TAKES: A Fast Method to Select Features in the Kernel Space*

Ye Xu[1], Furao Shen[2], Wei Ping[3], and Jinxi Zhao[2]
Computer Science Department, Dartmouth College, NH, USA[1]
ye@cs.dartmouth.edu
National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China[2]
{frshen,jxzhao}@nju.edu.cn
Department of Computer Science, University of California Irvine, CA, USA[3]
weiping.thu@gmail.com

ABSTRACT

Feature selection is an effective tool to deal with the "curse of dimensionality". To cope with the non-separable problem, feature selection in the kernel space has been investigated. However, previous study cannot adequately estimate the intrinsic dimensionality of the kernel space. Thus, it is difficult to accurately preserve the sketch of the kernel space using the learned basis, and the feature selection performance is affected. Moreover, the computing load of the algorithm reaches at least cubic with the number of training data. In this paper, we propose a fast framework to conduct feature selection in the kernel space. By designing a fast kernel subspace learning method, we automatically learn the intrinsic dimensionality and construct an orthogonal basis set of kernel space. The learned basis can accurately preserve the sketch of kernel space. Then backed by the constructed basis, we directly select features in kernel space. The whole proposed framework has a quadratic complexity with the number of training data, which is faster than existing kernel methods for feature selection. We evaluate our work under several typical datasets and find it not only preserves the sketch of the kernel space more accurately but also achieves better classification performance compared with many state-of-the-art methods.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Induction; I.5.4 [**Pattern Recognition**]: Application

General Terms

Theory, Algorithm, Performance

Keywords

Feature Selection, Kernel Space, Orthogonal Subspace Learning

*This paper was done when Ye Xu was affiliated with Nanjing University.

CIKM '11 October 24–28, 2011, Glasgow, Scotland, UK.

1. INTRODUCTION

The "curse of dimensionality" is a serious problem for classification tasks involving high-dimensional data. Reducing the dimensionality is an important tool to address this problem. By reducing the dimensionality, we obtain a reduced representation of input data that is much smaller in volume, and produce the same (or almost the same) analytical result.

Generally, techniques for reducing dimensionality can be categorized into two classes, i.e., dimensionality reduction and feature selection. Some typical dimensionality reduction methods, such as Principal Component Analysis (PCA) [9], Linear Discriminant Analysis (LDA) [21], Maximum Margin Criterion (MMC) [17], Locally Linear Embedding (LLE) [24], ISOMAP[31] and their invariants [8, 38, 37, 19, 23, 39, 5], try to use transformations to convert original high-dimensional data space into lower-dimensional feature space, which have been proved to be effective. However, the high dimensionality of data sets often fails many such algorithms due to their high computational cost.

In this paper we focus on feature selection techniques. Feature selection algorithms (e.g., [15, 40, 32, 33, 29, 14, 36]) directly remove non-discriminative features in order to build robust learning models based on some certain criterions, which attract a variety of attentions [16, 28, 12, 25].

Traditional methods for reducing dimensionality may work well in linear separable problems. However, for nonlinear problems, the performance may be affected. Therefore, kernel technique is proposed to tackle this issue. Some traditional dimensionality reduction methods have been extended into kernel version [22, 30, 1] and achieve sweet results.

For feature selection methods, some works [20, 18] have been done to employ kernels to help select features in original space, which improve classification performance to some extent. However, such algorithms still select features in the original space, which can not guarantee a good performance for the nonlinear problems. Because the dimensionality of the kernel space is implicit, it is a big challenge to directly select features in the kernel space. It is no wonder that few works can handle this thorny issue.

In this paper, we propose a feature selection method in the kernel space named as Two-stage fAst Kernel fEature Selection (TAKES). In the first stage, we design a fast kernel subspace learning method to learn the intrinsic dimensionality of the kernel space, and construct an orthogonal basis. Then in the second stage, via the learned orthogonal basis, we directly select discriminative features in the kernel space using the kernel trick. In summary, the first contribution of this paper is that, we design a framework to directly select

discriminative features in the kernel space. The classification performance under the selected features is better. The second contribution of our work is designing a kernel subspace learning algorithm to automatically estimate the intrinsic dimensionality of the kernel space and construct an orthogonal basis. The learned dimensionality of the kernel space is completely data-dependent and a small reconstruction cost [24] is guaranteed. Here, the reconstruction cost is a measure to evaluate whether the learned basis can accurately preserve the sketch of data [10] in the kernel space. The third contribution is the small computing load compared with other kernel methods for reducing dimensionality.

The rest of this paper is organized as follows. In section 2, we briefly discuss some related works. Our TAKES method is proposed in section 3. In section 4, we report on experimental results. Finally in Section 5, we conclude the paper.

2. RELATED WORK

Feature selection algorithms have been investigated for many years. Because obtaining an optimum set of features has been demonstrated as an NP-hard problem [35], many algorithms handle feature selection tasks by introducing weights on dimensionality to obtain approximate solutions. Such methods can be divided into several categories. Based on the assumption that the discriminative power of each feature is independent of each other, Information Gain (IG) [33] and χ^2-test [40] measure the amount of information obtained by each feature for classification and then discard those features whose discriminative powers are small. But these methods might fail when features are not independent of each other. Relief [15, 29] is another category feature selection algorithm. It estimates features according to how well their values distinguish among instances that are near each other, where smart heuristic is used. Boosting methods [32] have been used for feature selection, in which discriminative features are selected and combined into an ensemble classifier. The generalization error of boosting family algorithms [14] has been proven to be controlled by a bound in terms of a margin measure [34]. Therefore, such algorithms have better performance and wide applications [16, 28]. To improve the time complexity of boosting methods, Forward Feature Selection (FFS) [36] has been proposed. Unlike many other boosting algorithms, FFS trains weak classifiers only once, and thus is able to obtain lower time complexity (O(dN)) compared with most boosting family methods (O($dNlogN$)). Here, N is the size of the training set, and d is the number of selected features.

Our work also relates to kernel methods for feature selection [20, 18]. Most of such methods still select features in the original space, which cannot change the non-separability nature of the data. A better solution is directly selecting features in kernel space. To the best of our knowledge, [6] is the only work that selects features in the kernel space. In this work, by learning a basis for the kernel space, Relief [15] is employed to select features. However, the major shortcoming of [6] is that, the intrinsic dimensionality of the kernel space in their work is constantly set up as the size of training set. It is not accurate unless all mapped training data in the kernel space are independent of each other. However, they are not necessarily independent. Therefore, this heuristically determined dimensionality undermines the orthogonality of learned basis. In this case, the basis set cannot accurately preserve the sketch of the kernel space; the feature selection performance is influenced. Another limitation of [6] is the computing load. By employing a kernel subspace learning method such as Kernel Gram-Schmidt (KGS) or Kernel Principal Component Analysis (KPCA) [26] to construct the basis, the time complexity is at least $O(N^3)$.

In our work, we aim at selecting features in the kernel space. First we design a fast kernel subspace learning method to learn the intrinsic dimensionality of the kernel space, and construct an orthogonal basis to accurately preserve the sketch of the mapped data in the kernel space. Then with the aid of the basis, discriminative features in the kernel space are directly selected with a smaller computing load.

3. SELECT FEATURES IN THE KERNEL SPACE

Traditional feature selection methods may not work well in linearly non-separable problems. To overcome the shortcoming, in this paper, we select discriminative features directly from the kernel space instead of from the original data space.

Suppose ϕ maps input data $\boldsymbol{x} \in \mathcal{R}^D$ to a $\mathcal{H}$-dimensional kernel space:

$$\phi : \boldsymbol{x} \rightarrow [\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), ...\phi_{\mathcal{H}}(\boldsymbol{x})]^\top \tag{1}$$

In order to directly select discriminative features in the kernel space, we need to know the value of $\phi_i(\boldsymbol{x})$, i.e., the i^{th} feature of $\phi(\boldsymbol{x})$. However, in the kernel space, $\phi(\boldsymbol{x})$ and the number of features $\mathcal{H}$ are implicit. Thus it is difficult for us to achieve $\phi_i(\boldsymbol{x})$ explicitly.

