Efficient Sparse Kernel Feature Extraction Based on Partial Least Squares

Charanpal Dhanjal, Steve R. Gunn, and John Shawe-Taylor, *Member, IEEE*

Abstract—The presence of irrelevant features in training data is a significant obstacle for many machine learning tasks. One approach to this problem is to extract appropriate features and, often, one selects a feature extraction method based on the inference algorithm. Here, we formalize a general framework for feature extraction, based on Partial Least Squares, in which one can select a user-defined criterion to compute projection directions. The framework draws together a number of existing results and provides additional insights into several popular feature extraction methods. Two new sparse kernel feature extraction methods are derived under the framework, called Sparse Maximal Alignment (SMA) and Sparse Maximal Covariance (SMC), respectively. Key advantages of these approaches include simple implementation and a training time which scales linearly in the number of examples. Furthermore, one can project a new test example using only k kernel evaluations, where k is the output dimensionality. Computational results on several real-world data sets show that SMA and SMC extract features which are as predictive as those found using other popular feature extraction methods. Additionally, on large text retrieval and face detection data sets, they produce features which match the performance of the original ones in conjunction with a Support Vector Machine.

Index Terms—Machine learning, kernel methods, feature extraction, partial least squares (PLS).

1 Introduction

The popularity of machine learning has grown in recent years, as has its use in a variety of novel applications such as visual object detection, speech recognition, and gene selection from microarray data. Many of these applications are characterized by a high number of irrelevant features. Hence, a useful task is to identify relevant features, which is precisely the function of feature extraction methods. Not only does feature extraction improve prediction performance (e.g., with Support Vector Machines (SVMs) [1] in [2] and [3]), but it can provide computational and memory efficiency and aid in gaining a greater understanding of the data.

The type of features one requires are dependent on the inference problem, hence several general approaches to feature extraction have been proposed. One such approach is Projection Pursuit [4], [5], which extracts features by choosing projection vectors and then orthogonalizing repeatedly. A kernel Projection Pursuit method is given in [6] which chooses sparse directions and orthogonalizes in the same way as Principal Components Analysis (PCA) [7].

We adopt a similar methodology, and formalize a general framework for feature extraction based on the orthogonalization procedure used in Partial Least Squares (PLS) [8]. It brings together several existing results as well as augmenting them with further insight. Several useful properties of the framework are inherited from PLS. These include conjugate projection vectors with respect to the data, efficient projection of new test data, and the ability to operate in a kernel-defined feature space. Furthermore, one can specialize the framework to PCA, PLS, Boosted Latent Features (BLF) [9], and their kernel variants.

Using the framework, we derive two new sparse feature extraction methods that are formulated by maximizing covariance and kernel target alignment [10], respectively. The key advantages of these methods are that their computational and memory requirements are linear in the number of examples and the projection of a new test example requires only k kernel evaluations for k output features. Note that these feature extraction methods are extensions of our previous work given in [11]. Here, we present additional analysis of the new methods and derive a stability bound for the cumulative square covariance of their features. Furthermore, an empirical evaluation of the methods for both classification and regression scenarios is conducted on a set of real-world data sets.

This paper is organized as follows: First, a brief introduction to feature extraction and several important feature extraction methods is given. We then present the general framework for feature extraction in both its primal and dual forms and show how it specializes to existing approaches. In the following section, we describe a general approach for obtaining sparse projection directions, then derive and analyze SMA and SMC. Section 5 gives a stability result for the expectation of the cumulative square covariance of the features produced under the framework. Section 6 presents an empirical evaluation of the new feature extraction methods, and this paper concludes with Section 7.

- *C. Dhanjal and S.R. Gunn are with the Information: Signals Images Systems Research Group, School of Electronics and Computer Science, Building 1, University of Southampton, Highfield, Southampton, SO17 1BJ, UK. E-mail: {cd04r, srg}@ecs.soton.ac.uk.*
- *J. Shawe-Taylor is with the Centre for Computational Statistics and Machine Learning, Department of Computer Science, University College London, Gower Street, London, WC1E 6BT, UK. E-mail: jst@cs.ucl.ac.uk.*

Manuscript received 30 Jan. 2007; revised 1 Feb. 2008; accepted 13 June 2008; published online 19 June 2008.
Recommended for acceptance by A. Smola.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0058-0107.
Digital Object Identifier no. 10.1109/TPAMI.2008.171.

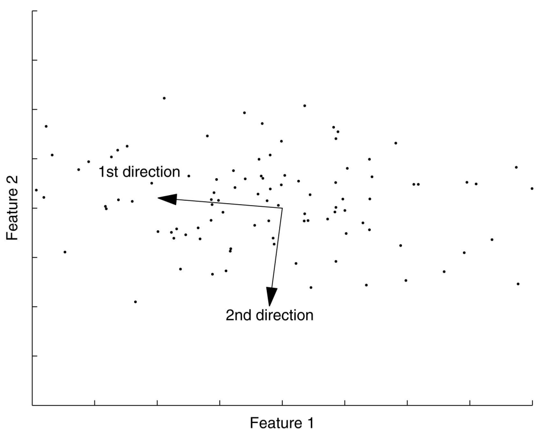


Fig. 1. Plot of the PCA projection vectors for an example 2D data set.

2 Feature Extraction

Let training sample $S = \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}$ contain ℓ observations or examples of a multivariate random variable $\mathbf{x} \in \mathbb{R}^m$ consisting of m features. This sample can be described as a matrix $\mathbf{X} \in \mathbb{R}^{\ell \times m}$ whose rows are composed of the observations. If one is performing supervised learning, the observations also have a corresponding vector of output labels $\mathbf{y} \in \mathbb{R}^\ell$. Feature extraction is the process of finding a new matrix $\hat{\mathbf{X}} \in \mathbb{R}^{\ell \times k}$, $k < m$, which is a low-dimensional representation of the useful information in $\mathbf{X}$.

Broadly, feature extraction methods can be categorized into linear and kernel methods. In the linear case, examples are projected onto the columns of a projection matrix $\mathbf{Z} = [\mathbf{z}_1, \dots \mathbf{z}_k]$, where $\mathbf{z}_1, \dots \mathbf{z}_k$ are projection directions, i.e., $\hat{\mathbf{X}} = \mathbf{XZ}$. For kernel-based feature extraction, the new data is given by the projection $\hat{\mathbf{X}} = \mathbf{KB}$, where $\mathbf{K}$ is a kernel matrix and $\mathbf{B}$ is a projection matrix whose columns are known as the dual projection vectors. One can see the equivalence to the primal case if the linear kernel is used, i.e., $\mathbf{K} = \mathbf{XX}'$, and $\mathbf{Z} = \mathbf{X}'\mathbf{B}$. If a nonlinear kernel is used, then one can imagine the examples living in a high-dimensional feature space and $\hat{\mathbf{X}}$ is a projection of the examples in that space. The resulting mapping in the input space is nonlinear.

In the remainder of this section, we outline several established feature extraction methods (e.g., PCA, PLS), as well as more recent advances.

2.1 Unsupervised Learning

We start with PCA which finds projection directions that explain the variance of the data (illustrated in Fig. 1). Often, the majority of this variance can be captured using a much smaller dimensionality than that of the original data. The residual variance can be seen as noise in the data, hence is not useful for learning. The PCA projection directions $\mathbf{u}_1, \dots, \mathbf{u}_k$ are given by the dominant eigenvectors of the covariance matrix $\mathbf{X}'\mathbf{X}$. Since this matrix is symmetric, both the projection directions and extracted features $\mathbf{Xu}_j$, $j = 1, \dots, k$, are orthogonal. Furthermore, when these features are used for least squares regression, the resulting combined approach is often referred to Principal Components Regression (PCR) [12].

Kernel PCA (KPCA) was introduced in [13] to address the problem of finding nonlinear features. In this case, one finds the dominant eigenvectors of the kernel matrix $\mathbf{K}$ and then projects $\mathbf{K}$ onto these directions. Let $\boldsymbol{\alpha}_j$ be the jth eigenvector of $\mathbf{K}$, then the KPCA features are given by $\hat{\mathbf{X}} = \mathbf{KA}\boldsymbol{\Lambda}^{-\frac{1}{2}}$, where $\mathbf{A}$ is the matrix with columns $\boldsymbol{\alpha}_j$, $j = 1, \dots, k$, and $\boldsymbol{\Lambda}$ is a diagonal matrix of corresponding eigenvalues.

With (K)PCA, the cost of solving the eigenvalue problem scales cubically in its size, and the resulting eigenvectors are difficult to interpret since they lack sparseness. Several authors have addressed these drawbacks. In [6], one finds a sparse projection direction that maximizes variance and then orthogonalizes the examples by projecting them into the space orthogonal to the direction, a process known as deflation. The directions are sparse since they are scalar multiples of a single deflated example. Later, it will become clear how this method is related to our approach to feature extraction. Both [14] and [15] address the NP-hard and nonconvex problem of maximizing variance subject to a cardinality constraint. In [14], the problem is relaxed to form a convex semidefinite programming formulation. In [15], both greedy and exact methods for solving the cardinality-constrained optimization are presented using insights between the eigenvalues of a positive definite covariance matrix and the eigenvalues of its submatrices. These results are generalized for Linear Discriminant Analysis (LDA) in [16].

2.2 Supervised Learning

Since feature extraction is often used as a precursor to prediction, there is a requirement for supervised algorithms, and PLS is a popular choice. It was initially popular in the field of chemometrics, where high-dimensional data are correlated representations are common, and has recently gained favor in the machine learning community. It was first introduced in [8] for regression tasks and has since seen many variants. For example, PLS can be used for classification [17] by setting up $\mathbf{Y}$ to be a "class membership" matrix and making a prediction for a new point using the closest cluster center with the same label. Here, we outline the method often called PLS2 or PLS regression, which we will simply refer to as PLS. A simple introduction to PLS is given in [18] and a general survey of PLS variants is presented in [19].

