a positive definite kernel function.

Subset KPCA approximates U by using a limited basis vectors $\Psi = [\phi(\boldsymbol{z}_1), \ldots, \phi(\boldsymbol{z}_M)]$, $M \leq N$,

$$\tilde{U} = \Psi \boldsymbol{B}, \tag{8}$$

where $\boldsymbol{B}$ is an $M \times r$ matrix [15]. We assume that Ψ is given. Then the problem (6) is reduced to

$$\begin{array}{ll} \max\limits_{\boldsymbol{B}} & \boldsymbol{B}^\top \boldsymbol{K}_{xz}^\top \boldsymbol{K}_{xz} \boldsymbol{B} \\ \text{subject to} & \boldsymbol{B}^\top \boldsymbol{K}_{\mathcal{B}} \boldsymbol{B} = \boldsymbol{I}_M, \end{array} \tag{9}$$

where $\boldsymbol{K}_{xz} = \Phi^* \Psi$ and $\boldsymbol{K}_{\mathcal{B}} = \Psi^* \Psi$, that is $(\boldsymbol{K}_{xz})_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{z}_j)$ and $(\boldsymbol{K}_{\mathcal{B}})_{i,j} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)$. The problem (9) is solved by generalized EVD, $\boldsymbol{K}_{xz}^\top \boldsymbol{K}_{xz} \boldsymbol{b} = \lambda \boldsymbol{K}_{\mathcal{B}} \boldsymbol{b}$, where $\boldsymbol{b}$ is a column of $\boldsymbol{B}$.

The size of the generalized EVD is M, since $\boldsymbol{K}_{xz}^\top \boldsymbol{K}_{xz} \in \mathbb{R}^{M \times M}$. Compared to using standard KPCA using $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M$ as the training samples (size of EVD is also M), the value of the objective function (6) for the subset KPCA is always greater than that of the standard KPCA. In [15], [16], the basis vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M$ are determined by K-means or random sample consensus (RANSAC). This results in a smaller approximation error than that obtained when using simple KPCA with a limited number of training vectors. We will discuss our efficient method for selecting basis vectors for classification in Sect. 3.

The projection norm of an input vector $\boldsymbol{x}$ is given by

$$g_c(\boldsymbol{x}) = \|U^* \phi(\boldsymbol{x})\|^2 = \|\boldsymbol{B}^\top \boldsymbol{k}_{\boldsymbol{x}}\|^2 \tag{10}$$

$$\boldsymbol{k}_{\boldsymbol{x}} = [k(\boldsymbol{z}_1, \boldsymbol{x}), \ldots, k(\boldsymbol{z}_M, \boldsymbol{x})]^\top. \tag{11}$$

The number of evaluations of the kernel function is M, and the total number of multiplications is $r(M + 1)$.

3. Proposed Classification Method

In order to develop a kernel version of ALSM, we will address the following three issues:

1. The updating rule in the feature space,
2. Selection of the basis vectors,
3. Extension of the reweighting rule.

3.1 The Updating Rule in the Feature Space

We extend the updating rule, Eq. (4) to a kernel version by using the subset approximation. The updated modified correlation matrix at the ith iteration, $\boldsymbol{R}_c^i$, is expressed as

$$\boldsymbol{R}_c^i = \boldsymbol{X}_c \boldsymbol{X}_c^\top + \alpha \boldsymbol{X}_{c,\mathcal{E}_c}^i (\boldsymbol{X}_{c,\mathcal{E}_c}^i)^\top - \beta \boldsymbol{X}_{c,\mathcal{S}_c}^i (\boldsymbol{X}_{c,\mathcal{S}_c}^i)^\top, \tag{12}$$

where the columns of $\boldsymbol{X}_{c,\mathcal{E}_c}^i$ and $\boldsymbol{X}_{c,\mathcal{S}_c}^i$ are cumulative vectors in the enhancement set $\mathcal{E}_c$ and the suppression set $\mathcal{S}_c$, respectively.

Let Φ_c, $\Phi_{\mathcal{E}}^c$, and $\Phi_{\mathcal{S}}^c$ be operators that arrange the mapped vectors in $\mathcal{X}_c$, $\mathcal{E}_c$, and $\mathcal{S}_c$, respectively, and let

$$\boldsymbol{K}_{\mathcal{X}\mathcal{B}}^c = \Phi_c^* \Psi \tag{13}$$

$$\boldsymbol{K}_{\mathcal{E}\mathcal{B}}^c = (\Phi_{\mathcal{E}}^c)^* \Psi \tag{14}$$

$$\boldsymbol{K}_{\mathcal{S}\mathcal{B}}^c = (\Phi_{\mathcal{S}}^c)^* \Psi \tag{15}$$

$$\boldsymbol{K}_c = (\boldsymbol{K}_{\mathcal{X}\mathcal{B}}^c)^\top \boldsymbol{K}_{\mathcal{X}\mathcal{B}}^c + \alpha (\boldsymbol{K}_{\mathcal{E}\mathcal{B}}^c)^\top \boldsymbol{K}_{\mathcal{E}\mathcal{B}}^c - \beta (\boldsymbol{K}_{\mathcal{S}\mathcal{B}}^c)^\top \boldsymbol{K}_{\mathcal{S}\mathcal{B}}^c, \tag{16}$$

where Ψ is the basis of the subset KPCA, and we assume Ψ is given. Then the subspace parameterized by $\boldsymbol{B}_c \in \mathbb{R}^{M \times r}$ $(U_c = \Psi \boldsymbol{B}_c)$ is obtained from the optimization problem,

$$\begin{array}{ll} \max\limits_{\boldsymbol{B}_c} & \mathrm{Trace}[\boldsymbol{B}_c^\top \boldsymbol{K}_c \boldsymbol{B}_c] \\ \text{subject to} & \boldsymbol{B}_c^\top \boldsymbol{K}_{\mathcal{B}} \boldsymbol{B}_c = \boldsymbol{I}_r. \end{array} \tag{17}$$

3.2 Selection of Basis Vectors

We have thus far assumed that the basis vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_M$, and the mapped set $\Psi = [\phi(\boldsymbol{z}_1), \ldots, \phi(\boldsymbol{z}_M)]$ are given. In [15], [16], the selection of basis vectors is discussed in terms of the approximation error of KPCA. However, since the purpose of the proposed method is classification, the criterion should be based on classification error. For example, in the case of SVM, the hinge loss that represents classification error is used, but l_2 or l_1-error that represents approximation error was not used. In the case of the relevance vector machine (RVM) classifier, the logistic function (measures classification error) is used, but Gaussian function (measures approximation, it is used in RVM regression) is not used. In the case of ALSM and CLAFIC, CLAFIC uses total approximation error whereas ALSM uses classification error. These precedents imply that the approximation error is not an optimal criterion for classification. From a practical viewpoint, roughly speaking, minimizing total approximation error fits the most probable patterns that are easy to be classified, whereas the classification error fits to samples that are easily misclassified. We thus propose a sample selection method based on the classification error.

