A Subspace Kernel for Nonlinear Feature Extraction

Mingrui Wu, Jason Farquhar
Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany
{firstname.lastname}@tuebingen.mpg.de

Abstract

Kernel based nonlinear Feature Extraction (KFE) or dimensionality reduction is a widely used pre-processing step in pattern classification and data mining tasks. Given a positive definite kernel function, it is well known that the input data are implicitly mapped to a feature space with usually very high dimensionality. The goal of KFE is to find a low dimensional subspace of this feature space, which retains most of the information needed for classification or data analysis. In this paper, we propose a *subspace kernel* based on which the feature extraction problem is transformed to a kernel parameter learning problem. The key observation is that when projecting data into a low dimensional subspace of the feature space, the parameters that are used for describing this subspace can be regarded as the parameters of the kernel function between the projected data. Therefore current kernel parameter learning methods can be adapted to optimize this parameterized kernel function. Experimental results are provided to validate the effectiveness of the proposed approach.

1 Introduction

Feature extraction or dimensionality reduction is a widely used pre-processing step for classification and data mining tasks, since extracting proper features can reduce the effect of noise and remove redundant information in the data that is irrelevant to the classification or data analysis tasks.

Suppose that we are given a set of n data points, $\{\mathbf{x}_i\}_{i=1}^n$, where $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$ is the input data, $\mathcal{X}$ is the input space. Traditional feature extraction approaches, such as the *Principle Component Analysis* (PCA) and *Linear Discriminant Analysis* (LDA) are linear methods and they project the input data $\mathbf{x}_i$ into a low dimensional subspace of the input space $\mathcal{X}$.

Recently, constructing nonlinear algorithms based on the kernel methods [Schölkopf and Smola, 2002] have proved successful. For a given positive definite kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the input data $\mathbf{x}_i$, $1 \leq i \leq n$ are implicitly mapped to a feature space $\mathcal{F}$ with usually very high dimensionality. Let $\phi(\cdot)$ denote the map from $\mathcal{X}$ to $\mathcal{F}$, then

$$K(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle, \quad 1 \leq i, j \leq n$$

A kernel based algorithm essentially applies linear methods in $\mathcal{F}$ for the mapped data $\{(\phi(\mathbf{x}_i)\}_{i=1}^n$. For example, in the *Kernel Principal Component Analysis* (KPCA) algorithm [Schölkopf and Smola, 2002], PCA is used to extract a representative subspace of $\mathcal{F}$. Compared with the traditional linear approaches, kernel methods are more powerful since they can explore nonlinear structures of the data, and more flexible as we can recover the linear algorithms by simply using the linear kernel in the kernel based methods.

Usually the dimensionality of $\mathcal{F}$ is very high or even infinite, which is helpful for separating different classes of data. However, such a high dimensional space $\mathcal{F}$ may contain some redundancy that is irrelevant or even noisy for the given classification or data mining tasks. Hence, as is the case for feature extraction in the input space, it may be also helpful for classification or data mining tasks to find a lower dimensional subspace $\mathcal{S}$ of $\mathcal{F}$.

Many *Kernel based Feature Extraction* (KFE) approaches have been proposed to find a lower dimensional subspace $\mathcal{S}$ of the feature space $\mathcal{F}$. For example, KPCA [Schölkopf and Smola, 2002] is widely used for this task. As mentioned above, it essentially performs linear PCA in the feature space $\mathcal{F}$. The goal is to find directions along which the data variance is the largest.

In this paper, we discuss feature extraction methods with the focus on improving the classification accuracy. In the c-class classification problem, each data point $\mathbf{x}_i$ is associated with a label $\mathbf{y}_i \in \mathbb{R}^c$, where $\mathbf{y}_i = [y_{i1}, \ldots, y_{ic}]^\top$, and $y_{ik} = 1$ $(1 \leq k \leq c)$ if $\mathbf{x}_i$ belongs to class k, and 0 otherwise.[1] It can be seen that KPCA may be not effective for classification problems since it is an unsupervised feature extraction method, which ignores the labels of the given data.

Hence several supervised KFE algorithms have been proposed, which make use of both the input data and the

[1]Other strategies for constructing the label $\mathbf{y}_i$ $(1 \leq i \leq n)$ are also possible.

corresponding labels. Like KPCA, they also perform linear feature extraction or linear dimensionality reduction in the feature space $\mathcal{F}$.

The *Kernel Fisher Discriminant Analysis* (KFDA) [Mika *et al.*, 2001] aims to find a data projection by minimizing the within-class variance and maximizing the between-class variance simultaneously, thus achieving good discrimination between different classes. An efficient variant of KFDA based on QR decomposition, called AKDA/QR, is proposed in [Xiong *et al.*, 2005]. A distinct property of AKDA/QR is that it scales as $O(ndc)$. And in AKDA/QR, the number of features extracted is fixed to the number of classes.