Essentially, PLS repeatedly extracts features which are most covariant with the labels, and then deflates. One then performs least squares regression on the extracted features. The projection direction $\mathbf{u}_j$ at the jth iteration is given by the left singular vector of the covariance matrix $\mathbf{X}_j'\mathbf{Y}$, where $\mathbf{X}_j$ is the jth residual data matrix (described shortly) and $\mathbf{Y}$ [1] is a matrix whose rows are the output labels. These directions correspond to those which maximize the covariance between the data and the labels.

The residual matrices $\mathbf{X}_j$, $j = 1, \dots, k$, are computed by projecting the columns of the data onto the orthogonal complement of $\mathbf{X}_1\mathbf{u}_1, \dots, \mathbf{X}_{j-1}\mathbf{u}_{j-1}$. The projection of the columns of $\mathbf{X}$ onto the space orthogonal to unit vector $\mathbf{w}$ is given by $\mathbf{X} \leftarrow (\mathbf{I} - \mathbf{ww}')\mathbf{X}$; hence, the PLS deflation is

$$\mathbf{X}_{j+1} = \left(\mathbf{I} - \frac{\mathbf{X}_j \mathbf{u}_j \mathbf{u}_j' \mathbf{X}_j'}{\mathbf{u}_j' \mathbf{X}_j' \mathbf{X}_j \mathbf{u}_j} \right) \mathbf{X}_j, \tag{1}$$

with $\mathbf{X}_1 = \mathbf{X}$.

1. For the classification variant of PLS, $\mathbf{Y}$ is substituted with $\tilde{\mathbf{Y}} = \mathbf{Y}(\mathbf{Y}'\mathbf{Y})^{-1/2}$ so that $\tilde{\mathbf{Y}}'\tilde{\mathbf{Y}} = \mathbf{I}$.

The extracted features $\mathbf{X}_j\mathbf{u}_j$ are computed in terms of the original data as $\hat{\mathbf{X}} = \mathbf{X}\mathbf{U}(\mathbf{P}'\mathbf{U})^{-1}$, where $\mathbf{U}$ has columns $\mathbf{u}_j$ and $\mathbf{P}$ has columns $\mathbf{p}_j = \mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j/\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j$, $j = 1, \ldots, k$. Details of the derivation of this result are given in [20]. Since the matrix $\mathbf{P}'\mathbf{U}$ is upper triangular (also tridiagonal), its inversion can be efficiently computed as the solution of k linear equations with k unknowns. An important property of the above deflation is that the extracted features are orthogonal. This ensures that the least squares regression coefficients, given by $\mathbf{C} = (\hat{\mathbf{X}}'\hat{\mathbf{X}})^{-1}\hat{\mathbf{X}}'\mathbf{Y}$, where $\hat{\mathbf{X}}'\hat{\mathbf{X}}$ is a diagonal matrix, are efficient to compute.

Several interesting connections have been made between PLS and other feature extraction methods. In particular, [21] unifies PCR, PLS, and Ordinary Least Squares (OLS) regression under a general approach called Continuum Regression (CR). A key component of CR is the following criterion:

$$T = (\mathbf{u}'\mathbf{X}'\mathbf{y})^2(\mathbf{u}'\mathbf{X}'\mathbf{X}\mathbf{u})^{\alpha/(1-\alpha)-1},$$

which is maximized to form the OLS[2] ($\alpha = 0$), PLS ($\alpha = \frac{1}{2}$), and PCR ($\alpha = 1$) directions, respectively. After selecting a direction, one uses deflation (1) and repeats. Our general approach to feature extraction encompasses these insights between PCA and PLS under a more general setting, and extends the results to the corresponding kernel variants.

This leads us onto the kernel version of PLS, which was introduced in [22] and shown to exhibit good performance with an SVM in [23]. In the primal case, the projection directions are in the row space of $\mathbf{X}_j$, i.e., $\mathbf{u}_j = \mathbf{X}_j'\boldsymbol{\beta}_j$ for some $\boldsymbol{\beta}_j$. One can introduce a set of dual features $\boldsymbol{\tau}_j = \mathbf{X}_j\mathbf{u}_j = \mathbf{X}_j\mathbf{X}_j'\boldsymbol{\beta}_j = \mathbf{K}_j\boldsymbol{\beta}_j$ and the kernel matrix is deflated in the following way:

$$\mathbf{K}_{j+1} = \left(\mathbf{I} - \frac{\boldsymbol{\tau}_j\boldsymbol{\tau}_j'}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j}\right)\mathbf{K}_j\left(\mathbf{I} - \frac{\boldsymbol{\tau}_j\boldsymbol{\tau}_j'}{\boldsymbol{\tau}_j'\boldsymbol{\tau}_j}\right),$$

which only requires kernel matrices.

The projection matrix required to extract features from a new test example starts with the primal projection matrix $\mathbf{U}(\mathbf{P}'\mathbf{U})^{-1}$. Using $\mathbf{U} = \mathbf{X}'\mathbf{B}$ and $\mathbf{P} = \mathbf{X}'\mathbf{T}(\mathbf{T}'\mathbf{T})^{-1}$, we obtain $\mathbf{Z} = \mathbf{X}'\mathbf{B}((\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{K}\mathbf{B})^{-1}$, where $\mathbf{B}$ is a matrix with columns $\boldsymbol{\beta}_j$, $j = 1, \ldots, k$. Hence, the projection of a new test example $\phi(\mathbf{x})$ is given by $\hat{\phi}(\mathbf{x})' = \mathbf{k}'\mathbf{B}((\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{K}\mathbf{B})^{-1}$, where $\mathbf{k}$ is a vector of inner products between the test point and the training examples, see [22] for details.

We now turn our attention to a pair of recent feature extraction methods inspired by Boosting [24]. The first of these is Kernel Boosting (KB) [25], which is an iterative approach for learning features using binary labeled data. At each iteration, one selects a projection direction and forms a kernel matrix using the projected examples. Each element of this kernel matrix is then reweighted to reflect the difficulty in predicting whether the corresponding pair of labels is the same. The intuition behind this strategy is to focus learning on those label pairs which are difficult to predict.

BLF is another Boosting-based algorithm; however, it is easier to think of it as a generalization of PLS. It allows one to select a user-defined loss function between the set of extracted features and labels. This criterion is used to select projection directions, which are in turn used to deflate the data in the same manner as PLS. The resulting features are then used for regression by minimizing the same user-defined loss function. Several example loss functions are given in [9], including least squares (which specializes BLF to PLS), 1-norm, and exponential loss. As we shall see later, our approach to feature extraction is also a generalization of PLS, but it allows for arbitrary projection directions. This allows it to specialize to BLF as well as its kernel variant, kernel BLF (KBLF).

The supervised algorithms presented so far use all of the training examples for computing projection directions. Sparse KPLS [26], however, computes projection directions using only a fraction $\nu \in [0, 1]$ of the training examples, where ν is user controlled. Each dual projection direction is found using an ϵ-insensitive loss function similar to that used in Support Vector Regression (SVR) [27] and the examples are deflated by projecting onto the space orthogonal to this direction. One of the main problems with this approach is the high computational cost of finding a direction at each iteration. This point is addressed in [28] which suggests two heuristics for finding the sparse projection vectors that have the advantage of reducing computation and not requiring the full kernel matrix in memory. The same general approach is applied to KBLF to form sparse KBLF.

An alternative sparse PLS approach takes the form of Reduced Kernel Orthonormalized PLS (rKOPLS) [29]. The authors develop rKOPLS from Orthonormalized PLS [30], which minimizes $\|\mathbf{Y} - \mathbf{X}\mathbf{U}\|_F^2$ (where $\|\cdot\|_F$ is the Frobenius norm) for centered $\mathbf{X}$ and $\mathbf{Y}$. The dual form of this optimization is equivalent to solving the eigenvalue problem

$$\mathbf{K}_x\mathbf{K}_y\mathbf{K}_x\boldsymbol{\beta} = \lambda\mathbf{K}_x\mathbf{K}_x\boldsymbol{\beta}, \qquad (2)$$

where $\mathbf{K}_x = \mathbf{X}\mathbf{X}'$ and $\mathbf{K}_y = \mathbf{Y}\mathbf{Y}'$. A deflation-based strategy is given for solving the above, which can iterate at most $\mathrm{rank}(\mathbf{Y})$ times. To form rKOPLS, the primal projection directions are selected from the space of a random subset of size r of the examples. Let this set of examples form the rows of $\mathbf{X}_r$ and $\mathbf{K}_x^r = \mathbf{X}_r\mathbf{X}'$, then rKOPLS solves $\mathbf{K}_x^r\mathbf{K}_y\mathbf{K}_x^{r\prime}\boldsymbol{\beta} = \lambda\mathbf{K}_x^r\mathbf{K}_x^{r\prime}\boldsymbol{\beta}$, which is computationally cheaper than (2).

3 Generalized Feature Extraction

The general framework for feature extraction is an approach to dimensionality reduction based on the PLS deflation method. It comes under projection pursuit techniques and also parallels the method used in [6], which is based on the PCA deflation. It provides a useful formalization of existing results, and additional insight into several popular feature extraction methods.

3.1 Primal Feature Extraction

We start with an analysis of the primal representation; however, it is easily extended to the dual case to allow the use of kernels. First of all, consider Algorithm 1, which shows the general feature extraction method. Essentially, the method operates iteratively, selecting a new feature direction $\mathbf{u}_j$ at iteration j and then deflating the data matrix $\mathbf{X}_j$ by projecting its columns into the space orthogonal to $\mathbf{X}_j\mathbf{u}_j$:

2. In this case, the resulting projection direction is equivalent to the vector of least square correlation coefficients and the algorithm stops at a single iteration.

$$\mathbf{X}_{j+1} = \left(\mathbf{I} - \frac{\mathbf{X}_j\mathbf{u}_j\mathbf{u}_j'\mathbf{X}_j'}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}\right)\mathbf{X}_j,$$

which is identical to the PLS deflation.