In kernel ALSM, the subspace is determined by the matrix $\boldsymbol{B}$ and the basis Ψ (Eq. (8)). $\boldsymbol{B}$ is determined by the kernel version of ALSM criterion (17). In order to determine the basis Ψ, we use the same criterion because it provides the optimal subspace for classification. Since it is difficult to solve the optimization problem for Ψ directly, we propose a greedy algorithm for sequentially selecting the basis samples. As in ALSM, the proposed method initially obtains subspaces by using subset KPCA with a limited number of basis vectors. These basis vectors can be obtained by using clustering methods, such as the K-means method.

Suppose the subset KPCA $U = \Psi \boldsymbol{B}$ is given by the current basis set $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$, $(\Psi = [\phi(\boldsymbol{z}_1, \ldots, \phi(\boldsymbol{z}_m))])$. We find the next optimal basis vector to maximize the criterion

(17),

$$\max_{z} \mathrm{Trace}[\boldsymbol{B}_c^\top \tilde{\boldsymbol{K}}_c \boldsymbol{B}_c] \tag{18}$$

where $\tilde{\boldsymbol{K}}_c$ is the updated kernel matrix of Eq. (16) calculated by the updated basis $\tilde{\Psi} = [\Psi|\phi(z)]$. However, when we update the basis vectors (i.e. when we update Ψ), we have to recompute the generalized eigenvector $\boldsymbol{B}$ for each of the candidate vectors. This is also computationally expensive.

In PCA, vectors can be approximated by a linear combination of a limited number of eigenvectors, because a small number of the eigenvalues of the correlation matrix have large values, and the remaining eigenvalues are small. In other words, we can obtain a good approximation to the entire input space by using the space spanned by a limited number of eigenvectors. In the case of (subset) KPCA, the space spanned by the mapped basis vectors (i.e., the range of Ψ) can also be approximated well by the space spanned by U. Therefore, we approximate the projection onto span(U) by the projection onto span(Ψ) [17]. By introducing this approximation, we can reduce the computational complexity for the selection basis vector.

The projector onto span(Ψ) is given by

$$P_\Psi = \Psi(\boldsymbol{K}_\mathcal{B})^{-1}\Psi^*. \tag{19}$$

If the projection norm $\|P_\Psi\phi(\boldsymbol{x})\|$ of a vector $\boldsymbol{x}$ in $\mathcal{E}_c$ or $\mathcal{S}_c$ is small, the current basis will not provide a good approximation for it. Thus, we add the vector to the basis set. Consequently, the sample selection rule is to add a vector z such that

$$\min_{z\in\mathcal{E}_c\cup\mathcal{S}_c} \frac{\|P_\Psi\phi(z)\|^2}{\|\phi(z)\|^2} = \langle \boldsymbol{k}_z, \boldsymbol{K}_\mathcal{B}^{-1}\boldsymbol{k}_z\rangle / k(z,z), \tag{20}$$

where $\boldsymbol{k}_z$ is a vector given by Eq. (11).

Suppose that we find a new basis vector z^* by Eq. (20), and we have the Cholesky decomposition of the current Gram matrix $\boldsymbol{K}_\mathcal{B} = \boldsymbol{L}\boldsymbol{L}^\top$, where $\boldsymbol{L}$ is a lower triangular matrix. Then the updated kernel matrix $\tilde{\boldsymbol{K}}_\mathcal{B}$ and its Cholesky decomposition $\tilde{\boldsymbol{L}}$ are given by

$$\tilde{\boldsymbol{K}}_\mathcal{B} = \begin{bmatrix} \boldsymbol{K}_\mathcal{B} & \boldsymbol{k}_{z^*} \\ \boldsymbol{k}_{z^*}^\top & k(z^*,z^*) \end{bmatrix} = \tilde{\boldsymbol{L}}\tilde{\boldsymbol{L}}^\top \tag{21}$$

$$\tilde{\boldsymbol{L}} = \begin{bmatrix} \boldsymbol{L} & \boldsymbol{0} \\ (\boldsymbol{L}^{-1}\boldsymbol{k}_{z^*})^\top & q \end{bmatrix} \tag{22}$$

$$q = \sqrt{k(z^*,z^*) - \|\boldsymbol{L}^{-1}\boldsymbol{k}_{z^*}\|^2}. \tag{23}$$

Note that inverse operation on the lower triangular matrix $\boldsymbol{L}^{-1}\boldsymbol{k}_{z^*}$ requires only $M(M+1)/2$ multiplications which is the same computational complexity as $\boldsymbol{L}\boldsymbol{k}_{z^*}$. Moreover, as we show next, we do not need to calculate this inverse operation during the updating process.

The numerator of the right-hand side of Eq. (20) is

$$\langle \boldsymbol{k}_z, \boldsymbol{K}_\mathcal{B}^{-1}\boldsymbol{k}_z\rangle = \|\boldsymbol{L}^{-1}\boldsymbol{k}_z\|^2. \tag{24}$$

We store vectors $\boldsymbol{l}_z = \boldsymbol{L}^{-1}\boldsymbol{k}_z$ for all $z \in \mathcal{E}_c \cup \mathcal{S}_c$. Then, from

Algorithm 2 Basis selection algorithm (the class index c is omitted)

Require: Enhance set: $\mathcal{E}$, Suppression Set: $\mathcal{S}$, initial basis $z_1, \ldots, z_m$, kernel function: $k(\cdot,\cdot)$, Basis size: M
Ensure: Basis vectors $z_1, \ldots, z_M$
1: **if** $|\mathcal{E}\cup\mathcal{S}| \le M - m$ **then**
2: Set all vectors in $\mathcal{E}\cup\mathcal{S}$ to the basis vectors
3: **else**
4: Let $\Psi = [\phi(z_1), \ldots, \phi(z_m)]$
5: Calculate kernel matrix $\boldsymbol{K}_\mathcal{B} = \Psi^*\Psi$ and its Cholesky decomposition $\boldsymbol{L}$ (this procedure can be done outside of the main loop of Algorithm 3)
6: Obtain kernel vectors for initial basis vectors, $\boldsymbol{k}_z$ and $\boldsymbol{l}_z = \boldsymbol{L}^{-1}\boldsymbol{k}_z$ for all $z \in \mathcal{E}\cup\mathcal{S}$
7: **for** $i = m+1$ **to** M **do**
8: Find the sample z^* to be added by the criterion (20),

$$z^* = \operatorname*{argmin}_{z\in\mathcal{E}\cup\mathcal{S}} \|\boldsymbol{l}_z\|^2 / k(z,z)$$