To address this problem, we construct an orthogonal basis set $\{\mathbf{w_i}\} = \{\sum_{j=1}^{N} \hat{\alpha}_{i,j}\phi(\boldsymbol{x_j})\}$ for the kernel space, where $\{\boldsymbol{x_j}\}_{j=1}^{N}$ are training data, $\hat{\alpha}_i$ are the coefficient vectors, and $\hat{\alpha}_{i,j}$ is the j^{th} component of $\hat{\alpha}_i$. Then $\phi_i(\boldsymbol{x})$ can be calculated by using the kernel trick:

$$\phi_i(\boldsymbol{x}) =< \mathbf{w_i}, \phi(\boldsymbol{x}) >= \sum_{j=1}^{N} \hat{\alpha}_{i,j}\phi(\boldsymbol{x_j})^\top \phi(\boldsymbol{x}) = \sum_{j=1}^{N} \hat{\alpha}_{i,j} K(\boldsymbol{x_j}, \boldsymbol{x}) \tag{2}$$

Therefore, a key issue for feature selection in the kernel space is to properly estimate the intrinsic dimensionality and find an orthogonal basis of the kernel space, so that the mapped data in the kernel space can be accurately represented. In other words, the $\phi_i(\boldsymbol{x})$ computed by (2) can be accurate only if the sketch of the kernel space is accurately preserved by the learned basis .

The dimensionality of basis set for kernel space is the rank of mapped data set $\{\phi(\boldsymbol{x_j})\}_{j=1}^{N}$. This rank equals to the numerical rank of kernel matrix $\mathbf{K}$, whose entries are defined by:

$$K_{ij} =< \phi(\boldsymbol{x_i}), \phi(\boldsymbol{x_j}) >= K(\boldsymbol{x_i}, \boldsymbol{x_j}) \tag{3}$$

Computing the rank of a matrix (i.e. the intrinsic dimensionality of kernel space) is a difficult issue, which has a time complexity of $O(N^3)$ [13]. It is, however, not an ideal choice to predetermine the target dimensionality as a constant, either. If it is too small, the learned basis will have a large distinction from the kernel space. On the contrary, a too large predetermined dimensionality, as adopted by [6], renders it difficult in learning orthogonal base vectors and aggravates computing load. Take the Hilbert Matrix [7] for example:

$$H_{ij} = (\frac{1}{i+j-1}) \tag{4}$$

The $n \times n$ Hilbert matrix $H(n)$ is a positive definite symmetric matrix. It means that, theoretically speaking, the dimensionality of the space that is spanned by $H(n)$'s column vectors should be n. However, in a case study, we validate that there are only about 18 linear independent base vectors in the space spanned by $H(100)$'s 100 column vectors. If we set up the dimensionality of the space spanned by $H(100)$'s column vectors as 100, many learned base vectors of this space are numerically non-orthogonal and the too

large predetermined dimensionality increases the computing load of the algorithm. Therefore, under both of the two conditions, the learned basis cannot accurately preserve the sketch of the kernel space and the performance of feature selection is undermined.

To tackle this problem, we design a fast kernel subspace learning method – Kernel Orthogonal Basis Learner (KOBL). KOBL automatically learns the intrinsic dimensionality of the kernel space, and constructs an orthogonal basis set with a small reconstruction cost [24]. Here, reconstruction cost is employed to measure the quality of the learned basis set. A small value of reconstruction cost means the learned basis set can accurately preserve the sketch of the data that are used to learn the basis, i.e., the quality of learned basis is high. By designing a stop criterion, KOBL has a time complexity of $O(dN^2)$, which outperforms many other kernel subspace learning methods such as KGS and KPCA (at least $O(N^3)$). Backed by a high quality basis learned by KOBL, our feature selection work achieves better performance. In the rest of Section 3, we give the detailed TAKES.

3.1 Learn the Intrinsic Dimensionality and Orthogonal Basis for the Kernel Space

In this part, we propose the KOBL algorithm to learn the intrinsic dimensionality of feature space and construct an orthogonal basis from the mapped training data.

At first, we initialize the data matrix $A^{(0)} = [\phi(\boldsymbol{x}_1)^{(0)}, \phi(\boldsymbol{x}_2)^{(0)}, ..., \phi(\boldsymbol{x}_N)^{(0)}]$, basis $\mathbf{M}$ of the kernel space to empty basis and its dimensionality $k = Dim(\mathbf{M}) = 0$. Therein, $\{\phi(\boldsymbol{x}_i)^{(0)}\}_{i=1}^{N}$ are mapped data in the kernel space. (In this paper, we use $\phi(\boldsymbol{x}_i)^{(0)}$ and $\phi(\boldsymbol{x}_i)$ interchangeably.)

In every iteration, we first find the $(k+1)^{th}$ pivot column as follows (Suppose the former k iterations have finished and the former k base vectors have been obtained.):

$$j = \underset{k<i\leq N}{\operatorname{argmax}} \|\phi(\boldsymbol{x}_i)^{(k)}\|_2 \tag{5}$$

Here, j is the column that has the largest column norm from the $(k+1)^{th}$ column to the N^{th} column.

Then we interchange the pivot column j with column $k+1$, where k is the current number of the learned base vectors. After the process of column pivoting and exchange, we calculate the norm of the $(k+1)^{th}$ column vector $r_{k+1,k+1}$ by:

$$r_{k+1,k+1} = \|\phi(\boldsymbol{x}_{k+1})^{(k)}\|_2. \tag{6}$$

And we obtain the normalized potential base vector $\mathbf{w_{k+1}}$ by

$$\mathbf{w_{k+1}} = \frac{\phi(\boldsymbol{x}_{k+1})^{(k)}}{r_{k+1,k+1}}. \tag{7}$$

Here, the potential base vector means the learned base vectors that may be accepted or rejected determined by a threshold policy that is discussed later. If $\mathbf{w_{k+1}}$ is accepted, the data matrix $A^{(k)}$ is updated using orthogonal transformation as follows,

$$\phi(\boldsymbol{x}_j)^{(k+1)} \leftarrow \phi(\boldsymbol{x}_j)^{(k)} - r_{k+1,j}\mathbf{w_{k+1}}, j = k+2 : N. \tag{8}$$

Therein, $r_{k+1,j}$ can be computed by

$$r_{k+1,j} = \mathbf{w_{k+1}}^{\top}\phi(\boldsymbol{x}_j)^{(k)}, j = k+2 : N. \tag{9}$$

Now, $A^{(k+1)} = [\mathbf{w_1}, ..., \mathbf{w_{k+1}}, \phi(\boldsymbol{x}_{k+2})^{(k+1)}, ..., \phi(\boldsymbol{x}_N)^{(k+1)}]$.

On the contrary, if the threshold condition does not satisfy, we will not accept $\mathbf{w_{k+1}}$.

Note that the kernel map ϕ is unseen, and the form of basis $\{\mathbf{w_j}\}$ is implicit. Therefore, using the kernel trick described in (2), the norm of mapped data in (5) can be computed by the following equation,

$$\begin{aligned}\|\phi(\boldsymbol{x}_i)^{(k)}\|_2 &= \|\phi(\boldsymbol{x_i})^{(0)} - \sum_{j=1}^{k}(\mathbf{w_j}^{\top}\phi(\boldsymbol{x_i})^{(0)})\mathbf{w_j}\|_2 \\ &= \sqrt{\mathbf{K}(\boldsymbol{x_i},\boldsymbol{x_i}) - \sum_{j=1}^{k}(\sum_{m=1}^{N}\hat{\alpha}_{j,m}\mathbf{K}(\boldsymbol{x_i},\boldsymbol{x_m}))^2}\end{aligned} \tag{10}$$

Therein, $\hat{\alpha}_1, ..., \hat{\alpha}_k$ are the k coefficient vectors achieved in the former k iterations.