There is one requirement that we impose on the choice of $\mathbf{u}_j$, that it should be in the row space of $\mathbf{X}$, i.e., for some β_j, $\mathbf{u}_j = \mathbf{X}'\beta_j$. This ensures a dual representation of the extracted features $\mathbf{X}_j\mathbf{u}_j$, $j = 1, \ldots, k$. Observe that these features are orthogonal since they are a linear combination of the columns of $\mathbf{X}_j$ that have been repeatedly projected into the orthogonal complement of previous $\mathbf{X}_i\mathbf{u}_i$, for $i < j$.

Algorithm 1. Pseudocode for primal general feature extraction

```
Inputs: Data matrix $\mathbf{X} \in \mathbb{R}^{\ell\times m}$, optional target vectors
  $\mathbf{Y} \in \mathbb{R}^{\ell\times n}$, dimension $k$
Process:
  1) $\mathbf{X}_1 = \mathbf{X}$
  2) for $j = 1, \ldots, k$
       a) Select $\mathbf{u}_j$ from the span of the rows of $\mathbf{X}$
       b) $\mathbf{X}_{j+1} = \left(\mathbf{I} - \frac{\mathbf{X}_j\mathbf{u}_j\mathbf{u}_j'\mathbf{X}_j'}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}\right)\mathbf{X}_j$
  3) end
Output: Directions $\mathbf{u}_j$ and features $\mathbf{X}_j\mathbf{u}_j$, $j = 1, \ldots, k$
```

The apparent disadvantage with this simple description is that the features $\mathbf{X}_j\mathbf{u}_j$ are defined relative to the deflated matrices $\mathbf{X}_j$. We would, however, like to be able to compute the extracted features directly from the original feature vectors so that one can extract features from a test set for example. The extracted features for a test point with feature vector $\phi(\mathbf{x})$ are given by

$$\hat{\phi}(\mathbf{x})' = \phi(\mathbf{x})'\mathbf{U}(\mathbf{P}'\mathbf{U})^{-1},$$

where the matrices $\mathbf{U}$ and $\mathbf{P}$ have their columns composed of the vectors $\mathbf{u}_j$ and $\mathbf{p}_j = \mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j/\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j$, $j = 1, \ldots, k$, respectively. The derivation of this result is identical to that of PLS. Furthermore, $\mathbf{P}'\mathbf{U}$ is upper triangular with diagonal 1 as in PLS; hence, its inverse is efficient to compute.

The vectors of feature values across the training set $\mathbf{X}_j\mathbf{u}_j$ are orthogonal and can be written as $\mathbf{X}\mathbf{U}(\mathbf{P}'\mathbf{U})^{-1}$; hence, $(\mathbf{U}'\mathbf{P})^{-1}\mathbf{U}'\mathbf{X}'\mathbf{X}\mathbf{U}(\mathbf{P}'\mathbf{U})^{-1}$ is a diagonal matrix. This conjugacy of the projection directions with respect to the deflated matrices ensures that the resulting features are as dissimilar as possible and that least squares regression coefficients can be efficiently computed.

3.1.1 Specialization to PCA

The connection between the PLS deflation and PCA was first identified in [21] and here we give an equivalent result using the iterative solution to PCA. To obtain PCA from Algorithm 1, one simply chooses $\mathbf{u}_j$ to be the first eigenvector of $\mathbf{X}_j'\mathbf{X}_j$. In this case, the effect of the deflation of $\mathbf{X}_j$ at each iteration can be seen as shrinking the largest eigenvalue of $\mathbf{X}_j'\mathbf{X}_j$ to zero since, $\mathbf{X}_{j+1}'\mathbf{X}_{j+1}\mathbf{u}_j = \hat{\lambda}_j\mathbf{u}_j = \mathbf{0}$, where $\hat{\lambda}_j$ is the eigenvalue corresponding to the eigenvector $\mathbf{u}_j$ of the deflated matrix $\mathbf{X}_{j+1}'\mathbf{X}_{j+1}$ and $\mathbf{u}_j \neq \mathbf{0}$. It follows that the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ extracted from this process are exactly the first k eigenvectors of $\mathbf{X}'\mathbf{X}$, and hence those needed for projecting the data as in the PCA method. As we are extracting the features iteratively, techniques such as the Power method [31] can be used to efficiently extract the first eigenvector at each iteration.

The resulting features $\mathbf{X}_1\mathbf{u}_1, \ldots, \mathbf{X}_k\mathbf{u}_k$ are also equivalent to those found by PCA since

$$\begin{aligned}\mathbf{X}_{j+1}\mathbf{u}_{j+1} &= \left(\mathbf{I} - \frac{\mathbf{X}_j\mathbf{u}_j\mathbf{u}_j'\mathbf{X}_j'}{\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}\right)\mathbf{X}_j\mathbf{u}_{j+1}\\ &= \left(\mathbf{X}_j - \frac{\mathbf{X}_j\mathbf{u}_j\mathbf{u}_j'}{\mathbf{u}_j'\mathbf{u}_j}\right)\mathbf{u}_{j+1}\\ &= \mathbf{X}_j\mathbf{u}_{j+1}\\ &= \mathbf{X}\mathbf{u}_{j+1},\end{aligned}$$

where the third line follows from the orthogonality of $\mathbf{u}_j$, $j = 1, \ldots, k$.

3.1.2 Specialization to PLS

If we consider the PLS algorithm, then this is also easily placed within the framework. In this case, $\mathbf{u}_j$ is chosen to be the first singular vector of $\mathbf{X}_j'\mathbf{Y}$. Furthermore, for the PLS case where one wishes to not just select features but also compute the overall regression coefficients for the primal PLS problem, they can be computed as $\mathbf{W} = \mathbf{U}(\mathbf{P}'\mathbf{U})^{-1}\mathbf{C}'$, where $\mathbf{C}$ is the matrix with columns $\mathbf{c}_j = \mathbf{Y}'\mathbf{X}_j\mathbf{u}_j/\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j$.

3.1.3 Specialization to BLF

To position BLF within the general framework, we require the projection directions to be computed as $\mathbf{u}_j = \mathbf{X}_j'\mathbf{w}_j/\|\mathbf{X}_j\mathbf{X}_j'\mathbf{w}_j\|$, where $\mathbf{w}_j$ is the negative gradient of the loss function. The scaling on $\mathbf{u}_j$ is required so that the extracted features have unit norm. Since the BLF deflation strategy is identical to that of the general framework, the resulting features are also the same. Furthermore, using the resulting features BLF computes regression coefficients, and the interested reader is directed to [9] for the derivation of loss function gradients and regression coefficients of several loss functions.

3.2 Kernel Feature Extraction

Here, we give a dual variable formulation of the framework, which allows feature extraction methods to be used in conjunction with kernels. The requirement given in Algorithm 1 that $\mathbf{u}_j$ be in the space of the rows of $\mathbf{X}$ implies that there is a vector β_j such that $\mathbf{u}_j = \mathbf{X}'\beta_j$. As we are using kernels, we do not want to compute the feature vectors explicitly and must work with the residual kernel matrix $\mathbf{K}_j$ at each stage. Given a choice of dual variables β_j, let $\tau_j = \mathbf{X}_j\mathbf{u}_j = \mathbf{K}_j\beta_j$. The deflation of $\mathbf{X}_j$ is given by $\mathbf{X}_{j+1} = (\mathbf{I} - \tau_j\tau_j'/\tau_j'\tau_j)\mathbf{X}_j$; hence, the kernel matrix is deflated by

$$\mathbf{K}_{j+1} = \left(\mathbf{I} - \frac{\tau_j\tau_j'}{\tau_j'\tau_j}\right)\mathbf{K}_j, \tag{3}$$

which is computable without explicit feature vectors. This deflation is different to the dual-sided one used in KPLS; however, it is more useful for sparse feature extraction and we shall see later that it still allows for the specialization to KPLS. The general kernel feature extraction method is given in Algorithm 2.

Algorithm 2. Pseudocode for general kernel feature extraction

```
Input: Kernel matrix $\mathbf{K} \in \mathbb{R}^{\ell\times\ell}$, optional target vectors
  $\mathbf{Y} \in \mathbb{R}^{\ell\times n}$, dimension $k$
```

```
Process:
  1) $\mathbf{K}_1 = \mathbf{K}$
  2) for $j = 1, \ldots, k$
       a) Choose dual vectors $\beta_j \in \mathbb{R}^\ell$
       b) Let $\tau_j = \mathbf{K}_j \beta_j$
       c) $\mathbf{K}_{j+1} = \left(\mathbf{I} - \frac{\tau_j \tau_j'}{\tau_j' \tau_j}\right)\mathbf{K}_j$
  3) end
Output: Dual vectors $\beta_j$ and features $\tau_j$, $j = 1, \ldots, k$.
```

In the primal case, we were able to give a closed form expression for the projection of a new test point, and we would like to obtain a similar result for the dual variable case. The derivation of the projection of a new point starts with the primal representation and is identical to that of kernel PLS. Again, we have $\mathbf{U} = \mathbf{X}'\mathbf{B}$ and $\mathbf{P} = \mathbf{X}'\mathbf{T}(\mathbf{T}'\mathbf{T})^{-1}$, where $\mathbf{B}$ is the matrix with columns β_j and $\mathbf{T}$ has columns τ_j, $j = 1, \ldots, k$. Hence, the final features are given by

$$\phi(\mathbf{x})'\mathbf{U}(\mathbf{P}'\mathbf{U})^{-1} = \mathbf{k}'\mathbf{B}\Big((\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{K}\mathbf{B}\Big)^{-1}, \tag{4}$$

where $\mathbf{k}$ is a vector of inner products between the test point and the training examples.