9: Set $z_i = z^*$ and update $\boldsymbol{l}_z$ by Eq. (25) for all $z \in \mathcal{E}\cup\mathcal{S}$
10: **end for**
11: **end if**

a property of the inverse of the partitioned matrix [18], we obtain the updated vector from

$$\tilde{\boldsymbol{l}}_z = \tilde{\boldsymbol{L}}^{-1}\tilde{\boldsymbol{k}}_z = \begin{bmatrix} \boldsymbol{l}_z \\ \frac{1}{q}(k(z,z^*) - \boldsymbol{l}_{z^*}^\top \boldsymbol{l}_z) \end{bmatrix} \tag{25}$$

$$\boldsymbol{l}_{z^*} = \boldsymbol{L}^{-1}\boldsymbol{k}_{z^*}.$$

Thus, the main computations required for selecting the basis vectors are $\boldsymbol{l}_{z^*}^\top \boldsymbol{l}_z$ and $\|\boldsymbol{l}_z\|^2$ for all $z \in \mathcal{E}_c \cup \mathcal{S}_c$. We summarize the basis selection algorithm as Algorithm 2.

Since $\|\boldsymbol{l}_z\|^2$ and $\boldsymbol{l}_{z^*}^\top \boldsymbol{l}_z$ require i multiplications, the total number of multiplications in the for-loop is $n\sum_{i=m+1}^{M} 2i+1$, where $n = |\mathcal{E}\cup\mathcal{S}|$. In addition, there are $(n+1)(M-m)$ evaluations of the kernel function and $(M-m)$ square-root operations.

3.3 Extension of the Reweighting Rule

The ALSM algorithm finds the samples that are not classified correctly and enhances or suppresses them in the next iteration. In the high-dimensional feature space of kernel machines, the number of degrees of freedom is higher than it is in the Euclidean input space. Therefore, the learning can easily over-fit the training samples. In other words, during the learning procedure, almost all of the training samples are correctly classified, and there is no improvement of the generalization performance.

To solve this problem, we introduce the idea of relative similarity of the training samples,

$$h(\boldsymbol{x}_i^c) = g_c(\boldsymbol{x}_i^c) / \max_{k\neq c} g_k(\boldsymbol{x}_i^c). \tag{26}$$

If $h(\boldsymbol{x}_i^c) < 1$, $\boldsymbol{x}_i^c$ is considered to be misclassified and is added to the enhancement set $\mathcal{E}_c$ and the suppression set $\mathcal{S}_c$ in the original ALSM algorithm. We tighten this condition, and add to $\mathcal{E}_c$ and $\mathcal{S}_c$ samples that are correctly classified by a narrow margin,

Algorithm 3 Proposed kernel learning subspace method

Require: Labeled training set: $\mathcal{X}_1, \ldots, \mathcal{X}_C$, Hyper-parameters, α, β: Strength of enhancement and suppression, θ: Threshold, T: Number of maximum iterations, r: Dimension of subspaces, $k(\cdot,\cdot)$: Kernel function, M: Maximum number of basis vectors
Ensure: Pair of matrix $\boldsymbol{B}^c$ and basis set $\mathcal{B}^c$
1: # Obtain the initial subspace by Kernel CLAFIC
2: **for** $c = 1$ **to** C **do**
3: Obtain the initial basis $\mathcal{B}_0^c$ from $\mathcal{X}_c$ by clustering
4: Perform subset KPCA using training samples $\mathcal{X}_c$ and the basis $\mathcal{B}_0^c$, and obtain $\boldsymbol{B}^c$.
5: **end for**
6: # Main loop
7: **for** *iter* $= 1$ **to** T **do**
8: Classify all training samples and obtain the relative similarity $h_c(\boldsymbol{x})$ for all classes c and training samples by using Eq. (26).
9: **for** $c = 1$ **to** C **do**
10: Add samples to $\mathcal{E}_c$ and $\mathcal{S}_c$ by Eqs. (27) and (28)
11: Obtain basis vectors $\mathcal{B}^c$ by Algorithm 2
12: Obtain $\boldsymbol{K}_c$ from Eqs. (13)-(16)
13: Perform subset KPCA and obtain $\boldsymbol{B}^c$
14: **end for**
15: Quit the iteration if the sum of $h(\boldsymbol{x}_i^c)$ is converge, or no vectors are added to $\mathcal{E}_c$ and $\mathcal{S}_c$.
16: **end for**

$$\mathcal{E}_c = \{\boldsymbol{x}_i^c | h(\boldsymbol{x}_i^c) < 1 + \theta\} \tag{27}$$

$$\mathcal{S}_c = \{\boldsymbol{x}_i^k \ (k \in \{1, \ldots, C\} \backslash c) | \ (h(\boldsymbol{x}_i^k) < 1 + \theta) \text{ and } (c = \underset{k' \neq k}{\operatorname{argmax}} g_{k'}(\boldsymbol{x}_i^k))\} \tag{28}$$

where $\theta \geq 0$ is a parameter that controls the margin. When $\theta = 0$, this is equivalent to the original ALSM.

We summarize the algorithm of the proposed classification method as Algorithm 3. When the dimension of the subspace, r is larger than the current (or initial) basis size $|\mathcal{B}^c|$, the solution of the problem (9) is not unique. In such case, we simply use the $|\mathcal{B}^c|$-dimensional subspace spanned by the current (or initial) basis vectors.

4. Computational Efficiency

The decision function of a kernel subspace method is a quadratic model,

$$f(\boldsymbol{x}) = \langle \phi(\boldsymbol{x}), A\phi(\boldsymbol{x}) \rangle, \tag{29}$$

$$A = \Phi \boldsymbol{B} \Phi^* = \sum_{i=1}^{M} \sum_{j=1}^{M} \beta_{ij} \phi(\boldsymbol{z}_i) \phi(\boldsymbol{z}_j)^*. \tag{30}$$

On the other hand, the decision function of sparse kernel machines such as SVMs, RVMs, and multiple kernel learning algorithms (MKL), is a linear model [19]–[21],

$$f(\boldsymbol{x}) = \langle w, \phi(\boldsymbol{x}) \rangle \tag{31}$$

$$w = \Phi \boldsymbol{\alpha} = \sum_{i=1}^{M} \alpha_i \phi(\boldsymbol{z}_i), \tag{32}$$

where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_M]^\top$ is a weight vector whose ith element α_i is associated with the basis vector $\boldsymbol{x}_i$. The weight of a sparse kernel machine is supported by a limited number of training vectors, the so-called support vectors. These sparse kernel machines are binary classifiers, and hence, when applying them to multiclass problems, a one-vs-the-rest or one-vs-one approach is required [22]. If the number of classes is large, the computational cost will be high.