Because the norm of $\mathbf{w_{k+1}}$ is

$$\begin{aligned}\|\mathbf{w_{k+1}}\|_2 = \|\phi(\boldsymbol{x}_{k+1})^{(k)}\|_2 = \\ \|\sum_{i=1}^{N}\alpha_{k+1,i}^{(k)}\phi(\boldsymbol{x}_i)\|_2 = \sqrt{{\alpha_{k+1}^{(k)}}^{\top}\mathbf{K}\alpha_{k+1}^{(k)}},\end{aligned} \tag{11}$$

we implicitly achieve the $(k+1)^{th}$ normalized base vector $\mathbf{w_{k+1}}$ by setting the $(k+1)^{th}$ coefficient vectors[1] ,

$$\hat{\alpha}_{k+1} = \alpha_{k+1}^{(k+1)} = \frac{\alpha_{k+1}^{(k)}}{\sqrt{{\alpha_{k+1}^{(k)}}^{\top}\mathbf{K}\alpha_{k+1}^{(k)}}} \tag{12}$$

Similarly, (8) is fulfilled via updating the coefficient vectors as follows,

$$\alpha_j^{(k+1)} \leftarrow \alpha_j^{(k)} - \hat{\alpha}_{k+1}\sum_{i=1}^{N}\alpha_{k+1,i}^{(k)}\mathbf{K}(\boldsymbol{x_j},\boldsymbol{x_i}), j = k+2 : N \tag{13}$$

It deserves mentioning that most kernel subspace learning methods such as KGS, KPCA and their invariants [27] ceaselessly learn the basis until all training data are used, which suffers from heavy computing time. In KOBL, a stop criterion is designed to overcome this shortcoming. In the rest of this part, we analyze KOBL algorithm in detail.

3.1.1 Independence Measurement and Learn Orthogonal Basis

To learn an orthogonal basis, KOBL takes the independence between the learned base vectors and the mapped data in the kernel space into consideration. Suppose we have learned basis $\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}$ sequentially by orthogonal transformations. If a mapped data $\phi(\boldsymbol{x}_{k+1})$ is dependent on the learned basis, the corresponding computed vector $\mathbf{w_{k+1}}$ is unsuitable to be accepted. It is because that if a base vector is transformed from a pattern that has strong dependence on the learned basis, the computation error would arise and affect the numerical orthogonality of the learned basis [13]. Therefore, it is necessary to consider the independence between the learned basis and the mapped data.

Here, based on the linear dependence theorem [13], we use the projection distance of a mapped data $\phi(\boldsymbol{x}_{k+1})$ onto the current learned basis space $span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}$ to measure the numerical independence between $\phi(\boldsymbol{x}_{k+1})$ and the learned basis for the kernel space: (Suppose we have already learned the base vectors $\mathbf{w_1}, \mathbf{w_2}, ...,$ and $\mathbf{w_k}$. The potential base vector $\mathbf{w_{k+1}}$ is currently computed by (7).)

$$\|\phi(\boldsymbol{x}_{k+1}) - span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}\|_2 \tag{14}$$

[1] At the beginning of the algorithm, the coefficient vectors are simply initialized as the unit vectors: $\alpha_j^{(0)} = \boldsymbol{e_j}$, for $j = 1 : N$. Therein, $\boldsymbol{e_j}$ is the vector whose j^{th} component is 1 and the rest components are 0.

The distance in (14) can be rewritten in the following form:

$$\min_{\hat{\mathbf{a}}} \|\phi(\boldsymbol{x}_{k+1}) - (a_1\mathbf{w_1} + a_2\mathbf{w_2} + ... + a_k\mathbf{w_k})\|_2 \qquad (15)$$

where $\hat{\mathbf{a}} = (a_1, a_2, ..., a_k)^\top$ is a coordinate vector.

We define matrix $\mathbf{W_k} = [\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}]$, thus (14) can be rewritten in the form:

$$\min_{\hat{\mathbf{a}}} \|\mathbf{W_k}\hat{\mathbf{a}} - \phi(\boldsymbol{x}_{k+1})\|_2 \qquad (16)$$

The column vectors of $\mathbf{W_k}$ are orthogonal. Thereby, we can construct an orthogonal matrix $\mathbf{Q_2}$ in the following form: $\mathbf{Q_2} = [\mathbf{W_k}, \mathbf{Q_1}]$.Therein, $\mathbf{Q_2} \in \mathbb{R}^{\mathcal{H}\times\mathcal{H}}$, $\mathbf{Q_1} \in \mathbb{R}^{\mathcal{H}\times(\mathcal{H}-k)}$, and the column vectors of $\mathbf{Q_1}$ and $\mathbf{W_k}$ are orthogonal.

Since orthogonal transformation never changes the 2-norm of a vector, (16) is equal to the following expression:

$$\begin{aligned}\|\mathbf{W_k}\hat{\mathbf{a}} - \phi(\boldsymbol{x}_{k+1})\|_2 &= \|\mathbf{Q_2}^\top(\mathbf{W_k}\hat{\mathbf{a}} - \phi(\boldsymbol{x_{k+1}}))\|_2 \\ &= \|\begin{pmatrix}\hat{\mathbf{a}}\\ \mathbf{0}\end{pmatrix} - \begin{pmatrix}\mathbf{W_k}^\top\phi(\boldsymbol{x}_{k+1})\\ \mathbf{Q_1}^\top\phi(\boldsymbol{x}_{k+1})\end{pmatrix}\|_2 \qquad (17)\end{aligned}$$

We let

$$\hat{\mathbf{a}} = \mathbf{W_k}^\top\phi(\boldsymbol{x_{k+1}}), \qquad (18)$$

thereby, the value of (14) is equal to $\|\mathbf{Q_1}^\top\phi(\boldsymbol{x}_{k+1})\|_2$, i.e., the independence between the learned base vectors and $\phi(\boldsymbol{x_{k+1}})$ can be measured by the value of $\|\mathbf{Q_1}^\top\phi(\boldsymbol{x_{k+1}})\|_2$.

According to (6) and (8), it is straightforward to validate that $r_{k+1,k+1}$ is the projection distance of initial mapped data $\phi(\boldsymbol{x}_{k+1})$ onto the orthogonal complement space $span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}^\perp$, i.e., $r_{k+1,k+1} = \|\mathbf{Q_1}^\top\phi(\boldsymbol{x}_{k+1})\|_2$. Therefore, we can determine whether to accept the computed vectors $\mathbf{w_{k+1}}$ into current basis $\mathbf{M}$ or not according to the value of $r_{k+1,k+1}$. As illustrated in Fig.1, If $r_{k+1,k+1}$ is small, it means that the mapped data $\phi(\boldsymbol{x}_{k+1})$ is nearly or even completely in the space spanned by the base vectors $span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}$. In this case, $\phi(\boldsymbol{x}_{k+1})$ has a strong dependence on $span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}$, and it is not reasonable to add the potential base vector $\mathbf{w_{k+1}}$ into the basis $\mathbf{M}$ since $\mathbf{w_{k+1}}$ is transformed from $\phi(\boldsymbol{x}_{k+1})$. On the contrary, when $r_{k+1,k+1}$ is large, $\phi(\boldsymbol{x}_{k+1})$ is orthogonal to $span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}$. It means that $\phi(\boldsymbol{x}_{k+1})$ is independent of $span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}$, and $\mathbf{w_{k+1}}$ is an ideal base vector.

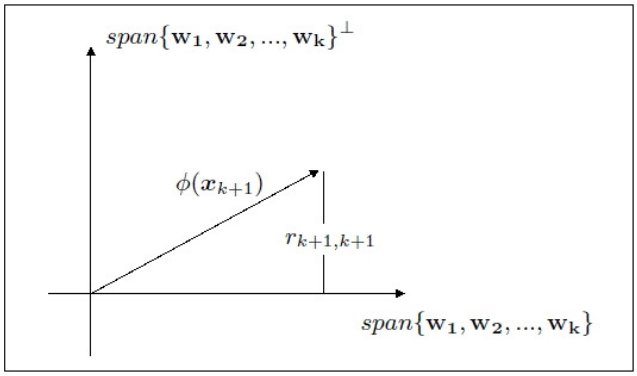

Figure 1: $\phi(\boldsymbol{x}_{k+1})$ is projected onto two orthogonal spaces: $span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}$ and $span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}^\perp$. The projection distance onto the space $span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}\}^\perp$ is $r_{k+1,k+1}$. If $r_{k+1,k+1}$ is large, we will accept $\mathbf{w_{k+1}}$ as an ideal base vector. Otherwise we have to discard it.