3.2.1 Specialization to KPCA

In KPCA, one usually performs a full eigendecomposition of the kernel matrix $\mathbf{K} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}'$ and then uses the first k eigenvectors to project the data onto the first k principal components. By choosing β_j to be $\mathbf{v}_j/\sqrt{\lambda_j}$, where $\mathbf{v}_j$, λ_j are the first eigenvector-eigenvalue pair of $\mathbf{K}_j$, and then deflating, the eigenvalue corresponding to β_j is set to zero:

$$\begin{aligned}\hat{\lambda}_j \beta_j &= \mathbf{K}_{j+1}\beta_j \\ &= \tau_j - \frac{\tau_j \tau_j' \tau_j}{\tau_j' \tau_j} \\ &= \mathbf{0},\end{aligned}$$

where $\hat{\lambda}_j$ is the eigenvalue of $\mathbf{K}_{j+1}$ corresponding to β_j and $\beta_j \neq \mathbf{0}$. Hence, the matrix $\mathbf{B}$ corresponds to the first k columns of $\mathbf{V}\boldsymbol{\Lambda}^{-\frac{1}{2}}$, and (4) gives the required result $\hat{\phi}(\mathbf{x})' = \mathbf{k}'\mathbf{B}$.

3.2.2 Specialization to KPLS

For the kernel variant of PLS, we need to deflate both the $\mathbf{X}$ and $\mathbf{Y}$ matrices. The deflation of $\mathbf{Y}$ in the primal version is redundant; however, in the dual case, it is required in order to obtain the dual representations of the extracted features. At each iteration, β_j is chosen to be the first eigenvector of $\mathbf{Y}_j\mathbf{Y}_j'\mathbf{K}_j$, normalized so that $\beta_j'\mathbf{K}_j\beta_j = 1$. The kernel matrix $\mathbf{K}_j$ is deflated as per the general framework and the same deflation is also applied to the columns of $\mathbf{Y}_j$, i.e., $\mathbf{Y}_{j+1} = (\mathbf{I} - \tau_j\tau_j'/\tau_j'\tau_j)\mathbf{Y}_j$.

To place KPLS within the general framework, we therefore need to modify Algorithm 2 as follows: At the start, we let $\mathbf{Y}_1 = \mathbf{Y}$ and at each iteration we choose β_j to be the normalized first eigenvector of $\mathbf{Y}_j\mathbf{Y}_j'\mathbf{K}_j$. After deflating $\mathbf{K}_j$, we must also deflate $\mathbf{Y}_j$ by the process outlined above. As β_j lies in the space orthogonal to $\tau_1, \ldots, \tau_{j-1}$, the final features $\tau_j = \mathbf{K}_j\beta_j$, $j = 1, \ldots, k$, are the same as those obtained using KPLS despite $\mathbf{K}_j$ not being deflated on its right side.

We can compute the regression coefficients for KPLS as $\mathbf{c}_j = \mathbf{Y}_j'\mathbf{X}_j\mathbf{u}_j/\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j = \mathbf{Y}'\tau_j/\tau_j'\tau_j$, making $\mathbf{C} = \mathbf{Y}'\mathbf{T}(\mathbf{T}'\mathbf{T})^{-1}$. Putting this together with (4), the dual regression variables are given by $\alpha = \mathbf{B}(\mathbf{T}'\mathbf{K}\mathbf{B})^{-1}\mathbf{T}'\mathbf{Y}$, which is identical to the expression given in [22].

3.2.3 Specialization to KBLF

As one might expect, KBLF is closely related to the kernel general feature extraction framework. Since the kernel matrix in KBLF is deflated on both sides, projection directions are given by $\mathbf{u}_j = q\mathbf{X}_j'\mathbf{w}_j = q\mathbf{X}'(\mathbf{I} - \mathbf{T}_{j-1}(\mathbf{T}_{j-1}'\mathbf{T}_{j-1})^{-1}\mathbf{T}_{j-1}')\mathbf{w}_j$, where $\mathbf{w}_j$ is the negative gradient of the loss function, $\mathbf{T}_j$ is a matrix composed of the first j columns of $\mathbf{T}$, and q is a normalization constant. Clearly, at each iteration, we have $\beta_j = (\mathbf{I} - \mathbf{T}_{j-1}(\mathbf{T}_{j-1}'\mathbf{T}_{j-1})^{-1}\mathbf{T}_{j-1}')\mathbf{w}_j$, normalized so that $\|\tau_j\|^2 = \beta_j'\mathbf{K}_j'\mathbf{K}_j\beta_j = 1$. Hence, the resulting features are computed as per (4), with $(\mathbf{T}'\mathbf{T})^{-1} = \mathbf{I}$, so that $\hat{\phi}(\mathbf{x})' = \mathbf{k}'\mathbf{B}(\mathbf{T}'\mathbf{K}\mathbf{B})^{-1}$.

4 Sparse Feature Extraction

Many of the kernel-based feature extraction methods considered so far are not sparse: To obtain projections for new test points, all training examples are often needed. Sparseness, however, is a desirable property, providing computation and efficiency benefits, and as such it is often the case that one is willing to see a small reduction in performance if a high degree of sparseness is achieved.

In common with the method adopted in [6], a sparse representation on β_j is achieved by choosing a vector with only one nonzero entry. If β_j has a nonzero entry at its ith position, then $\mathbf{u}_j$ is a scalar multiple of the ith deflated example and $\tau_j = \mathbf{K}_j\beta_j$ is a scalar multiple of the ith column of $\mathbf{K}_j$. Hence, after k iterations, only k residual training examples will contribute to the projection vectors. Using this approach, we derive two novel supervised sparse feature extraction methods.

4.1 Sparse Maximal Alignment

Our first sparse algorithm is called Sparse Maximal Alignment (SMA) and it is based on the kernel alignment between two kernel matrices $\mathbf{K}_1$ and $\mathbf{K}_2$:

$$A(\mathbf{K}_1, \mathbf{K}_2) = \frac{\langle \mathbf{K}_1, \mathbf{K}_2 \rangle_F}{\sqrt{\langle \mathbf{K}_1, \mathbf{K}_1 \rangle_F \langle \mathbf{K}_2, \mathbf{K}_2 \rangle_F}},$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product. Alignment measures the similarity between two kernel matrices and can be considered as the cosine of the angle between the concatenated rows of $\mathbf{K}_1$ and $\mathbf{K}_2$. It is a useful quantity to optimize since the alignment on a training set is likely to be similar to that of a test set. This result follows from the sharp concentration of alignment, proven in [10], which means that the probability of the empirical estimate of alignment deviating from its mean is an exponentially decaying function of that deviation.

SMA is derived by maximizing the kernel alignment between the kernel matrix given by projecting the examples, and the kernel matrix of the corresponding labels. The former kernel matrix is $\mathbf{K}_\mathbf{u} = \mathbf{X}\mathbf{u}\mathbf{u}'\mathbf{X}' = \mathbf{K}\beta\beta'\mathbf{K}'$ and has an alignment with the labels (also known as the kernel target alignment) of

$$A(\mathbf{K_u}, \mathbf{yy}') = \frac{\mathbf{y}'\mathbf{K}\beta\beta'\mathbf{K}'\mathbf{y}}{\mathbf{y}'\mathbf{y}\|\mathbf{K}\beta\beta'\mathbf{K}'\|_F} = \frac{(\mathbf{y}'\mathbf{K}\beta)^2}{\mathbf{y}'\mathbf{y}\beta'\mathbf{K}'\mathbf{K}\beta}.$$

If this alignment is 1, the kernel matrix elements are proportional to the corresponding inner products between the labels, i.e., $\kappa_{\mathbf{u}}(\mathbf{x}_i, \mathbf{x}_j) = cy_iy_j$, where c is a scaling factor and $\kappa_{\mathbf{u}}$ is the kernel function used for $\mathbf{K_u}$. By computing c, this implies a simple method for making a prediction for a test point $\mathbf{x}$.

It follows from the above that, at each iteration of SMA, $\boldsymbol{\beta}_j$ is solved using

$$\begin{aligned} \max \quad & \boldsymbol{\beta}_j'\mathbf{K}_j'\mathbf{y} \\ \text{s.t.} \quad & \boldsymbol{\beta}_j'\mathbf{K}_j'\mathbf{K}_j\boldsymbol{\beta}_j = 1 \\ & \operatorname{card}(\boldsymbol{\beta}_j) = 1, \end{aligned} \tag{5}$$

where $\operatorname{card}(\cdot)$ is the number of nonzero elements of its vector input. The solution can be found in an iterative manner by selecting each element of $\boldsymbol{\beta}_j$ in turn as the nonzero entry and choosing the one which gives maximal alignment.

4.2 Sparse Maximal Covariance

Our second sparse algorithm, Sparse Maximal Covariance (SMC), maximizes the empirical expectation of the covariance between the examples and their labels, subject to the above sparsity constraint. The empirical expectation of the covariance is

$$\hat{\mathbb{E}}\left[y\phi(\mathbf{x})'\mathbf{u}\right] = \frac{1}{\ell}\mathbf{u}'\mathbf{X}'\mathbf{y} = \frac{1}{\ell}\boldsymbol{\beta}'\mathbf{K}'\mathbf{y},$$

for zero mean y and $\phi(\mathbf{x})'\mathbf{u}$. Using a unit norm constraint on the primal projection vector, $\boldsymbol{\beta}_j$ is found with

$$\begin{aligned} \max \quad & \boldsymbol{\beta}_j'\mathbf{K}_j'\mathbf{y} \\ \text{s.t.} \quad & \boldsymbol{\beta}_j'\mathbf{K}\boldsymbol{\beta}_j = 1 \\ & \operatorname{card}(\boldsymbol{\beta}_j) = 1, \end{aligned} \tag{6}$$

where the first constraint follows from $\mathbf{u}_j'\mathbf{u}_j = \boldsymbol{\beta}_j'\mathbf{X}\mathbf{X}'\boldsymbol{\beta}_j = \boldsymbol{\beta}_j'\mathbf{K}\boldsymbol{\beta}_j = 1$. Notice the similarity to (5), which, unlike (6), is not influenced by directions with large variance.