Table 1 Comparison of computational efficiency, L: no. of classes, n: total number of samples in $\mathcal{E}, \mathcal{S}$ and the target class, M: no. of basis vectors (support vectors), r: rank (no. of dimension), N: no of samples in one class

Classifier	Training	Test
Proposed	$O(LnM^2)$	$O(rLM)$
one-vs-the-rest SVM	$O(L^3N^2)$	$O(LM)$
one-vs-one SVM	$O(L^2N^2)$	$O(L^2M)$
Kernel CLAFIC	$O(rLN^2)$	$O(rN)$

In the feature space, the quadratic model (29) has more degrees of freedom than does the linear model (31). Therefore, it is expected that the smaller number of basis vectors used in the proposed method will perform the classification as accuracy as does the linear model (31).

The classification methods for multiclass classification problems can be categorized as follows [22];

1. one-vs-one approach using binary classifiers
2. one-vs-the-rest (one-vs-all, one-vs-others) approach using binary classifiers
3. multiclass classifier
4. one-class classifier (density estimation)

The one-vs-one approach makes classifiers for all possible combinations, i.e., $L(L-1)/2$ classifiers for an L-class classification problem. Thus, the computational cost and memory are large if L is rather large.

The one-vs-the-rest approach requires L classifiers. However, the computational cost to obtain one classifier is higher than it is with the one-vs-one approach since all training samples are used for each classifier. For example, suppose N is the number of samples per class and L is the number of classes. Since the computational cost of training SVM is roughly quadratic in N [23], the one-vs-one approach costs $(L(L-1)/2$ classifiers$) \times (2N)^2$, i.e., $O(L^2N^2)$ plus lower-order terms. On the other hand, the computational cost of the one-vs-the-rest approach is (L classifiers) $\times (LN)^2$, i.e., $O(L^3N^2)$ plus lower-order terms.

A multiclass classifier obtains one classifier for a multiclass problem; e.g., multiclass SVM [24] and multilayer perceptrons [25]. They also use all of the training samples in order to obtain the classifier. If the number of classes L and the number of training samples are large, the computational cost is large.

A one-class classifier is a classifier whose decision function is learned from samples of one class. The k-nearest neighbor rule, the Bayesian decision rule, the single-class SVM [26], and the subspace classifier [9] are examples of one-class classifiers. Since a one-class classifier does not take into account the difference in features between classes, the accuracy, in general, is not as high as it is in other classifiers.

The proposed method takes into account the samples that are misclassified. Therefore, a limited number of sam-

ples are used to obtain the eigenvectors, and the computational complexity is realistic even when the quadratic decision function is used. We will give actual computational times for several datasets in Sect. 5.

The computational complexity of the proposed method primarily depends on the EVD for subset KPCA and the updating process. The computational complexity of subset KPCA is the generalized EVD for an $M \times M$ matrix. For self-evaluation, NM kernel evaluations and M^2 multiplications are required. After the self-evaluation, suppose that n is the total number of samples in $\mathcal{E}$, $\mathcal{S}$, and the target class. Then $nM + M^2$ kernel evaluations, and $nM(M+1)/2$ multiplications are required. It should be noted that M is tunable in our method.

For the evaluation stage, the linear sparse kernel machine, Eq. (31) requires M kernel evaluations and M multiplications. On the other hand, the quadratic sparse kernel machine, Eq. (29) requires M kernel evaluations and $M(M+1)/2 + M$ multiplications. In our case, since $\boldsymbol{B}$ in Eq. (29) is rank reduced, the number of multiplications is $(M+1)r$, where r is the dimension of the subspace. The required memory to store basis vectors is the same for both Eqs. (31) and (29).

5. Experiments

We show three experimental results. In Sect. 5.1, USPS handwritten-digits dataset is used to detailed comparison using various parameter selection. In Sect. 5.2, UCI datasets are used to compare the performance over various datasets. In Sect. 5.3, MNIST dataset is used to compare the computational efficiency for the large data.

5.1 USPS Handwritten-Digits Dataset

We used a US Postal Service (USPS) handwritten-digits dataset [27]. The dataset has 7291 images for training and 2007 images for the test. The images are grayscale, and each image comprises 16×16 pixels. For preprocessing, we normalized a 256-dimensional vector to a unit norm. To observe the statistical behavior, we mixed the original training and test samples, and randomly split all of the 9298 samples into a training set (1860 samples, 20% of total) and a test set (7438 samples, 80% of total). We obtained 20 realizations using random splitting.

The hyperparameters were obtained from a ten fold cross-validation for each realization, using the training samples. Thus, each realization could result in different estimations of the hyperparameters.

We compared the following classification methods;

1. proposed classification method (PRO)
2. proposed classification method without basis selection(PRO2)
3. k-nearest neighbor (KNN)
4. one-vs-the-rest SVM (SVM1)
5. one-vs-one SVM (SVM2)
6. simple MKL [28] (SMKL)
7. CLAFIC [1] (CLA)
8. ALSM [3] (ALSM)
9. Kernel CLAFIC [9], [10] (KCLA)

Table 2 Results of USPS handwritten digit recognition; mean test error and standard deviations over 20 realizations

Method	Test error[%]
PROl	3.807 ± 0.258
PROp	3.660 ± 0.187
PROr	3.764 ± 0.172
PRO2l	3.937 ± 0.291
PRO2p	3.649 ± 0.248
PRO2r	3.679 ± 0.186
KNN	6.304 ± 0.293
SVM1l	7.456 ± 0.330
SVM1p	3.905 ± 0.229
SVM1r	3.957 ± 0.218
SVM2l	5.654 ± 0.345
SVM2p	3.941 ± 0.269
SVM2r	4.026 ± 0.315
SMKL	4.347 ± 0.283
CLAFIC	5.098 ± 1.299
ALSM	4.818 ± 0.369
KCLAl	4.450 ± 0.239
KCLAp	4.068 ± 0.233
KCLAr	4.091 ± 0.237

We denote the kernel functions by l, p, and r, which respectively mean

- linear kernel: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle$,
- polynomial kernel: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = (\langle \boldsymbol{x}_1, \boldsymbol{x}_2 \rangle + 1)^n$,
- radial basis function (RBF) kernel (Gaussian kernel), $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp(-\gamma \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2)$.