3.1.2 Learn Intrinsic Dimensionality for the Kernel Space with Small Reconstruction Cost

As discussed above, to keep the independence of learned basis and mapped data, we make the decision of whether to accept the computed vectors $\mathbf{w_{k+1}}$ into basis according to the value of $r_{k+1,k+1}$. In KOBL, we use a threshold T to make the decision: if $r_{k+1,k+1} \geq T$, we accept the potential vector $\mathbf{w_{k+1}}$ and vice versa. The value of r_{11} is not smaller than r_{ii} for all $i > 1$. (It will be demonstrated in Theorem 2.) Therefore, for the purpose of convenient comparison, instead of $r_{k+1,k+1}$, we use $\frac{r_{k+1,k+1}}{r_{11}}$ to make the final decision.

The learned dimensionality for the kernel space equals to the number of accepted base vectors. Therefore, the value of threshold T plays an important role in estimating the intrinsic dimensionality of kernel space and affects the orthogonality of the learned basis. If the estimated dimensionality is too small, the learned basis will have a large distinction from the kernel space. On the contrary, a too large estimated dimensionality will render it difficult in learning orthogonal base vectors. Take the Hilbert Matrix [7] for example. If we simply set up the dimensionality of the space spanned by $H(100)$'s column vectors as 100, many base vectors learned are not orthogonal. Under both of the two cases, using (2) to compute $\phi_i(\boldsymbol{x})$ is not accurate, so that the performance of feature selection would be affected.

Many current subspace learning methods (e.g, KPCA and KGS) need a user to predetermine the intrinsic dimensionality for the kernel space. Although for KPCA and some KPCA based methods such as Kernel Feature Analysis [27], the intrinsic dimensionality for the kernel space can be determined based on the accumulation ratio whose lower bound comes from the tolerable approximation error, this lower bound has to be predetermined by users as parameters, which may still lead to an improper dimensionality for the kernel space. On the other hand, there are some reported schemes such as Relevant Dimension Estimation(RDE) [3, 4] that can provide an estimation for relevant dimensionality in the kernel space with respect to a supervised learning task. However, such schemes themselves suffer from heavy computational cost ($O(N^3)$) which is impractical.

In KOBL, instead of employing an user defined threshold, we aim at proposing an adaptive threshold scheme to learn the intrinsic dimensionality of kernel space without adding much extra computing burden.

Besides ensuring the orthogonality of the basis, we hope the adaptive threshold scheme to guarantee that the learned basis can preserve the sketch of the kernel space, namely, the reconstruction cost $\mathcal{E}(\mathbf{M})$ defined as follows is small.

$$\begin{aligned}\mathcal{E}(\mathbf{M}) &= \frac{1}{N}\sum_{j=1}^{N}\|\phi(\boldsymbol{x}_j) - span\{\mathbf{M}\}\|_2 \\ &= \frac{1}{N}\sum_{j=1}^{N}\|\phi(\boldsymbol{x}_j) - \sum_{i=1}^{d}(\mathbf{w_i}^\top\phi(\boldsymbol{x}_j))\mathbf{w_i}\|_2 \qquad (19)\end{aligned}$$

Therein, $\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_d}$ are all learned base vectors in basis set $\mathbf{M}$. Moreover, we hope that the threshold T is confined between 0 and 1, so that T is comparable to $\frac{r_{k+1,k+1}}{r_{11}}$. ($\frac{r_{k+1,k+1}}{r_{11}}$ varies between 0 and 1.)

To satisfy the above constraints, we propose a self-adaptive threshold: we accept the potential base vector $\mathbf{w_{k+1}}$ if

$$\frac{r_{k+1,k+1}}{r_{11}} \geq T = f(\frac{Dim(\mathbf{M})}{N}) \qquad (20)$$

Therein, $Dim(\mathbf{M})$ is the currently learned dimensionality of the

kernel space, and $f(t)$ is a monotonically increasing function and $0 \leq f(t) \leq 1$ when $0 \leq t \leq 1$.

Now, in the following theorem, we show how the threshold policy defined by (20) can ensure a small reconstruction cost.

THEOREM 1. *The value of $\mathcal{E}(\mathbf{M})$ for OCA has an upper bound of $r_{11}f(\frac{Dim(\mathbf{M})}{N})$, i.e., $\mathcal{E}(\mathbf{M}) < r_{11}f(\frac{Dim(\mathbf{M})}{N})$.*

PROOF. Suppose finally we obtain d base vectors for the kernel space: $\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_d}$.

Therefore, for those mapped data $\phi(\boldsymbol{x}_j)$ $(j \leq d)$, we have

$$\|\phi(\boldsymbol{x}_j) - span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_d}\}\|_2 = \sqrt{\|\phi(\boldsymbol{x}_j)\|_2^2 - \sum_{i=1}^{d}(\mathbf{w_i}^\top \phi(\boldsymbol{x}_j))^2} = 0$$

As for those mapped data $\phi(\boldsymbol{x}_j)$ $(j > d)$, due to the non-increasing monotonicity of r_{jj}, (which is demonstrated in Theorem 2,) we achieve the following equations,

$$\|\phi(\boldsymbol{x}_j) - span\{\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_d}\}\|_2 = \|\phi(\boldsymbol{x}_j) - \sum_{i=1}^{d}(\mathbf{w_i}^\top \phi(\boldsymbol{x}_j))\mathbf{w_i}\|_2 \leq r_{d+1,d+1}$$

As a result, $\mathcal{E}(\mathbf{M}) \leq \frac{1}{N}(N-d)r_{d+1,d+1}$. From (20), we know that,

$$\mathcal{E}(\mathbf{M}) < \frac{N-d}{N} r_{11} f(\frac{d}{N}) < r_{11} f(\frac{Dim(\mathbf{M})}{N})$$

which completes the proof. □

Since $f(t) \leq 1$ is a monotonically increasing function, and $Dim(\mathbf{M}) << N$, the value of $r_{11}f(\frac{Dim(\mathbf{M})}{N})$ would be small. It means the adaptive threshold technique ensures that the basis obtained by KOBL can accurately preserve the sketch of the kernel space, which meets our demand.

$N+1$ vectors of N dimensionality must be linearly dependent and non-orthogonal. The number of base vectors is always smaller than the dimensionality of input data, i.e., $Dim(\mathbf{M}) \leq N$. Therefore, the value of T is confined between 0 and 1.

We should notice that all elements in this threshold policy, namely $Dim(\mathbf{M})$, N, $r_{k+1,k+1}$, and r_{11}, are easily available and automatically updated during the learning process. It means that the threshold scheme is self-adaptive, and costs little extra computing load.

The very threshold policy helps KOBL learn the intrinsic dimensionality of the kernel space and construct an orthogonal basis with small reconstruction cost.

3.1.3 Decrease Time Complexity by Designing a Stop Criterion

Most subspace learning methods use all the data to learn the basis, which suffers from heavy computing load when the data volume is large. In KOBL, to address this problem, we propose a stop criterion based on the following theorem.