As a final point on SMC, we note that the above definition of covariance assumes that the data is centered; however, with sparse data, the sparsity is often lost through centering. Centering the data reduces the sum of the eigenvalues of the kernel matrix, and removes irrelevant variance due to a shift in the center of mass. With uncentered data, the first SMC direction represents this variance and, hence, the initial deflation acts as an approximation to the centering operation.

4.3 Cost of Sparsity

Computing projection directions in the way outlined above has many advantages; however, sparsity does come at a price and one would like to know if the benefits justify the disadvantages. Here, we derive what we will call p-sparse projection directions, which are those generated using at most p examples per direction, and compare them to 1-sparse directions.

First, consider finding a p-sparse projection direction which maximizes covariance, i.e., one which solves the following optimization:

$$\begin{aligned} \max \quad & \boldsymbol{\beta}_j'\mathbf{K}_j'\mathbf{y} \\ \text{s.t.} \quad & \boldsymbol{\beta}_j'\mathbf{K}\boldsymbol{\beta}_j = 1 \\ & \operatorname{card}(\boldsymbol{\beta}_j) \leq p. \end{aligned} \tag{7}$$

The solution can be found using a combinatorial approach. If one is already aware of which entries in $\boldsymbol{\beta}_j$ are nonzero and these indices are given by $\{i_1, \ldots, i_q\} \in [\ell]$, $q \leq p$, then the above problem can be rephrased by substituting $\boldsymbol{\beta}_j = \mathbf{E}\boldsymbol{\alpha}_j$, where $\mathbf{E} = [\mathbf{e}_{i_1}, \ldots, \mathbf{e}_{i_q}]$ and $\mathbf{e}_i$ is the ith standard vector. Hence, the new optimization is

$$\begin{aligned} \max \quad & \boldsymbol{\alpha}_j'\mathbf{E}'\mathbf{K}_j'\mathbf{y} \\ \text{s.t.} \quad & \boldsymbol{\alpha}_j'\mathbf{E}'\mathbf{K}\mathbf{E}\boldsymbol{\alpha}_j = 1, \end{aligned} \tag{8}$$

where the cardinality constraint has been removed. Using the Lagrangian method, the solution to optimization (8) is given by $\mathbf{E}'\mathbf{K}_j'\mathbf{y} = \lambda_j\mathbf{E}'\mathbf{K}\mathbf{E}\boldsymbol{\alpha}_j$, where λ_j is a scaling factor. Of course, finding the nonzero indices of $\boldsymbol{\beta}_j$ in the first place is a costly procedure. One must search all subsets of the numbers in $[\ell]$ of size at most p. The complexity of computing a projection direction using this approach scales proportional to $\binom{\ell}{p}$, $p \leq \ell$, where $\binom{\cdot}{\cdot}$ is the choose function. However, this method suffices to compute p-sparse directions for small p.

In the case where one wishes to find a p-sparse direction which maximizes kernel target alignment, it can be found using

$$\begin{aligned} \max \quad & \boldsymbol{\beta}_j'\mathbf{K}_j'\mathbf{y} \\ \text{s.t.} \quad & \boldsymbol{\beta}_j'\mathbf{K}_j'\mathbf{K}_j\boldsymbol{\beta}_j = 1 \\ & \operatorname{card}(\boldsymbol{\beta}_j) \leq p. \end{aligned} \tag{9}$$

Applying the same technique by substituting $\boldsymbol{\beta}_j = \mathbf{E}\boldsymbol{\alpha}_j$ yields

$$\begin{aligned} \max \quad & \boldsymbol{\alpha}_j'\mathbf{E}'\mathbf{K}_j'\mathbf{y} \\ \text{s.t.} \quad & \boldsymbol{\alpha}_j'\mathbf{E}'\mathbf{K}_j'\mathbf{K}_j\mathbf{E}\boldsymbol{\alpha}_j = 1, \end{aligned} \tag{10}$$

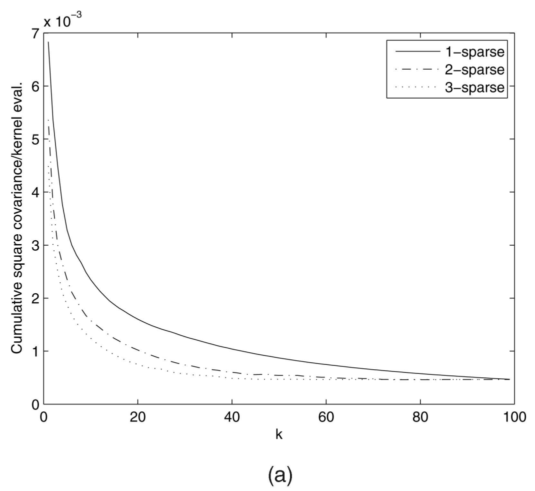
and, via the Lagrangian approach, the solution is found using $\mathbf{E}'\mathbf{K}_j'\mathbf{y} = \lambda_j\mathbf{E}'\mathbf{K}_j'\mathbf{K}_j\mathbf{E}\boldsymbol{\alpha}_j$.

These sparse directions are now compared using a synthetic data set consisting of 100 examples and 1,000 features, with labels computed using a linear combination of 50 of the features plus a small uniform noise component. The data is first centered and normalized so that each feature has unit norm, and (7) is solved for $p \in \{1, 2, 3\}$ on the resulting data. Note that p-sparse projection directions, $p > 3$, proved to be too computationally expensive on anything other than very small data sets. The extracted features are compared using the cumulative square covariance per kernel evaluation. For a sample S, this is given by $B(S)/r$, where

$$B(S) = \sum_{j=1}^{k} \hat{\mathbb{E}}\left[y\phi(\mathbf{x})_j'\mathbf{u}_j\right]^2 = \frac{1}{\ell^2}\sum_{j=1}^{k}\left(\mathbf{u}_j'\mathbf{X}_j'\mathbf{y}\right)^2 \tag{11}$$

and r is the number of kernel evaluations used to project a new test point.

The same test is repeated using the solution to (9), comparing the resulting features using the cumulative



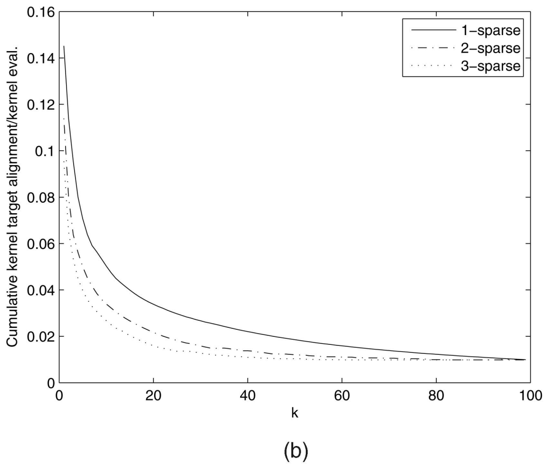


Fig. 2. Effect of using different numbers of examples per sparse direction. (a) Cumulative square covariance per kernel evaluation for p-SMC. (b) Cumulative kernel target alignment per kernel evaluation for p-SMA.

kernel target alignment per kernel evaluation. This quantity is given by $D(S)/r$, where

$$D(S) = \sum_{j=1}^{k} A\Big(\mathbf{X}_j\mathbf{u}_j\mathbf{u}_j'\mathbf{X}_j', \mathbf{y}\mathbf{y}'\Big) = \sum_{j=1}^{k} \frac{(\mathbf{y}'\mathbf{X}_j\mathbf{u}_j)^2}{\mathbf{y}'\mathbf{y}\,\mathbf{u}_j'\mathbf{X}_j'\mathbf{X}_j\mathbf{u}_j}, \quad (12)$$

which approaches 1 as k tends toward ℓ.

Fig. 2a shows the results of this test using the directions which maximize covariance. The 1-sparse method has a greater covariance per example than the 2 and 3-sparse methods, although this advantage diminishes with increased dimensionality. Similar curves are observed in Fig. 2b. Again, 1-sparse projection directions provide more alignment per example than the 2 or 3-sparse ones. Hence, in this case, 1-sparse directions are optimal in terms of capturing covariance or alignment using sparse directions. This may not be true for every case; however, any covariance or alignment lost through enforcing a high level of sparsity can be compensated for by iterating further.

4.4 Computational Efficiency

One of the benefits of the above approach to sparsity is its efficient memory usage. To compute new projection directions, the entire kernel matrix need not be present in memory. Furthermore, the computation of a projection direction requires only a single pass through the kernel matrix. However, the computational cost of both finding a projection direction and deflating the kernel matrix is $O(\ell^2)$, which becomes expensive for large ℓ. One would like to find projection directions without having to look through the entire kernel matrix, and here we show that approximate solutions for $\boldsymbol{\beta}_j$ can be found using only a subset of the kernel matrix columns. We use a small random subset of kernel matrix columns to select $\boldsymbol{\beta}_j$ and show that, in many cases, it yields a near-optimal solution.