For example, (PROp) stands for the proposed method with the polynomial kernel. For PRO2, all training vectors in the class, enhancement, and suppression sets are used for the basis vectors. LibSVM was used for SVM1 and SVM2 [29]. The RBF kernels with $\gamma = 10^{-3}, 10^{-2.5}, 10^{-2.0}, \ldots, 10^{3}$ were used for SMKL. For the proposed method, the number of iterations was set to ten, the initial basis size was fixed to ten, and $M = 200$. A comparison of these parameters will be presented below in this section. The other parameters were obtained from cross-validation.

The mean test errors and standard deviations over 20 realizations are shown in Table 2. For all three kernel functions, the proposed methods exhibited significantly lower error rates than did SVM1 and SVM2, as determined by a pair-wise one-sided t-test ($p \leq 0.01$).

For the linear kernel case, SVM1l exhibited a larger error than did the other classification methods. This is due to the problem that linear classifiers perform poorly with multiclass problems. It should also be noted that PROl exhibited a lower error rate than those of either SVM1p or SVM2p, although the models of SVMs with the second-order polynomial kernel includes the model of PROl. Since either the linear SVM or the second-order polynomial SVM can be calculated in an input space with no kernel functions, they are used in applications that are required to be fast and light, in

Table 3 Results of USPS handwritten-digit recognition; number of basis vectors averaged for twenty realizations, and their standard deviations. nBV1: total number of basis vectors for all classes. nBV2: number of basis vectors eliminating duplications for classes.

Method	nBV1	nBV2
PROl	1623 ± 437	856 ± 253
PROp	1720 ± 380	979 ± 238
PROr	1821 ± 271	1042 ± 175
SVM1l	1424 ± 165	868 ± 74
SVM1p	3470 ± 1002	1188 ± 92
SVM1r	3182 ± 584	1179 ± 83
SVM2l	2143 ± 280	719 ± 51
SVM2p	4669 ± 1025	988 ± 81
SVM2r	4543 ± 1220	983 ± 103
KCLA	1860 ± 0	1860 ± 0

particular, for higher-dimensional classification problems. Therefore, the proposed method can be an alternative classifier for such applications.

For the polynomial and RBF kernel case, we choose the kernel parameters n and γ by cross-validation. The mean values of optimal degree n were $n = 2.4$ for PROp, $n = 9.12$ for SVM1p, and $n = 6.75$ for SVM2p. The mean values of optimal γ were $\gamma = 0.521$ for PROr, $\gamma = 2.141$ for SVM1r, and $\gamma = 1.622$ for SVM2r. Even in the feature space, the proposed method, which has a quadratic similarity function, tends to make selections that have fewer degrees of freedom (smaller kernel parameters). Since the proposed method does not require much nonlinear mapping by the kernel function, it can prevent over-fitting, unlike that which occurs with the linear classification methods in the feature space.

In order to see the effect of the basis selection algorithm, we also conducted experiments of the proposed method using random basis selection and K-means basis selection proposed in [16]. In the random selection case, we randomly extract basis vectors from the initial basis set, the enhancement and suppression sets. The test errors and standard deviations were PROp: 3.87 ± 0.22% and PROr: 3.91 ± 0.22%. They are significantly higher than the case of the proposed basis selection method (paired t-test $\alpha = 0.01$). In the K-means selection case, the test errors and standard deviations were l: 4.31 ± 0.35%, p: 3.85 ± 0.24%, and r: 3.84 ± 0.26% ('l' and 'p' show statistical significance by the paired t-test $\alpha = 0.01$). These results show that the proposed basis selection method is useful for classification problems.

In the case that the reweighting rule in Sect. 3.3 is not used, the test error rates and standard deviations were PROl: 5.01 ± 0.34%, PROp: 5.03 ± 0.32%, and PROr: 5.03 ± 0.33%. The kernelization does not solely improve the classification performance of ALSM, but the combination with the proposed reweighting rule do. The reason is that almost all training samples were classified correctly in the early stage due to higher degree of freedom of kernel methods. In other words, the classifier fits to the training samples, and the empirical error to be minimized is already minimized in the early stage.

Table 3 lists the number of basis vectors (support vectors for SVM). nBV1 is the total number of the basis vectors for all classes. nBV2 is the number of the basis vectors eliminated duplications. Since M is the maximum basis size for each class, (10 classes) × ($M = 200$) = 2000 is the upper limit of nBV1. If some classes meet the condition in the first line of Algorithm 2 ($|\mathcal{E} \cup \mathcal{S}| \leq M - m$), nBV1 is smaller than 2000. The computational complexity for the inner product part or the quadratic form (M in Eqs. (32) and (30)) depends on nBV1. The required memory and computational complexity for kernel evaluation depends on nBV2. The numbers of total basis vectors, nBV1 for the proposed method are much less than those for SVM1 and SVM2, while the classification errors of the proposed methods are lower. Note that for the proposed method, the maximum number of basis vectors is fixed. In other words, we can tune the size of the basis. The size of the basis has a trade-off between classification accuracy and computational complexity. However, as we will discuss below, when $M = 200$, classification accuracy is almost saturated, and even if we increase the basis size, the classification accuracy is not improved. Since there are ten classes, the average nBV1 for the proposed method for one class is about 100 to 200. In this case, the size of the generalized EVD is less than that of the EVD in the original ALSM, for which the input dimension was 256. Even PROl and PROr exhibit kernel tricks; the size of the main problem was smaller than that of the original ALSM.

Table 4 Comparison of runtimes. Runtimes for cross-validation and data loading were excluded.