THEOREM 2. *If $j \leq k$ then $r_{jj} \geq r_{kk}$.*

PROOF. We denote the original data matrix $A^{(0)} = [\phi(\boldsymbol{x}_1)^{(0)}, \phi(\boldsymbol{x}_2)^{(0)}, ..., \phi(\boldsymbol{x}_N)^{(0)}]$, where N is the number of data. In every iteration of KOBL, the data matrix is transformed. Suppose we have learned k base vectors $\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_k}$ through k times iterations, then

$$A^{(k)} = [\mathbf{w_1}, ..., \mathbf{w_k}, \phi(\boldsymbol{x}_{k+1})^{(k)}, ..., \phi(\boldsymbol{x}_N)^{(k)}].$$

In the $(k+1)^{th}$ iteration of KOBL, we obtain the following equation:

$$\phi(\boldsymbol{x}_{k+1})^{(k+1)} = \mathbf{w_{k+1}} = \frac{\phi(\boldsymbol{x}_{k+1})^{(k)}}{r_{k+1,k+1}}$$

For all $j > k+1$, we have,

$$\phi(\boldsymbol{x}_j)^{(k+1)} = \phi(\boldsymbol{x}_j)^{(k)} - r_{k+1,j}\frac{\phi(\boldsymbol{x}_{k+1})^{(k)}}{r_{k+1,k+1}}$$

$\phi(\boldsymbol{x}_j)^{(k+1)}$ is orthogonal to $\mathbf{w_{k+1}}$, thus we can arrive at the following item:

$$\|\phi(\boldsymbol{x}_j)^{(k+1)}\|_2^2 = \phi(\boldsymbol{x}_j)^{(k)^\top}\phi(\boldsymbol{x}_j)^{(k)} - r_{k+1,j}^2 \leq \|\phi(\boldsymbol{x}_j)^{(k)}\|_2^2$$

Due to the column pivoting scheme, we have

$$\|\phi(\boldsymbol{x}_j)^{(k)}\|_2^2 \leq \|\phi(\boldsymbol{x}_{k+1})^{(k)}\|_2^2 = r_{k+1,k+1}^2$$

for all $j > k+1$, and

$$r_{k+2,k+2} = \max_{j \geq k+2} \|\phi(\boldsymbol{x}_j)^{(k+1)}\|_2.$$

Thereby, we obtain $r_{k+1,k+1} \geq r_{k+2,k+2}$, which completes the proof. □

According to the above theorem, we develop a stop criterion in KOBL: we stop the algorithm if we arrive at the first computed vector $\mathbf{w_d}$ satisfying the following inequality:

$$\frac{r_{dd}}{r_{11}} < T = f(\frac{Dim(\mathbf{M})}{N}) \tag{21}$$

Suppose $\mathbf{w_d}$ is the first computed vector that does not satisfy the threshold condition, i.e., $\frac{r_{dd}}{r_{11}} < T$. Obviously the threshold T is monotone nondecreasing, and $\frac{r_{dd}}{r_{11}}$ is monotone nonincreasing during training, as indicated in Theorem 2. Therefore, if the integer d does not satisfy $\frac{r_{dd}}{r_{11}} \geq T$, all the integers $i > d$ do not satisfy the threshold condition, either. It means even if we use the remaining training data to learn the basis, the potential base vectors learned later ($\{\mathbf{w_i}\}_{i=d+1}^{N}$) do not meet the threshold condition and will be rejected. Thus, we don't need to continue learning the basis any more.

Based on the stop criterion (21), KOBL is able to realize a low computing complexity. Without this stop criterion, KOBL would have to process all training data; the time complexity would reach $O(N^3)$. By the stop criterion, the time complexity reduces to $O(N^2 d)$. Here, N is the size of training set, and d is number of base vectors. Usually $d << N$, thus the stop criterion saves a lot of computing time, and helps KOBL outperform those typical kernel subspace learning methods such as KGS ($O(N^3)$) and KPCA ($O(N^3 + N^2 d)$).

According to the above discussion, we give the detailed KOBL in Algorithm 1.

3.2 Kernel Forward Feature Selection (KFFS)

Backed by the orthogonal basis set constructed by KOBL, we can select discriminative features in the kernel space. During past few years, many methods [33, 15, 32, 36] have been proposed for feature selection in the original space. In our work, we extend Forward Feature Selection (FFS) to select features in the kernel space due to its smaller computing load. In this part, we show how to adjust FFS, a method that was proposed to select features in the original space [36], to directly select features in the kernel space with the assistance of the orthogonal basis set learned by KOBL.

The key issue of FFS is to train weak classifiers and store the their classification results. Here, each weak classifier is trained

1: **Input:** Data matrix $A^{(0)} = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), ..., \phi(\boldsymbol{x}_N)]$.
2: Initialize the basis $\mathbf{M} = \emptyset$ and the coefficient vectors $\alpha_j^{(0)} = \boldsymbol{e_j}, j = 1 : N$.
3: Initialize the basis dimensionality $k = Dim(\mathbf{M}) = 0$.
4: Column pivoting: find the column j by (5).
5: Interchange column j and column $k+1$ of matrix $A^{(k)}$ if $j \neq k+1$.
6: Implicitly compute the potential base vector $\mathbf{w_{k+1}}$ by (12).
7: **if** Threshold condition (20) satisfies **then**
8: Accept $\mathbf{w_{k+1}}$ as a base vector and add it into $\mathbf{M}$.
9: Implicitly update matrix $A^{(k)}$ using (13).
10: Update basis dimensionality $k \leftarrow k+1$.
11: Goto step 4 to continue the learning process.
12: **else**
13: Stop the algorithm.
14: **end if**
15: **return** Basis $\mathbf{M}$ and $Dim(\mathbf{M})$.

Algorithm 1: Kernel Orthogonal Basis Learner

based on one feature. By the aid of the learned orthogonal basis set $\{\mathbf{w_i}\}_{i=1}^d$ of the kernel space, we obtain the i^{th} feature of the mapped data $\phi(\boldsymbol{x_j})$ when training the weak classifiers:

$$\phi_i(\boldsymbol{x_j}) =< \mathbf{w_i}, \phi(\boldsymbol{x_j}) >= \sum_{m=1}^{N} \hat{\alpha}_{i,m} \mathbf{K}(\boldsymbol{x_m}, \boldsymbol{x_j}) \tag{22}$$

In Adaboost and its variants, many supervised methods can be used to train the weak classifiers. The most commonly used method is Nearest Neighbor Classifier, which has a time complexity of $O(N^2)$. Thus its kernel version (Kernel Nearest Neighbor Classifier) is $O(N^3)$. In our work, Kernel Least Mean Square algorithm is employed because it only costs $O(N^2 d)$ computing time.

The detailed KFFS is given in Algorithm 2.

1: **Input:** Data set $\{\phi(\boldsymbol{x}_i), y_i\}_{i=1}^N$ and basis set $\{\mathbf{w_k}\}_{k=1}^d$.
2: **for** $i = 1$ to d **do**
3: Train a weak classifier only using the i^{th} feature of the mapped data by the aid of the orthogonal basis.
4: Store the classification result under all the training data $\phi(\boldsymbol{x}_j)$ into the i^{th} row of table V, $j = 1 : N$.
5: **end for**
6: **for** $t = 1$ to d **do**
7: Normalize the weights $c_{t,i} \leftarrow \frac{c_{t,i}}{\sum_j c_{t,j}}$.
8: **for** $i = 1$ to d **do**
9: $v' = v + V(i, :)$.
10: ϵ_i = the error rate of ensemble classifier under feature sets v'.
11: **end for**
12: $k = \text{argmin}_{1 \leq i \leq d} \epsilon_i$.
13: $v = v + V(k, :)$.
14: Update the weights: $c_{t+1,i} = c_{t,i}(\frac{\epsilon_k}{1-\epsilon_k})^{1-b_i}$, where $b_i = 0$ if samples $\phi(\boldsymbol{x}_i)$ is classified correctly, $b_i = 1$ otherwise.
15: Computed the weight of the selected feature by equation (23).
16: **end for**
17: **return** The weight of each feature $\{\gamma_i\}_{i=1}^d$.

Algorithm 2: Kernel Forward Feature Selection

The Kernel Forward Feature Selection (KFFS) operates as follows: first, weak classifiers are trained (step 2- step 5 of Algorithm 2). Then we select features that make the ensemble classifier have smallest error on the training set (step 6 - step 13). After the features in an ensemble are selected, we update the weights of features according to the smallest error rate ϵ_k (step 14- step 15):

$$\gamma_k = log \frac{1 - \epsilon_k}{\epsilon_k} \tag{23}$$

The discriminative ability of each feature is reflected by the computed weight. The time complexity of KFFS is $O(N^2 d)$.

3.3 Summary of Algorithm

In this section, we give the detailed TAKES in Algorithm 3. In the first stage, we learn an orthogonal basis for the kernel space by KOBL. Note that the intrinsic dimensionality of the feature space can be estimated by KOBL in the first stage. Then in the second stage, with the aid the basis learned in the first stage, the discriminative features are weighted directly in the kernel space.