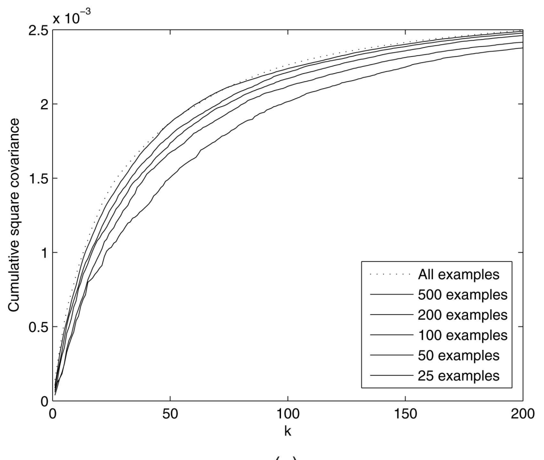
Lemma 4.1 (Maximum of Random Variables). *Denote by $\xi_1, \ldots, \xi_c$ identically distributed independent random variables with common cumulative distribution function $F(x)$. Then, the cumulative distribution function of $\xi = \max(\xi_i)$ is $G(x) = F(x)^c$.*

Let $\xi_1, \ldots, \xi_c$ represent the covariance of c columns of $\mathbf{K}_j$ with the labels. Then, $F(x)$ is the probability that each column has covariance less than or equal to x. Clearly, we have $P(\xi > x) = 1 - F(x)^c$ from the above lemma and it follows that this quantity rapidly becomes closer to 1 as c increases provided $F(x) \neq 1$. Hence, an approximation method for finding the optimal dual sparse direction is to only consider a few columns of the residual kernel matrix. A similar argument can be applied to justify using few kernel matrix columns for the computation of dual directions for SMA.

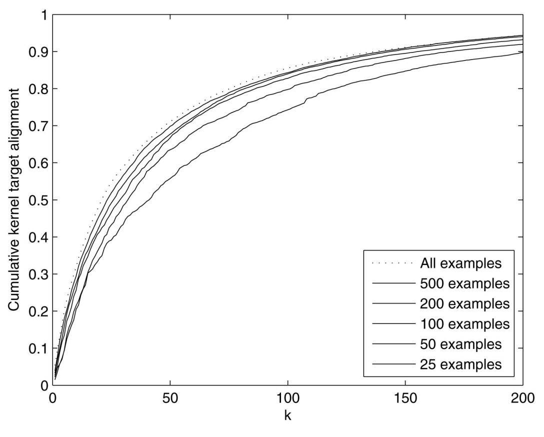
We test this strategy on a synthetic data set composed of 1,000 features and 2,000 examples, with labels computed as a linear combination of 50 features plus a small uniform noise component. The data is centered and normalized so that each feature has unit norm. SMC and SMA are run using the above strategy and sets of kernel matrix columns of different sizes. Fig. 3 shows how the cumulative square covariance of SMC and cumulative kernel target alignment of SMA vary using the above strategy. It is clear that using 500 kernel matrix columns is nearly indistinguishable from using all 2,000 in both the SMC and SMA case. With only 100 kernel matrix columns, the results are close to optimal; hence, the lemma is useful in this case. In the general case, the selection of c depends on the distribution of qualities of the kernel matrix columns. If this distribution is tail ended, for example, then one would need a larger proportion of kernel matrix columns to find one within the top few.

Using Lemma 4.1 for approximating the solutions to (5) and (6) reduces the complexity from $O(\ell^2)$ to $O(c\ell)$, where c is the number of residual kernel matrix columns selected to find $\boldsymbol{\beta}_j$. Furthermore, one need only deflate c columns of the original kernel matrix, which at the jth iteration requires $O(cj\ell + c\ell p)$ operations, where p is the cost of a kernel computation.[3] The final complexity of training both SMA and SMC is therefore $O(ck^2\ell + ck\ell p)$, which is linear in the number of examples and does not directly depend on the original dimensionality of the data. The projection of a new test example is computed in $O(kp + k^2)$ operations since $\mathbf{k}'\mathbf{B}$ requires k kernel evaluations and the resulting vector is then multiplied by the $k \times k$ matrix $((\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{K}\mathbf{B})^{-1}$. This compares favorably with the $O(\ell p + k\ell)$ operations required

3. We assume that kernel matrix elements are computed on demand; however, if they are precomputed, one can use $p = 1$, for example.



(a)



(b)

Fig. 3. Effect of using different sized random subsets of the kernel matrix columns at each iteration for the selection of β_j. Curves are shown in order of legend. (a) Cumulative square covariance of SMC. (b) Cumulative kernel target alignment of SMA.

for the projection of a test point onto k nonsparse dual directions.

The complexities of SMA and SMC are compared with several other feature extraction methods in Table 1.[4] Training sparse KPLS is efficient at $O(\nu k\ell^2 p)$ complexity, as its dual projection directions have $\nu\ell$ nonzero elements and each iteration requires the projection of the kernel matrix onto these directions. To evaluate a new test point, one must project onto a set of deflated dual directions as the original directions are computed relative to the deflated kernel matrices. These deflated directions are often nonsparse; hence, the projection of a new test point requires $O(\ell p + k\ell)$ operations.

To conclude, we note that the efficiency of training SMA and SMC arises through the particular deflation mechanism used, sparsity, and the use of randomization for selecting dual directions. Some of the methods listed in Table 1 could be made more efficient by using sparsity and randomization in a similar fashion. On the other hand, KPLS, for example, does not lend itself easily to efficiency improvements in this way

4. We assume that the data is already centered, which is an $O(m\ell)$ operation in the primal case and $O(\ell^2)$ for a kernel matrix. For BLF, the complexities of computing the loss function and gradient of the loss function are denoted by q and r in the primal and dual cases, respectively. For PLS and KPLS, we assume there is a single label per example.

TABLE 1
The Training and Test Complexities of Some Feature Extraction Algorithms

	Primal		Dual	
Algorithm	Train	Test	Train	Test
PCA	$O(m^2\ell + m^3)$	$O(mk)$	$O(\ell^2 p + \ell^3)$	$O(\ell p + k\ell)$
PLS	$O(mk\ell)$	$O(mk)$	$O(\ell^2 p + k\ell^2)$	$O(\ell p + k\ell)$
KB	N/A	N/A	$O(\ell^2 p + k\ell^3)$	$O(\ell p + k\ell)$
BLF	$O(mk\ell + kq)$	$O(mk)$	$O(\ell^2 p + k\ell^2 + kr)$	$O(\ell p + k\ell)$
Sp. KPLS	N/A	N/A	$O(\nu k\ell^2 p)$	$O(\ell p + k\ell)$
SMA/SMC	N/A	N/A	$O(ck\ell p + ck^2\ell)$	$O(kp + k^2)$

since its deflation operates on the entire kernel matrix, in contrast with deflation (3) which can be applied to kernel matrix columns independently.

5 Stability of the Framework

An important question about the general framework is how the quality of the generated subspace varies over different data sets from the same source. Such results have previously been derived for KPCA in [32] and we will show a similar result here. As PLS and the sparse algorithms extract features that are predictive toward the labels, it is intuitive to measure the covariance of the extracted features with the labels. One would like to know in what circumstances a high covariance on the training set is maintained for a test set.

The value of interest is the expectation of the cumulative square covariance of the features with respect to the labels. The cumulative square covariance of a single projected example is $f(\mathbf{x}, y^2) = \sum_{i=1}^{k} (y\phi(\mathbf{x})'\mathbf{w}_i)^2 = y^2\phi(\mathbf{x})'\mathbf{W}\mathbf{W}'\phi(\mathbf{x})$, where $\mathbf{W}$ has its columns composed of directions $\mathbf{w}_i$, $i = 1, \dots, k$, and $(\phi(\mathbf{x}), y)$ is an example-label pair. Hence, our aim is to provide a lower bound on $\mathbb{E}[f(\mathbf{x}, y^2)]$. In the case of the general framework, the projections of the training examples are given by $\mathbf{T}$ and the corresponding empirical estimate of the expectation is

$$\hat{\mathbb{E}}\left[f(\mathbf{x}, y^2)\right] = \frac{1}{\ell}\sum_{i=1}^{\ell} y_i^2 \phi(\mathbf{x}_i)'\mathbf{W}\mathbf{W}'\phi(\mathbf{x}_i) = \frac{1}{\ell}\mathrm{tr}(\mathbf{T}'\tilde{\mathbf{Y}}\mathbf{T}),$$

where $\tilde{\mathbf{Y}}$ is a diagonal matrix with diagonal entries $\tilde{\mathbf{Y}}_{ii} = y_i^2$, $i = 1, \dots, \ell$.

In order to derive a bound on the expected covariance, we apply Rademacher theory [33], which is concerned with how functions from a certain function class are able to fit random data. The path we take will start with the definition of the *Rademacher complexity* of a real-valued function class, which is a measure of its capacity. We then introduce the function class of f and its corresponding Rademacher complexity. One can bound the expectation of f using its Rademacher complexity and empirical expectation.

Definition 5.1. *For a sample $S = \{\mathbf{x}_1, \dots, \mathbf{x}_\ell\}$ generated by a distribution $\mathcal{D}$ on a set $\mathcal{X}$ and a real-valued function class $\mathcal{F}$ with domain $\mathcal{X}$, the* empirical Rademacher complexity *of $\mathcal{F}$ is the random variable*

$$\hat{R}_\ell(\mathcal{F}) = \mathbb{E}_\sigma\left[\sup_{f\in\mathcal{F}}\left|\frac{2}{\ell}\sum_{i=1}^{\ell}\sigma_i f(\mathbf{x}_i)\right|\middle|\mathbf{x}_1,\ldots,\mathbf{x}_\ell\right],$$

where $\sigma = \{\sigma_1,\ldots,\sigma_\ell\}$ are independent uniform $\{\pm1\}$-valued (Rademacher) random variables. The Rademacher complexity *of $\mathcal{F}$ is*

$$R_\ell(\mathcal{F}) = \mathbb{E}_S\left[\hat{R}_\ell(\mathcal{F})\right] = \mathbb{E}_{S\sigma}\left[\sup_{f\in\mathcal{F}}\left|\frac{2}{\ell}\sum_{i=1}^{\ell}\sigma_i f(\mathbf{x}_i)\right|\right].$$

Notice that the term inside the sup is proportional to the covariance of the Rademacher variables and f. In other words, if f can be chosen to fit random data easily in general, then the Rademacher complexity of $\mathcal{F}$ is high. It follows that there is a greater possibility of detecting a spurious pattern which will not generalize to a test set.