Methods	Training [s]	Test [s]
PROl	2.91 ± 0.98	0.33 ± 0.08
PROp	3.37 ± 0.96	0.39 ± 0.09
PROr	4.02 ± 0.74	0.56 ± 0.08
PRO2l	5.53 ± 3.23	0.55 ± 0.15
PRO2p	5.03 ± 2.63	0.58 ± 0.14
PRO2r	6.45 ± 2.62	0.86 ± 0.15
SVM1l	1.26 ± 0.07	0.02 ± 0.00
SVM1p	2.53 ± 0.69	0.89 ± 0.27
SVM1r	2.42 ± 0.41	1.21 ± 0.25
SVM2l	0.36 ± 0.03	0.08 ± 0.00
SVM2p	0.69 ± 0.17	1.22 ± 0.24
SVM2r	0.71 ± 0.20	1.77 ± 0.41
SMKL	121.07 ± 5.95	33.63 ± 4.37
CLAFIC	0.13 ± 0.00	0.09 ± 0.06
ALSM	1.78 ± 0.04	0.07 ± 0.01
KCLAl	0.08 ± 0.00	0.37 ± 0.01
KCLAp	0.10 ± 0.00	0.46 ± 0.06
KCLAr	0.11 ± 0.00	0.61 ± 0.03

Table 4 compares the runtimes. We conducted the experiment on a PC with an Intel Core-i5 3550 (3.30 GHz) CPU and 16GB of RAM. The proposed method was implemented and executed on a GNU Octave 3.6.1 with the Intel Math Kernel Library (MKL). Single process was used for all cases. The runtime for data loading was excluded. For the training stage, the computational complexity of the proposed method was about 1.3 to 2.3 times that of SVM1. For the test stage, the proposed method was faster than both SVM1 and SVM2, even though it has a quadratic dis-

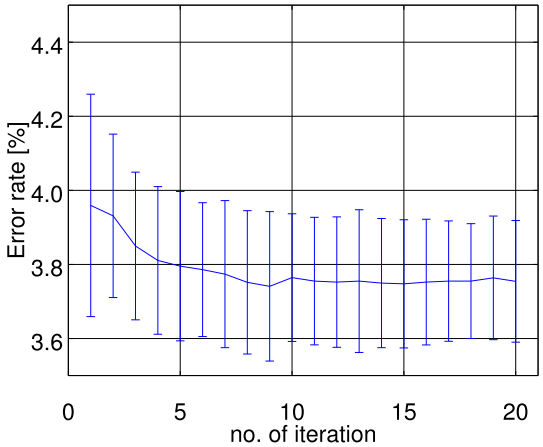

Fig. 2 Test error and the number of iterations

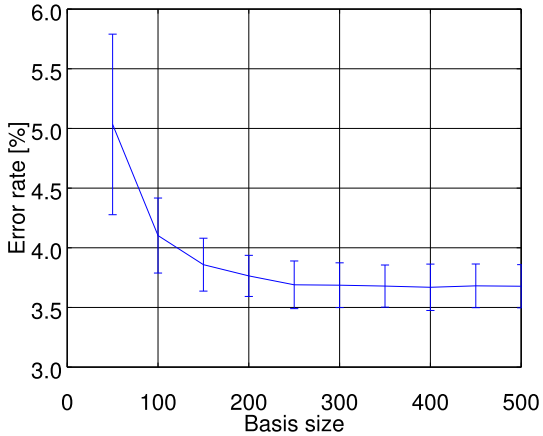

Fig. 3 Test error and maximum number of basis vectors

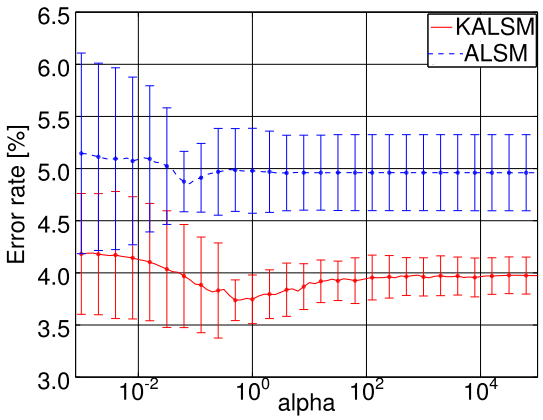

Fig. 4 Test error and α

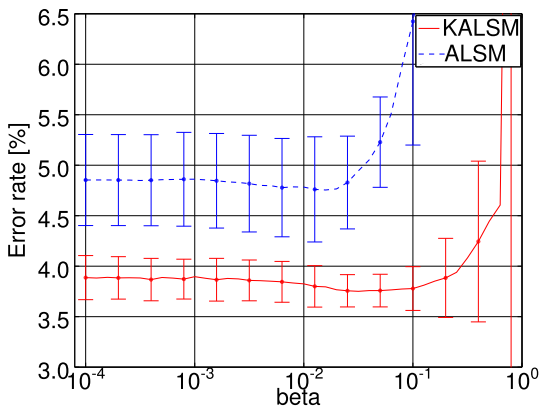

Fig. 5 Test error and β

criminant function. Note that, for a fair comparison, the classifications of SVM1 and SVM2 (Eq. (32)), the test programs were implemented on a GNU Octave that uses an Intel MKL. The Intel MKL is much faster than the implementation of LibSVM.

Compared to kernel CLAFIC, the proposed method exhibited higher classification performance using smaller number of nBV1 because kernel CLAFIC learns from only the target class samples whereas the proposed method learns from the target and non-target class samples. Since nBV1 of the proposed method is fewer than that of kernel CLAFIC, the computational cost of the proposed method in the test stage is smaller than that of kernel CLAFIC. However, in the training stage, kernel CLAFIC performs only one EVD for each target class whereas the proposed method requires several EVDs. Thus, even if the size of EVD is smaller in the proposed method, total computational cost of the proposed method in the training stage is larger than that of kernel CLAFIC. In many practical problems, a classifier is trained once, and then it tests many times. Therefore, the computational complexity in the test stage is important. Since Kernel CLAFIC is trained from samples belonging to the target class, its computational cost in the test stage is not so large although kernel CLAFIC does not use the basis reduction.

When we do not use the proposed basis selection method in Sect. 3.2 (PRO2), the computational cost of PRO2 is larger than that of PRO since the number of basis is larger. Moreover, for larger dataset, the difference will be larger.

Figure 2 shows the relation between the test error of PROr and number of iterations. The parameters were fixed to the optimal values obtained by cross-validation. The solution was stable after ten iterations.

Figure 3 shows the relation between the test error of PROr and the number of basis vectors M. The results of the proposed method (maximum size of basis M is 200), shown in Table 2 had a sufficiently large basis.

Figures 4 and 5 show the relation between α and β, respectively, and the test error of PROr. The proposed method had the lowest error rates at larger value of α and β than did ALSM. This is due to the larger number of degrees of the freedom in the proposed method. Even when α was larger than the optimal value, the performance was still better than it was for $\alpha = 0$. For β, the optimal value shows slightly better performance than the case $\beta = 0$, and too large β may corrupt the accurate classification.