The Partial Least Squares (PLS) algorithm [Wold, 1975] has been widely applied in the domain of chemometrics. Unlike the PCA algorithm, which extracts features only based on the variance of the input data, the PLS algorithm uses the covariance between the inputs and the labels to guide the extraction of features. The Kernel PLS (KPLS) algorithm is proposed in [Rosipal and Trejo, 2001].

The *Orthogonal Centroid* (OC) [Park and Park, 2004] algorithm is a linear dimensionality reduction method that preserves the cluster structure in the data. In this algorithm, the given data are firstly clustered, and then projected into a space spanned by the centroids of these clusters. An orthogonal basis of this subspace is computed by applying QR decomposition to the matrix whose columns consist of the cluster centroids. In [Kim *et al.*, 2005], this method is applied for dimensionality reduction in text classification tasks and exhibits good results. Its kernel based nonlinear extension, i.e. the *Kernel Orthogonal Centroid* (KOC) algorithm is also presented in [Park and Park, 2004]. To incorporate the label information, the KOC (and OC) algorithm treats input data in the same class as one single cluster, therefore the number of extracted features equals the number of classes. However this method can be easily extended by allowing more clusters in each class.

In this paper, we propose a subspace kernel, based on which the nonlinear feature extraction problem can be transformed into a kernel parameter learning problem.

The rest of this paper is organized as follows. In section 2, we propose the basic idea of our approach and formulate the subspace kernel. Some connections to the related methods are described in section 3. In section 4, we present one possible way to optimize the proposed subspace kernel. Experimental results are provided in section 5 and we conclude the paper in the last section.

2 Nonlinear Feature Extraction via Kernel Parameter Learning

2.1 Basic Idea

As mentioned before, a given positive definite kernel K implicitly introduces a mapping of the given data $\phi(\mathbf{x}_i)$, $1 \leq i \leq n$, to a usually high dimensional feature space $\mathcal{F}$. *When projecting $\phi(\mathbf{x}_i)$ ($1 \leq i \leq n$) into a subspace $\mathcal{S}$ of $\mathcal{F}$, the kernel function has to be modified correspondingly since the feature space has changed from $\mathcal{F}$ to $\mathcal{S}$.* For convenience, we call this modified kernel function the *subspace kernel.* As will be shown later, *the parameters that are used for describing $\mathcal{S}$ are also the parameters of the corresponding subspace kernel.* Therefore current kernel parameter learning methods can be adapted to optimize this kernel function. This way we can find a discriminating subspace $\mathcal{S}$ where different classes of data are well separated. In the following, we will explain the above idea in detail by formulating the aforementioned subspace kernel.

2.2 The Subspace Kernel

Suppose $\mathcal{S}$ is an n_f dimensional subspace of $\mathcal{F}$ and $\mathbf{O} = [\mathbf{o}_1, \ldots, \mathbf{o}_{n_f}]$ is a matrix whose columns constitute an orthogonal basis of $\mathcal{S}$. Let $\mathcal{T}$ denote the subspace spanned by the mapped data $\phi(\mathbf{x}_i)$ ($1 \leq i \leq n$) in $\mathcal{F}$, then each $\mathbf{o}_i$ can be uniquely decomposed into two parts, one is contained in $\mathcal{T}$ and the other one is in the orthogonal complement of $\mathcal{T}$,

$$\mathbf{o}_k = \mathbf{o}_k^{\parallel} + \mathbf{o}_k^{\perp}, \quad 1 \leq k \leq n_f$$

where $\mathbf{o}_k^{\parallel} \in \mathcal{T}$ and $\langle \mathbf{o}_k^{\perp}, \phi(\mathbf{x}_i) \rangle = 0$ for $1 \leq i \leq n$. Therefore for any $\phi(\mathbf{x}_i)$, its projection into $\mathcal{S}$ can be computed as

$$\mathbf{O}^\top \phi(\mathbf{x}_i) = (\mathbf{O}^{\parallel})^\top \phi(\mathbf{x}_i) \tag{1}$$

where $\mathbf{O}^{\parallel} = [\mathbf{o}_1^{\parallel}, \ldots, \mathbf{o}_{n_f}^{\parallel}]$.[2]

Equation (1) indicates that to compute the projection of $\phi(\mathbf{x}_i)$ in $\mathcal{S}$, it is enough to only consider the case where $\mathcal{S}$ is a subspace of $\mathcal{T}$, which implies that any vector in $\mathcal{S}$ can be expressed as a linear combination of $\phi(\mathbf{x}_i)$, $1 \leq i \leq n$. Therefore, for any n_f vectors $\mathbf{z}_1, \ldots, \mathbf{z}_{n_f} \in \mathcal{S}$, let $\mathbf{Z}$ denote $[\mathbf{z}_1, \ldots, \mathbf{z}_{n_f}]$, and $\mathbf{X}$ denote $[\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)]$, then $\mathbf{Z}$ can be written as

$$\mathbf{Z} = \mathbf{X}\mathbf{W} \tag{2}$$

where $\mathbf{W} = [w_{ik}] \in \mathbb{R}^{n \times n_f}$ is a matrix of combination coefficients.