1: **Input:** Data $\{\boldsymbol{x}_i\}_{i=1}^N$ and label $\{y_i\}_{i=1}^N$.
2: Construct an orthogonal basis set $\{\mathbf{w_k}\}_{k=1}^d$ of the kernel space by KOBL.
3: Calculate weight $\{\gamma_k\}_{k=1}^d$ by KFFS.
4: Project the data into learned subspace with the aid of the learned weights.

Algorithm 3: TAKES

As discussed above, the time complexity of constructing basis set is $O(N^2 d)$, and KFFS is $O(N^2 d)$. Thus, the total time complexity of the proposed TAKES is $O(N^2 d)$. Therein, N is the number of training data, and d is the dimensionality of the original data. To compare with KOBL, we analyze the time complexity of some other kernel based feature selection such as *Feature Selection via KPCA* (FSKPCA) and *Feature Selection via KGS*(FSKGS)[6], and some dimensionality reduction methods such as *Kernel Discriminant Analysis*(KDA) [22] and *Generalized Discriminant Analysis*(GDA)[1]. The results are listed in Table 1.

Table 1 indicates that the proposed TAKES outperforms the only existing kernel feature selection methods FSKPCA and FSKGS [6] as far as we know. Compared with some kernel based dimensionality reduction methods, such as KDA and GDA, TAKES is encouraging to the aspects of time complexity.

4. EXPERIMENTS

To validate the effectiveness and efficiency of the proposed methods, we conduct experiments under some widely used real-world databases from [2] and [11]: Australian, Diabetes, Ionosphere, Heart, Sonar, and Thyroid. Some typical kernel methods for reducing dimensionality are compared with our TAKES. As discussed above, as far as we know, the only existing work that selects features in the kernel space is [6]. Thus, we compare our work with FSKGS and FSKPCA [6]. Kernel Discriminant Analysis (KDA), a typical kernel dimensionality reduction method, is also used to compare with our work.

4.1 Reconstruction Cost

In our work, we estimate the intrinsic dimensionality for the kernel space and construct an orthogonal basis by the proposed KOBL to conduct feature selection in the kernel space. In this part, we employ the reconstruction cost $\mathcal{E}(\mathbf{M})$ [24] to testify whether the learned basis can accurately preserve the sketch of the mapped data in the kernel space.

Table 1: Time complexity of TAKES, FSKPCA, FSKGS, KDA, and GDA

Algorithm	TAKES	FSKPCA	FSKGS	KDA	GDA
Time Complexity	$O(N^2d)$	$O(N^3+N^2d)$	$O(N^3)$	$O(N^3)$	$O(N^3+N^2d)$

Table 2: Reconstruction Cost $\mathcal{E}(\mathbf{M})$ of Each Method

Dataset	KOBL		KGS		KPCA		KDA	
	$\mathcal{E}(\mathbf{M})$	Dim	$\mathcal{E}(\mathbf{M})$	Dim	$\mathcal{E}(\mathbf{M})$	Dim	$\mathcal{E}(\mathbf{M})$	Dim
Australian	**0.044**	36	0.063	345	0.412	345	0.523	36
Diabetes	**0.062**	58	0.082	468	0.392	468	0.496	58
Heart	**0.145**	44	0.158	170	0.384	170	0.503	44
Ionosphere	0.111	69	**0.098**	281	0.383	281	0.915	69
Sonar	**0.047**	12	0.067	104	0.519	104	0.949	12
Thyroid	**0.043**	20	0.083	140	0.407	140	0.700	20

To compare with the quality of basis set constructed by our work, we use some typical kernel subspace learners such as KGS, KPCA, and KDA to compare with KOBL under some widely used databases [2, 11]. Without loss of generality, we use RBF kernel for KOBL, KGS, KPCA, and KDA respectively. It deserves noting KOBL automatically estimates the intrinsic dimensionality of the kernel space, which is listed in Table 2. But for KGS and KPCA, the dimensionality of the kernel space needs users to predetermine. For KGS and KPCA, we take the policy adopted by [6]: the target dimensionality is predetermined as the size of training set. For KDA, the target dimensionality needs users to set up as well. For the purpose of fair comparison, in KDA we set up the dimensionality for the kernel space the same as it learned by KOBL.

The results of $\mathcal{E}(\mathbf{M})$ listed in Table 2 indicate that under most conditions, KOBL achieves the smallest reconstruction cost compared with other methods. It is because for KOBL, the proper learned intrinsic dimensionality of the kernel space and the orthogonality of constructed basis guarantee that the sketch of the kernel space is accurately preserved. For KGS, the improper predetermined dimensionality for the kernel space undermines the orthogonality of basis set. KPCA achieves higher reconstruction cost than KOBL under all datasets. It demonstrates again that setting up the number of mapped data as the kernel space dimensionality adopted by [6] is unreasonable because the too large predetermined dimensionality of the kernel space renders it difficult to obtain orthogonal basis, and preserve the sketch of the data accurately. The basis set constructed by KDA is not necessary orthogonal, thus it is no wonder that its reconstruction cost is poor.

From the experiments, we can conclude that, compared with other methods, KOBL constructs the basis set with the smallest reconstruction cost. It means the learned basis set by our work can preserve the sketch of data in the kernel space accurately.

4.2 Classification Performance

In this section, we conduct experiments to validate the classification power of the proposed method. Similar to section 4.1, the six datasets, i.e., Australian, Diabetes, Ionosphere, Heart, Sonar, and Thyroid, are also used in this experiment.

4.2.1 Case Study: the Effectiveness of the Constructed Basis

In our algorithm, we design the KOBL to learn the intrinsic dimensionality and the orthogonal basis of the kernel space (i.e., the first stage), and then adopt KFFS to directly select features in the kernel space (i.e., the second stage). In this subsection, to further validate the effectiveness of the basis constructed by KOBL, we compare the classification accuracy of TAKES with another case that the feature selection stage is the same but the basis learners are different. Namely, in this case, we use KGS to learn the basis (in the first stage), and then apply KFFS to select features in the kernel space (in the second stage).

We execute the above two methods to obtain the selected features. Then we simply use $1-$Kernel Nearest Neighbor Classifier (KNNC) to classify data in test sets under the selected features. We depict the classification results of each method under different databases in Fig.2. In the figures, the abscissa stands for the number of selected features and the vertical ordinate stands for the corresponding classification accuracy. It deserves noting that for the proposed TAKES, the dimensionality of the kernel space is automatically learned. In the stage of selection, we employ KFFS to weight all features according to their respective discriminative abilities. Therefore, the number of selected features for TAKES is data-dependent, and we only need to portray a point instead of a curve line to reflect the classification performance of TAKES.

Fig.2 dedicates that the classification accuracy of TAKES is better than the KGS+KFFS case under most databases. The feature selection method (in the second stage) used in the case study is the same, thus we can conclude the basis learned by KOBL is the best. It is because different from KGS, KOBL can estimate the intrinsic dimensionality for the kernel space, which ensures the sketch of the kernel space is accurately preserved by the learned orthogonal basis and the classification accuracy is good. As for the KGS+KFFS case, the heuristically predetermined target dimensionality undermines the orthogonality of the learned basis set, and leads to poorer classification performance.

4.2.2 Compared with Typical Methods

In this part, we compare the classification accurcy of TAKES with the only existing feature selection works in the kernel space [6] as far as we know: FSKGS and FSKPCA. KDA is also used to compare with TAKES.

We execute each algorithm to obtain the selected features. Then we use $1-$KNNC for classification under the selected features. Meanwhile, we compute the classification accuracy that is obtained merely by KNNC without dimensionality reduction or feature selection scheme, which reflects the classification performance of all features in the kernel space. We depict the classification results in Fig.3.

Fig. 3 indicates that TAKES can obtain sound classification performance compared with other typical kernel methods for reducing dimensionality. Compared with KNNC, TAKES obtains better classification results under all datasets. It indicates that after feature selection, the selected features have better discriminative power than all features in the kernel space.