Since the proposed method uses K-means to select the initial basis vectors, the result may depend on the initial setting of K-means. In order to investigate this, we conducted experiment for the different random seed of K-means 100

Table 5 Multiclass classification problems of UCI datasets

No.	Name	No. of classes	No. of instances	No. of dimensions
1	Balance Scale	3	625	4
2	Ecoli	5	327	7
3	Glass Identification	6	214	9
4	Iris	3	150	4
5	Letter Recognition	26	20000	16
6	Thyroid (new-thyroid)	3	215	5
7	Optical Recognition	10	5620	64
8	Pen-Based Recognition	10	10992	16
9	Teaching Assistant Eval.	3	151	5
10	Wine	3	178	13

Table 6 Classification error rates and standard deviations of UCI benchmark datasets; '*' denotes it was significantly better ($p < 0.05$) between Proposed and SVM.

No.	Proposed	SVM1	ALSM	KCLA
1	2.30 ± 3.02	**1.35** ± 2.01	70.71 ± 10.42	**1.35** ± 1.65
2	15.30 ± 6.56	**13.94** ± 6.98	39.09 ± 9.17	15.00 ± 6.77
3	**33.18** ± 10.44	33.86 ± 8.53	55.00 ± 9.99	35.68 ± 7.84
4	**4.33** ± 4.97	**4.33** ± 4.97	48.67 ± 11.87	5.33 ± 5.56
5	***11.32** ± 0.44	11.71 ± 0.51	40.62 ± 1.86	11.52 ± 0.53
6	4.32 ± 5.41	**3.18** ± 5.34	39.77 ± 11.02	4.77 ± 4.77
7	**1.04** ± 0.39	1.06 ± 0.49	5.35 ± 0.90	1.49 ± 0.55
8	1.43 ± 0.31	***1.06** ± 0.49	12.78 ± 1.91	1.71 ± 0.26
9	***33.75** ± 9.60	45.63 ± 10.56	57.19 ± 10.78	36.25 ± 9.20
10	1.11 ± 2.90	1.67 ± 3.17	30.83 ± 12.42	**0.83** ± 2.04

times for each realization, and obtained the standard deviation of the error rates. The averaged standard deviations over the 20 realizations are PROl: 0.0667%, PROp: 0.0632%, and PROr: 0.0626%.

5.2 UCI Benchmark Datasets

The second experiment involved ten multiclass classification problems from The University of California, Irvine (UCI) machine learning repository. The datasets we used are listed in Table 5. For the Ecoli dataset, we removed three-classes omL, imL, and imS, because each of these had too few samples.

For preprocessing, we normalized the feature vectors to a unit norm. In a manner similar to the previous experiments, 90% of the samples were used for training, and the remaining 10% were used for testing; exceptions were the Letter and Pen-Based Recognition sets. This separation was random, and we tested 20 realizations. For the Letter and Pen-Based Recognition sets, since the number of samples was rather large, we used 10% of the samples for training, and the remaining 90% were used for testing.

The Gaussian kernel was used for all methods. The optimal parameters, such as the soft-margin parameter and the kernel parameter γ, were obtained from ten fold cross-validations. There were ten iterations, and the maximum basis size was fixed to be $M = 200$.

The classification error rates and standard deviations are listed in Table 6. The proposed method showed the best classification accuracy in four of the problems. The averaged numbers of basis (support) vectors are listed in Table 7.

Table 7 Average number of basis (support) vectors. nBV1 (nSV1): total number of the basis (support) vectors, nBV2 (nSV2): number of basis (support) vectors eliminating duplication

	Proposed		SVM1	
Dataset	nBV1	nBV2	nSV1	nSV2
1	461	260	178	113
2	635	286	255	166
3	345	169	520	154
4	64	46	93	49
5	3026	1461	4890	1633
6	351	165	80	44
7	1772	1156	4128	2019
8	936	503	1420	456
9	320	120	261	93
10	349	179	227	114

Table 8 Result of MNIST dataset: P200 and P500 denote the proposed method with $M = 200$ and $M = 500$ respectively.

	P200	P500	SVM1	KCLA	ALSM
Training runtime [s]	160	372	3822	150	95
Test runtime [s]	1.85	4.58	22.9	58.0	0.21
No. of BV1 (SV)	2000	5000	27142	60000	–

The proposed method was compared with KCLA by the Wilcoxon signed-ranks test. The averaged ranks for 10 (datasets) × 20 (resampling) were $R^+ = 7697$ (Proposed) and $R^- = 12403$ (KCLA), and the proposed method showed significantly lower error than KCLA ($p < 0.01$).

5.3 MNIST Dataset

In order to show the performance with a big dataset, we used MNIST (Mixed National Institute of Standards and Technology) handwritten dataset. The dataset consists of 60,000 training images and 10,000 test images. Each image is grayscale and contains 28×28 pixels. We vectorized and normalized to a unit norm. It should be noted that for this dataset, the optimal parameters were chosen by using test samples, and so the accuracy of the classification may not be used to evaluate the performance. However, it is worth comparing the computational complexity and required number of support vectors.

We compared the proposed method with $M = 200$, $M = 500$ (i.e. maximum nBV1 is 2000 or 5000), SVM1, and KCLA. The Gaussian kernel function was used for all methods. The results are shown in Table 8. The classification performance of the proposed method is comparable to that of SVM1, while the proposed method trains ten times faster than SVM1, and tests five times faster. The computational cost of KCLA is large compared to the other datasets because EVD costs $O(rN^2)$ in the training stage. In the test stage, for the proposed method and KCLA, the computational cost is proportional to N. If we do not use the basis selection method for the proposed method, the computational complexity of the proposed method will be more than ten times that of KCLA in the case of the number of iteration is ten.

6. Conclusion

We proposed a quadratic kernel learning subspace classification method that is based on ALSM [3] and subset KPCA [15], [16]. We (i) derived a learning algorithm, (ii) proposed a method for selecting basis vectors for the classifier, (iii) proposed a new rule to add enhancement and suppression sets, and (iv) explained the results of experiments conducted to demonstrate the effectiveness of the proposed method.

Compared to other kernel classification methods, such as the SVM, MKL, and RVM, the proposed method has the following advantages;

1. For each class, all samples in the class and a subset of samples that are not in the class are used for EVD. Therefore, even if the number of classes is large, the computational cost of the proposed method is relatively low compared to the one-vs-the-rest and one-vs-one approaches.
2. In the USPS dataset, the proposed method achieved a lower classification error while using fewer basis vectors. This is due to the quadratic decision function, Eq. (29). This reduces both the memory requirements and the computational complexity during the testing stage. Although there are many methods for reducing the number of support vectors [30]–[32], their classifications are not as accurate as that of the original SVM.
3. The limit on the maximum number of basis vectors is a tunable parameter. We can choose a suitable basis size depending on the problem. However, this can also be disadvantageous, since we must determine the optimal hyperparameters using cross-validation.
4. The main computations in the proposed method are a generalized EVD and the selection of the basis vectors. These are more easily implemented than convex quadratic optimization, which is used in SVM. Furthermore, estimation of the computational cost is easier than it is with convex quadratic optimization. In the MNIST experiment in Sect. 5.3, the proposed method trained ten times faster than SVM1 and tested five times faster.