Moreover, if $\mathbf{z}_1, \ldots, \mathbf{z}_{n_f}$ are linearly independent, then the n_f dimensional subspace $\mathcal{S}$ can be spanned by these n_f vectors. Thus the elements of $\mathbf{W}$ introduce a subspace $\mathcal{S}$ of $\mathcal{F}$, for which we have the following lemma.

Lemma 1. *When projecting the data $\phi(\mathbf{x}_i)$ into $\mathcal{S}$, the kernel matrix of the projected data in $\mathcal{S}$ can be computed as,*[3]

$$\begin{aligned} \mathbf{K}^w &= (\mathbf{X}^\top \mathbf{Z})(\mathbf{Z}^\top \mathbf{Z})^{-1}(\mathbf{X}^\top \mathbf{Z})^\top && (3) \\ &= (\mathbf{K}\mathbf{W})(\mathbf{W}^\top \mathbf{K}\mathbf{W})^{-1}(\mathbf{K}\mathbf{W})^\top && (4) \end{aligned}$$

where $\mathbf{K} = [k_{ij}] \in \mathbb{R}^{n \times n}$ is the kernel matrix of the input data, i.e. $k_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$.

[2]More precisely, the result of equation (1) is the coordinate of the projection of $\phi(\mathbf{x}_i)$ in $\mathcal{S}$. As is widely done in the literature of feature extraction and dimensionality reduction, this coordinate will be used as the extracted features for classification.

[3]Here the "kernel matrix of the projected data" refers to the matrix whose elements equal the inner product of the projected data in $\mathcal{S}$.

Proof. For any $\phi(\mathbf{x}_i)$, $1 \leq i \leq n$, in order to calculate its projection into the subspace $\mathcal{S}$, spanned by the columns of $\mathbf{Z} = \mathbf{X}\mathbf{W}$, we need an orthogonal basis $\mathbf{U}$ of $\mathcal{S}$. We build $\mathbf{U}$ as follows:

$$\mathbf{U} = \mathbf{Z}\mathbf{T} \tag{5}$$

In the above equation, $\mathbf{T}$ is computed as follows: Assume $\mathbf{K}_z = \mathbf{Z}^\top\mathbf{Z}$ then

$$\mathbf{T} = \mathbf{V}\mathbf{\Lambda}^{-\frac{1}{2}} \tag{6}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{n_f \times n_f}$ is a diagonal matrix of eigenvalues of matrix $\mathbf{K}_z$, and $\mathbf{V} \in \mathbb{R}^{n_f \times n_f}$ is a matrix whose columns are eigenvectors of $\mathbf{K}_z$. Equation (6) leads to

$$\mathbf{K}_z^{-1} = \mathbf{T}\mathbf{T}^\top \tag{7}$$

and

$$\mathbf{T}^\top\mathbf{K}_z\mathbf{T} = \mathbf{I} \tag{8}$$

where $\mathbf{I}$ is the unit matrix. The following equation follows from (5) and (8),

$$\mathbf{U}^\top\mathbf{U} = \mathbf{T}^\top\mathbf{K}_z\mathbf{T} = \mathbf{I}$$

So the columns of $\mathbf{U}$ form an orthogonal basis of the subspace $\mathcal{S}$.

Thus, for $\phi(\mathbf{x}_i) \in \mathcal{F}$, $1 \leq i \leq n$, their projections into the subspace $\mathcal{S}$ can be computed as

$$\mathbf{X}_w = \mathbf{U}^\top\mathbf{X} = \mathbf{T}^\top\mathbf{Z}^\top\mathbf{X} \tag{9}$$

where $\mathbf{X}_w$ is the matrix whose columns are the projections of $\phi(\mathbf{x}_i)$ in $\mathcal{S}$, $1 \leq i \leq n$.