In this case, the class of functions of interest is linear with bounded norm

$$\left\{\mathbf{x}\mapsto\sum_{i=1}^{\ell}\alpha_i\kappa(\mathbf{x},\mathbf{x}_i) : \mathbf{x}_i\in\mathcal{X},\sum_{i,j}\alpha_i\alpha_j\kappa(\mathbf{x}_i,\mathbf{x}_j)\le B^2\right\}$$
$$\subseteq\{\mathbf{x}\mapsto\langle\phi(\mathbf{x}),\mathbf{w}\rangle : \|\mathbf{w}\|\le B\} = \mathcal{F}_B,$$

where $\mathcal{X}$ is the domain of $\mathcal{F}_B$, α_i, $i=1,\ldots,\ell$, are dual variables, and B is an upper bound on the norm of the functions in $\mathcal{F}_B$. Note that this definition does not depend on any particular training set.

We now introduce a bound on the empirical Rademacher complexity of $\mathcal{F}_B$ using the following theorem from [34].

Theorem 5.2. *If $\kappa:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ is a kernel and $S=\{\mathbf{x}_1,\ldots,\mathbf{x}_\ell\}$ is a sample of points from $\mathcal{X}$, then the empirical Rademacher complexity of the class $\mathcal{F}_B$ satisfies*

$$\hat{R}_\ell(\mathcal{F}_B)\le\frac{2B}{\ell}\sqrt{\sum_{i=1}^{\ell}\kappa(\mathbf{x}_i,\mathbf{x}_i)} = \frac{2B}{\ell}\sqrt{\mathrm{tr}(\mathbf{K})}.$$

It is intuitive that the Rademacher complexity of $\mathcal{F}_B$ depends on B. Less so is its dependence on the trace of the kernel matrix. However, the trace of $\mathbf{K}$ is the sum of its eigenvalues and also the cumulative variance of the examples. It follows that, since w is chosen from the space of the training examples, the corresponding Rademacher complexity is dependent on this quantity.

The last main ingredient for our bound is a theorem which lower bounds the expectation of a function in terms of its empirical expectation and the Rademacher complexity of its function class. This results from a small modification of a theorem from [18] which provides an upper bound to the expectation.

Theorem 5.3. *Fix $\delta\in(0,1)$ and let $\mathcal{F}$ be a class of functions mapping from Z to $[0,1]$. Let $(\mathbf{z}_i)_{i=1}^{\ell}$ be drawn independently according to a probability distribution $\mathcal{D}$. Then, with probability at least $1-\delta$ over random draws of samples of size ℓ, every $f\in\mathcal{F}$ satisfies*

$$\mathbb{E}_\mathcal{D}[f(\mathbf{z})]\ge\hat{\mathbb{E}}[f(\mathbf{z})]-R_\ell(\mathcal{F})-\sqrt{\frac{\ln(2/\delta)}{2\ell}}$$
$$\ge\hat{\mathbb{E}}[f(\mathbf{z})]-\hat{R}_\ell(\mathcal{F})-3\sqrt{\frac{\ln(2/\delta)}{2\ell}}.$$

To tie up loose ends, we introduce Theorem 5.4, which provides some useful properties of Rademacher complexity.

Theorem 5.4. *Let $\mathcal{F}$ and $\mathcal{G}$ be classes of real functions. Then,*

1. *if $\mathcal{F}\subseteq\mathcal{G}$, then $\hat{R}_\ell(\mathcal{F})\le\hat{R}_\ell(\mathcal{G})$, and*
2. *for every $c\in\mathbb{R}$, $\hat{R}_\ell(c\mathcal{F}) = |c|\hat{R}_\ell(\mathcal{F})$.*

The previous definitions and theorems are now used to derive a theorem which lower bounds the expectation of our original function $f(\mathbf{x},y^2)$.

Theorem 5.5. *Let $f(\mathbf{x},y^2)=y^2\mathbf{k}'\mathbf{A}\mathbf{A}'\mathbf{k}$ be formulated by performing general feature extraction on a randomly drawn training set S of size ℓ in the feature space defined by a kernel $\kappa(\mathbf{x},\mathbf{z})$ and projecting new data using the dual projection matrix $\mathbf{A}=\mathbf{B}((\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{K}\mathbf{B})^{-1}$. Fix c and let $f(\mathbf{x},y^2)$ belong to a class of linear functions $\mathcal{F}_c$ with norm bounded by c. With probability greater than $1-\delta$ over the generation of the sample S, the expected value of f is bounded by*

$$\mathbb{E}_D\left[f(\mathbf{x},y^2)\right]\ge\frac{1}{\ell}\mathrm{tr}(\mathbf{T}'\tilde{\mathbf{Y}}\mathbf{T})-\frac{2c}{\ell}\sqrt{\mathrm{tr}(\hat{\mathbf{K}})}-3cR^2\sqrt{\frac{\ln(2/\delta)}{2\ell}},\tag{13}$$

where $\hat{\mathbf{K}}$ is the kernel matrix defined by $\hat{\kappa}(\mathbf{x},\mathbf{z})=y(\mathbf{x})^2y(\mathbf{z})^2\langle\phi(\mathbf{x}),\phi(\mathbf{z})\rangle^2$, $y(\mathbf{x})$ is the label corresponding to $\mathbf{x}$, and R is radius of the hypersphere enclosing the examples $y\phi(\mathbf{x})$.

Proof. First, consider the following:

$$\begin{aligned}f(\mathbf{x},y^2)&=y^2\phi(x)'\mathbf{W}\mathbf{W}'\phi(\mathbf{x})\\&=y^2\sum_{i,j}\phi(\mathbf{x})_i\phi(\mathbf{x})_j(\mathbf{W}\mathbf{W}')_{ij}\\&=y^2\langle\tilde{\phi}(\mathbf{x}),\tilde{\mathbf{W}}\rangle_F,\end{aligned}$$

where $\tilde{\phi}(\mathbf{x})_{ij}=\phi(\mathbf{x})_i\phi(\mathbf{x})_j$ and $\tilde{\mathbf{W}}_{ij}=(\mathbf{W}\mathbf{W}')_{ij}$. Hence, f can be considered as a linear function of its inputs with norm bounded by $c\ge\|\tilde{\mathbf{W}}\|_F$, provided $\mathbf{x}$ is mapped to the feature space defined by $\tilde{\phi}$. The kernel function corresponding to $\tilde{\phi}$ is

$$\begin{aligned}\langle\tilde{\phi}(\mathbf{x}),\tilde{\phi}(\mathbf{z})\rangle_F&=\sum_{i,j}\phi(\mathbf{x})_i\phi(\mathbf{x})_j\phi(\mathbf{z})_i\phi(\mathbf{z})_j\\&=\sum_i\phi(\mathbf{x})_i\phi(\mathbf{z})_i\sum_j\phi(\mathbf{x})_j\phi(\mathbf{z})_j\\&=\kappa(\mathbf{x},\mathbf{z})^2.\end{aligned}$$

An application of Theorem 5.2 provides a bound on the empirical Rademacher complexity of $\mathcal{F}_c$

$$\hat{R}_\ell(\mathcal{F}_c)\le\frac{2c}{\ell}\sqrt{\sum_i^{\ell}y_i^4\kappa(\mathbf{x}_i,\mathbf{x}_i)^2}.$$

Define $h(\mathbf{x},y^2)=f(\mathbf{x},y^2)/cR^2$, which belongs to a class of linear functions $\mathcal{H}$ with bounded norm. Using Theorem 5.3,

$$\mathbb{E}_\mathcal{D}\left[h(\mathbf{x},y^2)\right]\ge\hat{\mathbb{E}}\left[h(\mathbf{x},y^2)\right]-\hat{R}_\ell(\mathcal{H})-3\sqrt{\frac{\log(2/\delta)}{2\ell}}.\tag{14}$$

The Rademacher complexity of $\mathcal{H}$ is

$$\hat{R}_\ell(\mathcal{H}) = \frac{1}{cR^2}\hat{R}_\ell(\mathcal{F}) \leq \frac{2}{\ell R^2}\sqrt{\sum_{i=1}^{\ell} y_i^4 \kappa(\mathbf{x}_i, \mathbf{x}_i)^2},$$

which follows from an application of part 2 of Theorem 5.4. Substituting into (14) and multiplying by cR^2 gives

$$\mathbb{E}_{\mathcal{D}}\big[f(\mathbf{x}, y^2)\big] \geq \hat{\mathbb{E}}\big[f(\mathbf{x}, y^2)\big] - \frac{2c}{\ell}\sqrt{\mathrm{tr}(\hat{\mathbf{K}})} - 3cR^2\sqrt{\frac{\ln(2/\delta)}{2\ell}},$$

and then making a substitution $\hat{\mathbb{E}}[f(\mathbf{x}, y^2)] = \frac{1}{\ell}\mathrm{tr}(\mathbf{T}'\tilde{\mathbf{Y}}\mathbf{T})$ produces the required result. □

This theorem indicates that the expected cumulative square covariance of the features produced under the general framework will be close to its empirical estimate provided the Rademacher and final terms are proportionately small. As we are working in kernel-defined feature spaces, the original dimensionality is unimportant and the final two terms grow inversely proportional to the root of the number of examples. For the middle term, the trace of $\hat{\mathbf{K}}$ can be understood as the cumulative variance of the examples given by $S' = \{\tilde{\phi}(\mathbf{x}_1)y_1^2, \ldots, \tilde{\phi}(\mathbf{x}_\ell)y_\ell^2\}$. Hence, to detect stable patterns and allow a large covariance to be captured, one requires a rapid decay in the eigenvalues of $\hat{\mathbf{K}}$ and also a small value of $1/\sqrt{\ell}$.