On the other hand, there is a drawback to the proposed method in that it requires more hyperparameters than do the other methods. Most of them have to be tuned by cross-validations. This problem will be addressed in our future work.

Acknowledgments

This work was supported by Grant-in-Aid for Young Scientists (B) No. 24700163.

References

[1] S. Watanabe, Knowing & Guessing — Quantitative Study of Inference and Information, John Wiley & Sons, 1969.
[2] T. Iijima, H. Genchi, and K. Mori, "A theory of character recognition by pattern matching method," Proc. 1st Int. J. Conf. on Pattern Recognition, pp.50–56, 1973.
[3] E. Oja, Subspace Methods of Pattern Recognition, Wiley, New-York, 1983.
[4] O. Yamaguchi, K. Fukui, and K. Maeda, "Face recognition using temporal image sequence," Proc. Third International Conference on Automatic Face and Gesture Recognition, pp.318–323, 1998.
[5] K. Fukui and O. Yamaguchi, "Face recognition using multi-viewpoint patterns for robot vision," Proc. 11th International Symposium of Robotics Research (ISRR 03), vol.15, pp.192–201, 2003.
[6] K. Fukui and O. Yamaguchi, "Constrained mutual subspace method using a generalized difference subspace," IEICE Trans. Inf. & Syst. (Japanese Edition), vol.J87-D-II, no.8, pp.1622–1631, 2004.
[7] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," IEEE Trans. Pattern Anal. Mach. Intell., vol.32, no.9, pp.1627–1646, 2010.
[8] B. Schölkopf, A. Smola, and K.-R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," Neural Computation, vol.10, no.5, pp.1299–1319, 1998.
[9] K. Tsuda, "Subspace classifier in the Hilbert space," Pattern Recognition Letters, vol.20, no.5, pp.513–519, 1999.
[10] E. Maeda and H. Murase, "Multi-category classification by kernel based nonlinear subspace method," IEEE International Conference On Acoustics, speech, and signal processing (ICASSP), pp.1025–1028, IEEE press., 1999.
[11] Y. Washizawa and Y. Yamashita, "Kernel projection classifiers with suppressing feature of other classes," Neural Computation, vol.18, no.8, pp.1932–1950, 2006.
[12] M. Ichino, H. Sakano, and N. Komatsu, "Speaker recognition using kernel mutual subspace method," International Conference on Control, Automation, Robotics and Vision, pp.397–402, 2004.
[13] H. Sakano, N. Mukawa, and T. Nakamura, "Kernel mutual subspace method and its application for object recognition," Electronics and Communications in Japan, vol.E88, no.6, pp.45–53, 2005.
[14] K. Fukui and O. Yamaguchi, "The kernel orthogonal mutual subspace method and its application to 3D object recognition," Lecture Notes in Computer Science, vol.4844, pp.467–476, 2007.
[15] Y. Washizawa, Subset basis approximation of kernel principal component analysis, ch. 4, pp.67–90, InTech, 2012.
[16] Y. Washizawa, "Subset kernel principal component analysis," IEEE International Workshop on Machine Learning for Signal Processing (MLSP2009), pp.1–6, 2009.
[17] Y. Washizawa and Y. Yamashita, "Kernel projection classifiers with suppressing feature of other classes," Neural Computation, vol.18, no.8, pp.1932–1950, 2006.
[18] D. Harville, Matrix Algebra from a Statistician's Perspective, Springer-Verlag, New York, 1997.
[19] V.N. Vapnik, Statistical Learning Theory, Wiley, New-York, 1998.
[20] M. Tipping, "Sparse baysian learning and the relevance vector machine," J. Machine Learning Research, vol.1, pp.211–244, 2001.
[21] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," J. Machine Learning Research, vol.12, pp.2211–2268, 2011.
[22] C.-W. Hsu and C.-J. Lin, "A comparison of mathods for multiclass support vector machines," IEEE Trans. Neural Netw., vol.13, no.2, pp.415–425, 2002.
[23] C. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
[24] K. Crammer and Y. Singer, "On the algorithmic implementation of multi-class SVMs," J. Machine Learning Research, vol.2, pp.265–292, 2001.
[25] S. Haykin, Neural Networks, a Comprehensive Foundation, Prentice-Hall, 1999.
[26] B. Schölkopf, J.C. Platt, J. S.-Taylor, A.J. Smola, and R.C. Williamson, "Estimating the support of a high-dimensional distribution," Neural Computation, vol.13, no.7, pp.1443–1471, 2001.

[27] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel, "Backpropagation applied to handwritten zip code recognition," Neural Computation, vol.1, no.4, pp.541–551, 1989.
[28] A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet, "More efficiency in multiple kernel learning," International Conference on Machine Learning (ICML 2007), pp.775–782, 2007.
[29] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," ACM Transactions on Intelligent Systems and Technology, vol.2, no.27, pp.1–27, 2011.
[30] M. Wu, B. Schölkpf, and G. Bakir, "Building sparse large margin classifiers," International Conference on Machine Learning (ICML 2005), pp.996–1003, 2005.
[31] Y.J. Lee and O. Mangasarian, "RSVM: reduced support vector machines," First SIAM International Conference on Data Mining, pp.0–7, 2001.
[32] K.-M. Lin and C.-J. Lin, "A study on reduced support vector machines," IEEE Trans. Neural Netw., vol.14, no.6, pp.1449–1459, 2003.

Yoshikazu Washizawa received the B.E. degree from the Nagoya Institute of Technology in 2002, the M.E. and Ph.D. (Dr. Eng) degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 2004 and 2008, respectively. From 2004 to 2005, he was with the TOSHIBA corporation. In 2005, he was with the Tokyo Institute of Technology as a JSPS Research Fellow. From 2005 to 2011, he was with the Laboratory for Advanced Brain Signal Processing at the Brain Science Institute, RIKEN, Saitama, Japan. Since 2011, he has been an Assistant Professor with the University of Electro-Communications, Japan, and a Visiting Researcher with the Laboratory for Advanced Brain Signal Processing at the Brain Science Institute, RIKEN. His research interests include pattern recognition, machine learning, (bio-medical) signal processing. He is a member of IEEE and IEICE.