Having obtained the projected data $\mathbf{X}_w$, we can now compute the inner product between points in the subspace as the following:

$$\begin{aligned}
\mathbf{K}^w &= \mathbf{X}_w^\top\mathbf{X}_w = \mathbf{X}^\top\mathbf{U}\mathbf{U}^\top\mathbf{X} & (10)\\
&= \mathbf{X}^\top\mathbf{Z}\mathbf{T}\mathbf{T}^\top\mathbf{Z}^\top\mathbf{X} \\
&= (\mathbf{X}^\top\mathbf{Z})\mathbf{K}_z^{-1}(\mathbf{X}^\top\mathbf{Z})^\top \\
&= (\mathbf{X}^\top\mathbf{Z})(\mathbf{Z}^\top\mathbf{Z})^{-1}(\mathbf{X}^\top\mathbf{Z})^\top & (11)\\
&= (\mathbf{X}^\top\mathbf{X}\mathbf{W})(\mathbf{W}^\top\mathbf{X}^\top\mathbf{X}\mathbf{W})^{-1}(\mathbf{X}^\top\mathbf{X}\mathbf{W})^\top \\
&= (\mathbf{K}\mathbf{W})(\mathbf{W}^\top\mathbf{K}\mathbf{W})^{-1}(\mathbf{K}\mathbf{W})^\top & (12)
\end{aligned}$$

where we used equation (5) in the second line, equation (7) in the third line and equation (2) in the fifth line. The equations (11) and (12) are identical to (3) and (4) respectively, therefore the lemma is proven. □

The proof also tells that for a given $\mathbf{W}$, the projection of the data into the subspace $\mathcal{S}$ introduced by $\mathbf{W}$ can be computed as equation (9).

Let $K_w(\cdot,\cdot)$ denote corresponding subspace kernel function. Then according to (3) and (4), for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, the subspace kernel $K_w(\cdot,\cdot)$ between them can be computed as

$$\begin{aligned}
K_w(\mathbf{x}, \mathbf{x}') &= \phi(\mathbf{x})^\top\mathbf{Z}(\mathbf{Z}^\top\mathbf{Z})^{-1}\mathbf{Z}^\top\phi(\mathbf{x}') & (13)\\
&= \psi(\mathbf{x})^\top\mathbf{W}(\mathbf{W}^\top\mathbf{K}\mathbf{W})^{-1}\mathbf{W}^\top\psi(\mathbf{x}') & (14)
\end{aligned}$$

where

$$\psi(\mathbf{x}) = [K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n)]^\top$$

is the empirical kernel map [Schölkopf and Smola, 2002].

Equation (14) illustrates that the elements of $\mathbf{W}$, by which the subspace $\mathcal{S}$ is described, also serve as the kernel parameters of $K_w(\cdot,\cdot)$. So in order to find a discriminating subspace $\mathcal{S}$ where different classes of data are well separated, we can turn to optimize the corresponding subspace kernel K_w.

3 Connections to Related Work

3.1 Feature Selection via Kernel Parameter Learning

In [Weston *et al.*, 2000; Chapelle *et al.*, 2002], kernel parameter learning approaches are adopted for feature *selection* problem. The kernel of the following form is considered

$$K_\theta(\mathbf{u}, \mathbf{v}) = K(\boldsymbol{\theta}.*\mathbf{u}, \boldsymbol{\theta}.*\mathbf{v}) \tag{15}$$

where $.*$ denotes the component-wise product between two vectors. Namely, for $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_d]^\top$ and $\mathbf{u} = [u_1, \ldots, u_d]^\top$, $\boldsymbol{\theta}.*\mathbf{u} = [\theta_1 u_1, \ldots, \theta_d u_d]^\top$. By optimizing the kernel parameter $\boldsymbol{\theta}$ with margin maximization or Radius-Margin bound [Chapelle *et al.*, 2002] minimization, and with a 1-norm or 0-norm penalizer on $\boldsymbol{\theta}$, feature selection can be done by by choosing the features corresponding to the large elements of the optimized $\boldsymbol{\theta}$.

Feature selection locates a discriminating subspace of the input space $\mathcal{X}$. Similarly as the above approaches, we also use kernel parameter learning algorithms to find a discriminating subspace. However, in this paper, we address the problem of feature *extraction* but not feature *selection*, and the subspace we want to find is contained in the feature space $\mathcal{F}$ but not the input space $\mathcal{X}$.

3.2 Sparse Kernel Learning Algorithms

The subspace kernel function given by (14) is in a general form. As described before, each column in the matrix $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_{n_f}]$ (c.f (2)) is a vector in the feature space $\mathcal{F}$. Now we show that this kernel relates to the work of [Wu *et al.*, 2005] in the special case where each column of $\mathbf{Z}$ has a pre-image [Schölkopf and Smola, 2002] in the input space $\mathcal{X}$. That is, for each $\mathbf{z}_i \in \mathcal{F}$, there exists a vector $\hat{\mathbf{z}}_i \in \mathcal{X}$, such that $\mathbf{z}_i = \phi(\hat{\mathbf{z}}_i)$. So now the subspace $\mathcal{S}$ can be spanned by $\phi(\hat{\mathbf{z}}_1), \ldots, \phi(\hat{\mathbf{z}}_{n_f})$.