The classification accuracy of the proposed TAKES outperforms FSKGS and FSKPCA under almost all the datasets. It is because that TAKES can properly estimate the intrinsic dimensionality of the kernel space and obtain orthogonal basis. Another point worth mentioning is that, compared with the classification accuracy of

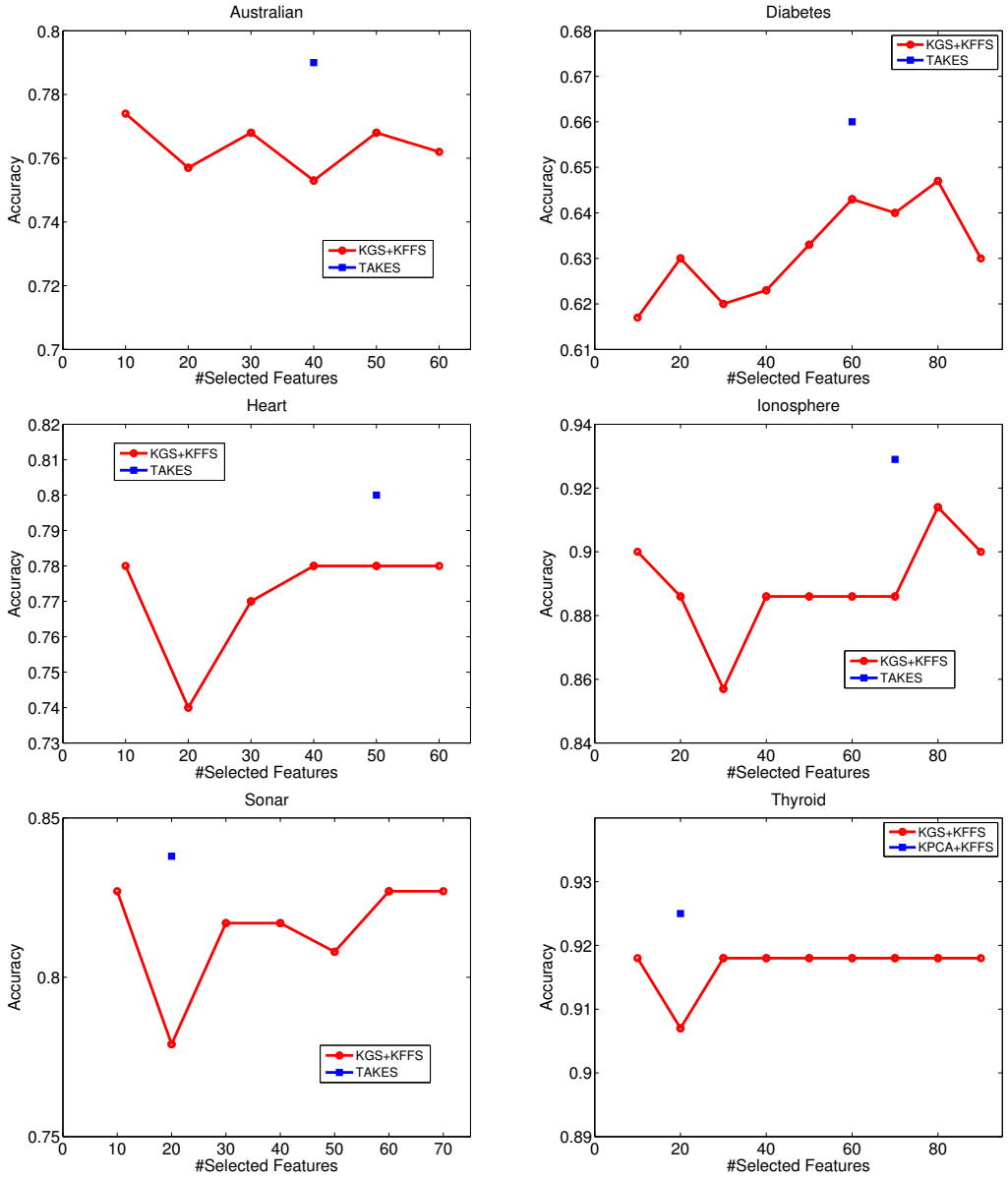

Figure 2: The classification accuracy of TAKES and KGS+KFFS under different number of selected features: the abscissa coordinate stands for the selected feature number, while the vertical coordinate stands for the corresponding classification accuracy. Noting that TAKES can learn the intrinsic dimensionality of the kernel space. Therefore, we only portray a point instead of a curve line to show the classification accuracy of TAKES.

FSKGS, KGS+KFFS (shown in the last subsection) is a little better under most datasets. (KGS+KFFS is better under the Diabetes, Heart, Ionosphere, Sonar, and Thyroid.) It demonstrates that the Kernel Forward Feature Selection (KFFS) adopted by the proposed TAKES is effective to detect discriminant features in the kernel space.

Compared to KDA, which is a kernel dimensionality reduction method that aims at obtaining a transformation matrix to convert original data into data in feature space, TAKES works better under all conditions. Although KDA improves the classification accuracy by maximizing the between-class scatter and minimizing the within-class scatter, TAKES, which directly selects discriminative features from the kernel space, is proven to be more effective.

Then considering the fact that the time complexity of TAKES is better than many typical kernel methods for reducing dimensionality, our method is promising.

5. CONCLUSION

In this paper, we propose a framework for feature selection in the kernel space. In the first stage, we design a subspace learning method KOBL to estimate the intrinsic dimensionality and construct an orthogonal basis for the kernel space. The designed subspace learning algorithm automatically learns the intrinsic dimen-