For convenience, let $\hat{\mathbf{Z}} = [\phi(\hat{\mathbf{z}}_1), \ldots, \phi(\hat{\mathbf{z}}_{n_f})]$ (note that $\hat{\mathbf{Z}} = \mathbf{Z}$). Then in this case, according to (13), the subspace kernel function now becomes:

$$\begin{aligned}
K_w(\mathbf{x}, \mathbf{x}') &= \phi(\mathbf{x})^\top\hat{\mathbf{Z}}(\hat{\mathbf{Z}}^\top\hat{\mathbf{Z}})^{-1}\hat{\mathbf{Z}}^\top\phi(\mathbf{x}') \\
&= \psi_{\hat{z}}(\mathbf{x})\mathbf{K}_{\hat{z}}^{-1}\psi_{\hat{z}}(\mathbf{x}') & (16)
\end{aligned}$$

where $\psi_{\hat{z}}(\mathbf{x}) = \phi(\mathbf{x})^\top\hat{\mathbf{Z}} = [K(\mathbf{x}, \hat{\mathbf{z}}_1), \ldots, K(\mathbf{x}, \hat{\mathbf{z}}_{n_f})]^\top$, and $\mathbf{K}_{\hat{z}} = \hat{\mathbf{Z}}^\top\hat{\mathbf{Z}}$.

In [Wu *et al.*, 2005], an algorithm for building *Sparse Large Margin Classifiers* (SLMC) is proposed, which builds a sparse *Support Vector Machine* (SVM) [Vapnik, 1995] with n_f expansion vectors, where n_f is an given integer. In [Wu *et al.*, 2005], it is pointed out that building an SLMC is equivalent to building a standard

SVM with the kernel function computed as (16). And the SLMC algorithm essentially finds an n_f dimensional subspace of $\mathcal{F}$, which is spanned by $\phi(\hat{\mathbf{z}}_1), \ldots, \phi(\hat{\mathbf{z}}_{n_f})$, and where the different classes of data are linearly well separated.

In [Wu *et al.*, 2005], the kernel function (16) is obtained with the Lagrange method, which is different from the one adopted in the above. And the kernel function (16) is a special case of the subspace kernel (14). Therefore it can be seen that based on the general subspace kernel (14), useful special cases can be derived for some applications.

4 Optimizing K_w

We optimize K_w based on the *Kernel-Target Alignment* (KTA) [Cristianini *et al.*, 2002], which is a quantity to measure the degree of fitness of a kernel for a given learning task. In particular, we compute $\mathbf{W}$ by solving the following KTA maximization problem:

$$\max_{\mathbf{W} \in \mathbb{R}^{n \times n_f}} A(\mathbf{W}) = \frac{\langle \mathbf{K}^w, \mathbf{K}^y \rangle_F}{\sqrt{\langle \mathbf{K}^w, \mathbf{K}^w \rangle_F \langle \mathbf{K}^y, \mathbf{K}^y \rangle_F}} \tag{17}$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius product between two matrices that are of the same size, i.e. for any two equally sized matrices $\mathbf{M}$ and $\mathbf{N}$, $\langle \mathbf{M}, \mathbf{N} \rangle_F = \sum_{ij} \mathbf{M}_{ij} \mathbf{N}_{ij}$. In (17), $\mathbf{K}^y \in \mathbb{R}^{n \times n}$ is the gram matrix between the labels, defined by

$$\mathbf{K}^y = \mathbf{Y}^\top \mathbf{Y} \tag{18}$$

where $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n] \in \mathbb{R}^{c \times n}$, and $\mathbf{y}_i$ is the label of $\mathbf{x}_i$, $1 \le i \le n$.

The elements in $\mathbf{K}^y$ reflect the similarities between labels, as $\mathbf{K}^y_{ij}$ equals 1 if $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same class, and 0 otherwise. Therefore 'aligning' $\mathbf{K}^w$ with $\mathbf{K}^y$ will make the similarities between the data points in the same class higher than the similarities between the points in different classes. Thus by maximizing $A(\mathbf{W})$, we can find a subspace of $\mathcal{F}$, where points in the same class are closer to each other than those in different classes. Hence a good classification performance can be expected for the data projected into this subspace.

Note that the subspace kernel allows us to apply many kernel parameter learning algorithms to the feature extraction problem. Therefore apart from KTA, we can also choose other approaches to compute $\mathbf{W}$, such as the one based on the *Radius-Margin Bound* [Chapelle *et al.*, 2002]. For simplicity, we use KTA in this paper.

Gradient based algorithms can be used to maximize $A(\mathbf{W})$. In our implementation, we use the conjugate gradient algorithm to solve problem (17). To compute $A(\mathbf{W})$, we utilize the fact that $\mathbf{K}^w = \mathbf{X}_w^\top \mathbf{X}_w$ (see (10)) and $\mathbf{K}^y = \mathbf{Y}^\top \mathbf{Y}$ (see (18)). Thus, we can decompose $\mathbf{K}^w$ and $\mathbf{K}^y$ as follows

$$\mathbf{K}^w = \sum_{i=1}^{n_f} \hat{\mathbf{x}}_i \hat{\mathbf{x}}_i^\top \tag{19}$$

$$\mathbf{K}^y = \sum_{j=1}^{c} \hat{\mathbf{y}}_j \hat{\mathbf{y}}_j^\top \tag{20}$$

where $\hat{\mathbf{x}}_i \in \mathbb{R}^n$ $(1 \le i \le n_f)$ denotes the i-th column of $\mathbf{X}_w^\top$ and $\hat{\mathbf{y}}_j \in \mathbb{R}^n$ $(1 \le j \le c)$ denotes the j-th column of $\mathbf{Y}^\top$.

Based on the above two equations, we have

$$\langle \mathbf{K}^w, \mathbf{K}^y \rangle_F = \sum_{i=1}^{n_f} \sum_{j=1}^{c} (\hat{\mathbf{x}}_i^\top \hat{\mathbf{y}}_j)^2 \tag{21}$$

$$\langle \mathbf{K}^w, \mathbf{K}^w \rangle_F = \sum_{i=1}^{n_f} \sum_{j=1}^{n_f} (\hat{\mathbf{x}}_i^\top \hat{\mathbf{x}}_j)^2 \tag{22}$$

Equation (21) and (22) can be computed with time complexity $O(ncn_f)$ and $O(nn_f^2)$ respectively. When both n_f and c are small, they are more efficient than computing the Frobenius product directly, which requires time complexity of $O(n^2)$.

Similarly, to compute $\nabla A(\mathbf{W})$, we can use the following equations:

$$\langle \frac{\partial \mathbf{K}^w}{\partial w_{uv}}, \mathbf{K}^y \rangle_F = \sum_{i=1}^{c} \hat{\mathbf{y}}_i^\top (\frac{\partial \mathbf{K}^w}{\partial w_{uv}}) \hat{\mathbf{y}}_i \tag{23}$$

$$\langle \frac{\partial \mathbf{K}^w}{\partial w_{uv}}, \mathbf{K}^w \rangle_F = \sum_{j=1}^{n_f} \hat{\mathbf{x}}_j^\top (\frac{\partial \mathbf{K}^w}{\partial w_{uv}}) \hat{\mathbf{x}}_j \tag{24}$$

where w_{uv} $(1 \le u \le n, 1 \le v \le n_f)$ is the element of $\mathbf{W}$. Inspired by (23) and (24), we investigate how to compute $\boldsymbol{\alpha}^\top (\frac{\partial \mathbf{K}^w}{\partial w_{uv}}) \boldsymbol{\alpha}$, where $\boldsymbol{\alpha} \in \mathbb{R}^n$ is an arbitrary vector. Actually, by performing linear algebra straightforwardly, we have

$$\boldsymbol{\alpha}^\top \frac{\partial \mathbf{K}^w}{\partial w_{uv}} \boldsymbol{\alpha} = 2 t_u \beta_v \tag{25}$$

where β_v is the v-th element of a vector $\boldsymbol{\beta}$, computed as

$$\boldsymbol{\beta} = (\mathbf{W}^\top \mathbf{K} \mathbf{W})^{-1} (\mathbf{W}^\top \mathbf{K}) \boldsymbol{\alpha} \tag{26}$$

and in (25), t_u is the u-th element of a vector $\mathbf{t}$, defined as:

$$\mathbf{t} = \mathbf{K}\boldsymbol{\alpha} - \mathbf{K}\mathbf{W}\boldsymbol{\beta} \tag{27}$$

Note that for any given $\boldsymbol{\alpha}$, the vectors $\boldsymbol{\beta}$ and $\mathbf{t}$ need to be computed only once, according to (26) and (27) respectively, then $\boldsymbol{\alpha}^\top \frac{\partial \mathbf{K}^w}{\partial w_{uv}} \boldsymbol{\alpha}$ can be calculated as (25) for $1 \le u \le n$ and $1 \le v \le n_f$. Now we can apply (25) to (23) and (24), and $\nabla A(\mathbf{W})$ can be calculated.

5 Experimental Results

5.1 Experimental Settings

We empirically investigate the performance of the following KFE algorithms on classification tasks: KPLS, KOC, AKDA/QR and the proposed *Subspace Kernel based Feature Extraction* (SKFE) method. Following the same scheme in [Xiong *et al.*, 2005], the features extracted by each KFE algorithm are input to a *1-Nearest Neighbor* (1-NN) classifier, and the classification performance on the test data is used to evaluate the extracted features. As a reference, we also report the classification results of the 1-NN algorithm using the input data directly without KFE.