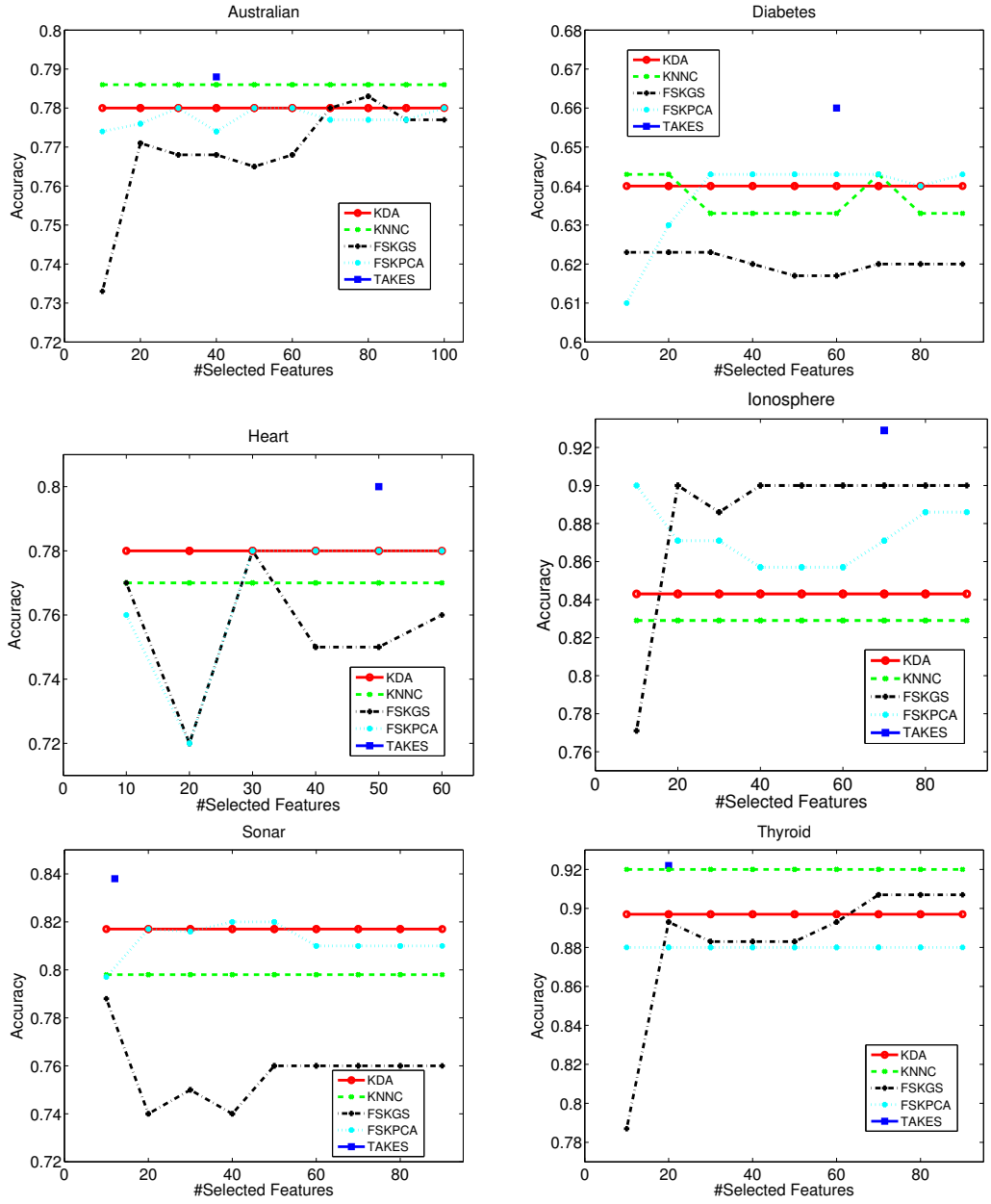

Figure 3: The classification accuracy of TAKES, FSKGS, FSKPCA, KDA, and KNNC: the abscissa coordinate stands for the number of selected features, while the vertical coordinate stands for the corresponding classification accuracy. Noting that TAKES can learn the intrinsic dimensionality of the kernel space. Therefore, different from other algorithms, we only portray a point instead of a curve line to show the classification accuracy of TAKES.

sionality of the kernel space guaranteeing that the sketch of the kernel space is accurately preserved. In the second stage, with the aid of the learned basis set, we select features directly in the kernel space. The whole procedure enjoys a smaller time complexity compared with other typical kernel methods.

In the experiments, we compare the proposed TAKES framework with FSKGS and FSKPCA (the only existing feature selection methods as far as we know), and KDA (a famous kernel dimensionality reduction method) under several typical datasets. The results dictate that the reconstruction cost and the classification performance of TAKES are the best.

6. ACKNOWLEDGEMENT

We would like to thank Shaohan Hu from UIUC and the anonymous reviewers for their invaluable inputs. This work was supported in part by the Fund of the National Natural Science Foundation of China (Grant #60975047, 60723003, 60721002), 973 Program (2010CB327903), and Jiangsu NSF grant (#BK2009080).

7. REFERENCES

[1] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural Computation*, 12(10):2385–2404, 2000.

[2] C. L. Blake and C. J. Merz. UCI repository of machine learning databases. *Irvine, CA: University of California Department of Information*, 1996.

[3] M. L. Braun and J. M. Buhmann and K.-L. Muller. Denoising and Dimension Reduction in Feature Space. In *the Advances in Neural Information Processing Systems*, pages 185–192, 2006.

[4] M. L. Braun and J. M. Buhmann and K.-L. Muller. On Relevant Dimensions in Kernel Feature Spaces. *Journal of Machine Learning Research*, 9:1875–1908, 2008.

[5] D. Cai and X. He. Orthogonal locality preserving indexing. In *ACM SIGIR*, pages 3–10, 2005.

[6] B. Cao, D. Shen, J. Sun, Q. Yang, and Z. Chen. Feature selection in a kernel space. In *ICML*, pages 121–128, 2007.

[7] M. D. Choi, "Tricks or treats with the hilbert matrix," *American Mathematical Monthly*, 90:301–312, 1983.

[8] I. Dagher and R. Nachar. Face recognition using IPCA-ICA algorithm. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(6):996–1000, 2006.

[9] P. A. Devijver and J. Kittler. *Pattern Recognition: A Statistical Approach*. Prince-Hall, 1982.

[10] W. Dong, M. Charikar, and K. Li. Asymmetric distance estimation with sketches for similarity search in high-dimensional spaces. In *ACM SIGIR*, pages 479–490, 2008.

[11] M. Duarte and Y. H. Hu. Vehicle classification in distributed sensor networks. *J. Parallel and Distributed Computing*, 64(7):826–838, 2004.

[12] X. Geng and T.-Y. Liu and T. Qin and H. Li. Feature selection for ranking. In *ACM SIGIR*, pages 407–414, 2007.

[13] G. H. Golub and C. F. V. Loan. *Matrix Computations, 3rd Edition*. Johns Hopkins University Press, 1996.

[14] H. Grabner and H. Bischof. On-line boosting and vision. In *IEEE CVPR*, pages 260–267, 2006.

[15] K. Kira and L. A. Rendell. A practical approach to feature selection. In *ICML*, pages 249–256, 1992.

[16] A. Kolcz, X. Sun, and J. Kalita. Efficient handling of high-dimensional feature spaces by randomized classifier ensembles. In *ACM SIGKDD*, pages 307–313, 2002.

[17] H. Li, T. Jiang, and K. Zhang. Efficient and robust feature extraction by maximum margin criterion. In *the Advances in Neural Information Processing Systems*, pages 97–104, 2003.

[18] Z. Liang and T. Zhao. Feature selection for linear support vector machines. In *ICPR*, pages 606–609, 2006.

[19] Y. Ma, S. Lao, E. Takikawa, and M. Kawade. Discriminant analysis in correlation similarity measure space. In *ICML*, pages 577–584, 2007.

[20] S. Martin, M. Kirby, and R. Miranda. Kernel/feature selection for support vector machines applied to materials design. In *the Symposium on Artificial Intelligence in Real Time Control*, pages 29–34, 2004.

[21] A. M. Martinez and A. C. Kak. PCA versus LDA. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(2):228–233, 2001.

[22] S. Mika, G. Ratsch, and K. R. Muller. A mathematical programming approach to the kernel fisher algorithm. In *the Advances in Neural Information Processing Systems*, pages 591–597, 2001.

[23] W. Ping, Y. Xu, K. Ren, C.-H. Chi, and F. Shen. Non-I.I.D. Multi-Instance Dimensionality Reduction by Learning a Maximum Bag Margin Subspace. In *AAAI*, pages 551–556, 2010.

[24] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.

[25] Y. Saeys and I. Inza and P. Larranaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[26] B. Scholkopf, A. Smola, and K. R. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[27] A. J. Smola and O. L. Mangasarian and B. Scholkopf. parse Kernel Feature Analysis. *Technical Report*, 1999.

[28] Y. Song, D. Zhou, J. Huang, I. G. Councill, H. Zha, and C. L. Giles. Boosting the feature space: Text classification for unstructured data on the web. In *IEEE ICDM*, pages 1064–1069, 2006.

[29] Y. Sun. Iterative relief for feature weighting: Algorithms, theories, and applications. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(6):1035–1051, 2007.

[30] X. Tao, J. Ye, Q. Li, R. Janardan, and V. Cherkassky. Efficient kernel discriminant analysis via QR decomposition. In *the Advances in Neural Information Processing Systems*, pages 1529–1536, 2004.

[31] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290:2319–2323, 2000.

[32] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE CVPR*, pages 511–518, 2001.

[33] G. Wang, F. H. Lochovsky, and Q. Yang. Feature selection with conditional mutual information maximin in text categorization. In *ACM CIKM*, pages 342–349, 2004.

[34] L. Wang, M. Sugiyama, Z. H. Zhou, and J. Feng. On the margin explanation of boosting algorithms. In *COLT*, pages 479–490, 2008.

[35] J. Weston, A. Elisseeff, and B. Scholkopf. Use of zero-norm with linear models and kernel methods. *J. Machine Learning Research*, 3:1439–1461, 2003.

[36] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg. Fast asymmetric learning for cascade face detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(3):369–382, 2008.

[37] Y. Xu. Orthogonal component analysis for dimensionality reduction. *Master Thesis, Nanjing University*, 2010.

[38] Y. Xu, S. Furao, J. Zhao, and O. Hasegawa. To obtain orthogonal feature extraction using training data selection. In *ACM CIKM*, pages 1819–1822, 2009.

[39] J. Yan, B. Zhang, S. Yan, Q. Yang, H. Li, and Z. Chen. IMMC: incremental maximum margin criterion. In *ACM SIGKDD*, pages 725–730, 2004.

[40] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *ICML*, pages 412–420, 1997.