As mentioned before, in a c-class classification problem, the number of features n_f extracted by both AKDA/QR and KOC is fixed at c. To compare with these two algorithms, the value of n_f for SKFE is also set to c in the experiments, although the number of features extracted by SKFE can be varied. For KPLS, three different values of n_f are tried: $c/4$, $c/2$ and c. The best results are reported for KPLS.[4]

For our proposed SKFE algorithm, the function $A(\mathbf{W})$ in (17) is not convex, so the optimization result depends on the initial choice of $\mathbf{W}$. To get a good initial guess, we can use the subspaces found by other KFE algorithms for initialization. In the experiments, for efficiency we use the KOC algorithm to compute the initial $\mathbf{W}$.

5.2 Experiments on Microarray Gene Expression Data

In this subsection, we take seven microarray gene datasets to test various KFE methods: Brain Tumor1, Brain Tumor2, Leukemia1, Leukemia2, Prostate Tumor, DLBCL and 11_Tumors.[5] Descriptions of these datasets are presented in Table 1. As shown in Table 1, a typical characteristic of these datasets is that the number of data n is much smaller than the data dimensionality d.

Table 1: Datasets adopted in the experiments. The first seven are microarray gene datasets, while the last seven are text datasets. For each of them, the number of data n, the dimensionality d and the number of classes c are provided.

Dataset	type	n	d	c
B.Tumor1	GENE	90	5920	5
B.Tumor2	GENE	50	10367	4
Leukemia1	GENE	72	5327	3
Leukemia2	GENE	72	11225	3
P.Tumor	GENE	102	10509	2
DLBCL	GENE	77	5469	2
11_Tumors	GENE	174	12534	11
tr11	TEXT	414	6424	9
tr23	TEXT	204	5832	6
tr41	TEXT	878	7454	10
tr45	TEXT	690	8261	10
la1	TEXT	3204	31472	6
la2	TEXT	3075	31472	6
hitech	TEXT	2301	10080	6

A Gaussian kernel is used in the experiments:

$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \parallel \mathbf{x} - \mathbf{x}' \parallel^2) \qquad (28)$$

Five fold cross validation is conducted for parameter selection, and the best cross validation error rate is used to measure the performance of different algorithms. The experiment is repeated 20 times independently. And the results in Table 2 show the mean cross validation error and the standard deviation over these 20 runs.

[4]When $c = 2$, only two values of n_f are tried for KPLS: 1 and 2.

[5]They are available at http://www.gems-system.org.

From Table 2, we can observe that SKFE and KPLS compare favorably to the other KFE algorithms. In particular, SKFE improves the results of KOC algorithm in all cases, although KOC is used to initialize SKFE. It can also be seen that SKFE and KPLS are competitive with each other. They are are not significantly different (judged by t-test) on Leukemia1, Leukemia2, DLBCL and 11_Tumors, and KPLS is better than SKFE on Brain Tumor2, while SKFE outperforms KPLS on Brain Tumor1 and Prostate Tumor.

5.3 Experiments on Text Classification

In this subsection, we investigate different KFE methods on the text classification task. It has been observed that there usually exist cluster structures in the text data. The OC algorithm (or equivalently the KOC algorithm with the linear kernel), which can keep these structures, is used for dimensionality reduction in text classification tasks in [Kim *et al.*, 2005] and exhibits good results.

Seven text datasets from the TREC collections are adopted: tr11, tr23, tr41, tr45, la1, la2 and hitech. More information about these seven datasets are available at Table 1.

Similar to the microarray gene data, the data used in text classification tasks are also of very high dimensionality. Another characteristic of these seven datasets is that they are highly unbalanced, which means that the number of data contained in different classes are quite different. For example, in the tr11 dataset, there are 132 data points contained in the seventh class, while just 6 data points in the ninth class, only 4.6% of the former.

On each dataset, we randomly select half of the data from each class to form the training set and use the remaining data for test. As is done in the OC algorithm, the linear kernel is used in this set of experiments. Similarly as before, for each dataset, the experiment is repeated independently 20 times. The average test error and the standard deviation over these 20 runs are reported in Table 3.

Table 3 illustrates that SKFE outperforms other KFE methods on most datasets. Also it can be seen from both Table 2 and 3 that in most cases, all the KFE algorithms obtain better performances than the 1-NN algorithm with the raw data, whilst reducing the data dimensionality dramatically from d to n_f, where $n_f << d$. (c.f. section 5.1 for the choice of n_f.)

Although SKFE compares favorably to the other KFE methods in terms of the classification accuracy, its computational cost is higher than the others. For the problems reported in Table 1, on a 2.2 GHz Pentium-4 PC, KPLS requires from 0.15 to 39 seconds, AKDA/QR takes between 0.35 and 3 seconds, KOC requires between 0.11 and 5 seconds, while SKFE takes between 0.38 to 69 seconds. The optimization step of SKFE is implemented in C++, and the others are implemented in Matlab.

6 Conclusion

We have presented a subspace kernel based on which nonlinear feature extraction can be conducted by kernel

Table 2: Average cross validation error rates (%) and the standard derivations (%) on the seven microarray gene datasets. For each dataset, the results shown in boldface are significantly better than the others, judged by t-test, with a significance level of 0.01.

Dataset	1-NN	KPLS	AKDA/QR	KOC	SKFE
B.Tumor1	14.33±1.65	14.28±1.26	13.89±1.42	14.83±1.58	**12.00±1.42**
B.Tumor2	28.90±2.79	**24.70±3.57**	26.70±3.63	28.30±2.99	27.30±2.27
Leukemia1	11.94±2.13	**5.21±1.49**	7.50±1.65	7.57±1.88	**4.86±1.46**
Leukemia2	6.74±1.64	**3.96±2.27**	7.01±1.77	7.36±1.81	**3.68±0.82**
P.Tumor	24.17±1.94	19.80±1.87	28.82±2.59	20.88±2.38	**14.85±1.63**
DLBCL	14.03±1.72	**4.09±1.21**	14.74±1.80	9.22±1.93	**3.57±0.93**
11_Tumors	16.52±1.09	**11.44±0.85**	12.70±0.89	15.11±1.01	**11.84±0.96**

Table 3: Average test error rates (%) and the standard deviations (%) on the seven text datasets. For each dataset, the results shown in boldface are significantly better than the others, judged by t-test, with a significance level of 0.01.

Dataset	1-NN	KPLS	AKDA/QR	KOC	SKFE
tr11	26.66±2.64	21.71±2.61	23.63±4.32	**15.29±3.14**	**15.78±2.47**
tr23	28.25±4.76	30.05±6.35	27.30±4.88	23.20±4.40	**18.85±3.86**
tr41	20.17±3.02	9.56±1.55	9.59±1.40	7.66±0.96	**6.14±1.08**
tr45	28.48±2.68	23.45±2.24	19.33±3.84	15.23±2.15	**9.85±1.80**
la1	40.82±1.68	19.23±0.69	18.67±0.77	18.28±0.82	**14.51±0.96**
la2	38.82±1.59	16.94±0.76	16.25±1.02	16.36±0.94	**13.23±0.74**
hitech	57.83±1.69	33.14±1.57	31.88±1.08	31.71±1.41	**29.71±1.12**

parameter learning. Connections to related work have been explained. In particular, the comparison with the Spare Large Margin Classifier (SLMC) [Wu *et al.*, 2005] illustrates that useful special cases can be derived from the proposed subspace kernel for some applications. We have also described a method to optimize the subspace kernel by Kernel-Target Alignment (KTA) [Cristianini *et al.*, 2002] maximization. But other kernel parameter learning approaches can also be applied. Finally, experimental results have been provided to validate the effectiveness of our approach.

References

[Chapelle *et al.*, 2002] O. Chapelle, V. Vapnik, O. Bousquet, and S. Mukherjee. Choosing multiple parameters for support vector machines. *Machine Learning*, 46(1-3):131–159, 2002.

[Cristianini *et al.*, 2002] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel-target alignment. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[Kim *et al.*, 2005] H. Kim, P. Howland, and H. Park. Dimension reduction in text classification with support vector machines. *Journal of Machine Learning Research*, 6:37–53, 2005.

[Mika *et al.*, 2001] S. Mika, G. Rätsch, and K. R. Müller. A mathematical programming approach to the kernel fisher algorithm. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, Cambridge, MA, 2001. The MIT Press.

[Park and Park, 2004] C. H. Park and H. Park. Nonlinear feature extraction based on centroids and kernel functions. *Pattern Recognition*, 37:801–810, 2004.

[Rosipal and Trejo, 2001] R. Rosipal and L. J. Trejo. Kernel partial least squares regression in reproducing kernel hilbert space. *Journal of Machine Learning Research*, 2:97–123, 2001.

[Schölkopf and Smola, 2002] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, Cambridge, MA, 2002.

[Vapnik, 1995] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[Weston *et al.*, 2000] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection for svms. In *Advances in Neural Information Processing Systems 12*, Cambridge, MA, 2000. MIT Press.

[Wold, 1975] H. Wold. Soft modeling by latent variables; the nonlinear iterative partial least squares approach. In J. Gani, editor, *Perspectives in Probability and Statistics*, pages 520–540, London, 1975. Academic Press.

[Wu *et al.*, 2005] M. Wu, B. Schölkopf, and G. Bakir. Building sparse large margin classifiers. In L. D. Raedt and S. Wrobel, editors, *Proc. 22th International Conference on Machine Learning*, pages 1001–1008. ACM, 2005.

[Xiong *et al.*, 2005] T. Xiong, J. Ye, Q. Li, R. Janardan, and V. Cherkassky. Efficient kernel discriminant analysis via QR decomposition. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1529–1536. MIT Press, Cambridge, MA, 2005.