To illustrate the effectiveness of Theorem 5.5, we consider a function $g(\mathbf{x}, y^2) = f(\mathbf{x}, y^2) - qk$ which introduces a cost q on the number of features k. This cost might represent extra computation or memory requirements and indicates a preference for low-dimensional subspaces. We use two DELVE data sets for this evaluation. The first is one of the Pumadyn data sets composed of 8,192 examples, 32 features, classified as "fairly linear," and with "high noise." The second is called "bank-32nm," from the bank family of data sets, and also has 8,192 examples and 32 features. The original features are normalized, the value of c for Theorem 5.5 is estimated from the data, and we set $\delta = 0.1$. As a simple heuristic to select the value of q, we use half the gradient of $\hat{\mathbb{E}}[f(\mathbf{x}, y^2)]$ at the first iteration. Using two thirds of the examples for training and the remaining for testing, we observe how the lower bound of $\mathbb{E}_D[g(\mathbf{x}, y^2)]$ varies with the number of iterations of SMC with $c = 500$. This bound is compared with the empirical expectation on the test examples.

Fig. 4 shows the results of this test and, with the Pumadyn data set, the bound of $\mathbb{E}_D[g(\mathbf{x}, y^2)]$ and empirical expectation of $g(\mathbf{x}, y^2)$ on the test set are close together. Although the peaks of these curves are dependent on the manner in which q is selected, one could potentially use the bound as a method of selecting k in this case. Another possible use of Theorem 5.5 is for selecting the number of examples required for the lower bound of $\mathbb{E}_D[g(\mathbf{x}, y^2)]$ to be close to its empirical estimate. For these applications, Theorem 5.5 is most useful when the final two terms of (13) are small relative to the empirical expectation. One can then say, with high probability, that the empirical expectation of the covariance of the features on a test set is close to the lower bound of $\mathbb{E}_D[g(\mathbf{x}, y^2)]$.

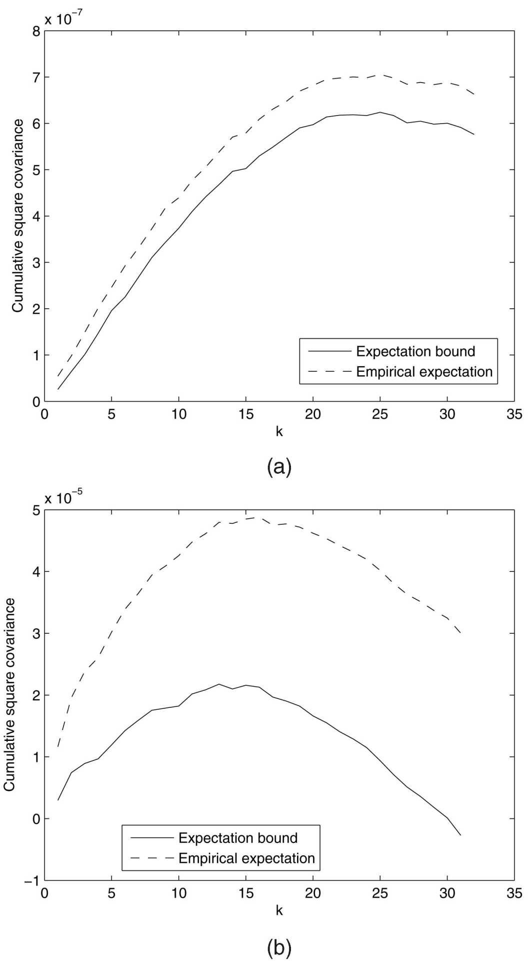


Fig. 4. Plot of the lower bound of $\mathbb{E}_D[g(\mathbf{x}, y^2)]$ and empirical expectation of $g(\mathbf{x}, y^2)$ on a test set. (a) Pumadyn data set. (b) Bank data set.

With the bank data set, the bound is less close to the empirical expectation of $g(\mathbf{x}, y^2)$ on the test set. In this case, a large value of R causes the final term of (13) to be significant in comparison the empirical expectation of $f(\mathbf{x}, y^2)$. However, with this data set, most of the examples from S' lie in a hypersphere of small radius. Hence, to improve the statistical stability of the extracted features, one could remove the examples in S' with large norm, which can be considered as outliers.

6 Computational Results

SMA and SMC are supervised methods and here we compare the predictive performance of their features with several other feature extraction algorithms. This comparison is performed first on a few Bilkent University function approximation [35] and UCI [36] data sets and then on a large sample of the Reuters Corpus Volume 1 data set [37]. We complete this section by applying SMA and SMC to an example face detection application, an area in which PCA has traditionally been a popular choice.

6.1 Regression Experiment

This first experiment compares the regression performance of the features extracted by SMC and SMA to those generated by KPLS, sparse KPLS, and rKOPLS. We use

TABLE 2
Information about the UCI (Top) and Bilkent University Function Approximation (Bottom) Data Sets

Dataset	Examples	Features
Ionosphere	355	34
Sonar	208	60
SPECTF	267	44
WDBC	569	30
Ailerons	7154	40
Baseball	337	16
Pole Telecomm	9065	48

the data sets shown at the bottom of Table 2 and each one has its examples and labels centered and normalized to have zero mean and unit norm. With large data sets, only the first 1,000 examples are used so that the tests can be run within a reasonable time frame. All experimental code is implemented in Matlab.

We evaluate the methods by following feature extraction with least squares regression, which in a kernel-defined feature space finds the minimum of $\|\mathbf{K}\mathbf{c} - \mathbf{y}\|^2$. The solution to this optimization is $\mathbf{c} = \mathbf{K}^{-1}\mathbf{y}$, assuming $\mathbf{K}$ is full rank, which with the linear kernel is identical to OLS regression. After performing regression, the root mean squared error is recorded, given by $\|f(\mathbf{X}) - \mathbf{y}\|/\sqrt{\ell}$, where $f(\mathbf{X})$ is a vector of predicted labels of length ℓ.

In order to reduce bias caused by sample selection, the error is measured using fivefold cross validation (CV). The selection of parameters for each feature extraction method is performed at each fold using an inner CV loop. For each unique combination of parameters, the error is recorded using fivefold CV and the parameters resulting in the lowest error are output at this stage.

The parameters for the feature extraction methods are selected using the following values: The number of extracted features is chosen from 1 to the rank of the data. The sparse KPLS sparsity parameter ν is selected from $\{0.125, 0.25, 0.5, 1.0\}$ and the heuristic used to compute dual projection directions is selected as either the Maximal Information (MI) criterion with a kernel cache size of 500 or the Maximum Residual (MR) criterion. For rKOPLS, the r parameter is chosen from $\{50, 100, 200, 400\}$. SMA and SMC are run using 500 kernel matrix columns for the selection of each dual projection direction. To test feature extraction in a dual space, each method is also run using the Radial Basis Function (RBF) kernel, defined by $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma^2)$, with kernel width σ selected from 0.125 to 16 in exponentially increasing steps.

The results, shown in Table 3, indicate that feature extraction often improves over the use of the least squares method for these data sets. Most notably, KPLS often results in the lowest error in both the linear and RBF spaces, although SMA and SMC are only slightly worse in general. We observed that the number of features chosen for KPLS through the CV procedure is on average less than the corresponding number used for SMA and SMC. One might expect this to be the case since, in contrast to PLS, SMA and SMC have strict sparsity constraints. The rKOPLS method produces a single feature based on the projection onto a linear combination of a random subset of the examples. This did not generalize well as rKOPLS results in the highest errors in most cases.

TABLE 3
Errors and Standard Deviations (in Parentheses) of Feature Extraction Followed by Least Squares Regression, with Best Results for Primal and RBF Methods in Bold

Method	Ailerons	Baseball	Pole Telecomm
All features	0.0168 (0.0021)	0.0329 (0.0043)	0.0232 (0.0007)
PLS	0.0160 (0.0005)	**0.0321** (0.0026)	**0.0231** (0.0005)
SMA	**0.0158** (0.0008)	0.0327 (0.0038)	0.0233 (0.0004)
SMC	0.0159 (0.0007)	0.0323 (0.0026)	**0.0231** (0.0005)
Sp. KPLS	0.0160 (0.0015)	0.0328 (0.0050)	0.0232 (0.0006)
rKOPLS	0.0315 (0.0035)	0.0453 (0.0122)	0.0316 (0.0010)
RBF Features	**0.0156** (0.0011)	0.0344 (0.0063)	0.0196 (0.0011)
RBF PLS	0.0158 (0.0012)	**0.0330** (0.0041)	**0.0094** (0.0014)
RBF SMA	0.0157 (0.0004)	0.0342 (0.0059)	0.0157 (0.0026)
RBF SMC	0.0162 (0.0019)	0.0343 (0.0059)	0.0124 (0.0014)
Sp. RBF PLS	0.0161 (0.0015)	0.0353 (0.0036)	0.0117 (0.0005)
RBF rKOPLS	0.0170 (0.0020)	0.0371 (0.0057)	0.0114 (0.0020)

6.2 UCI Classification Experiment

We now study another common scenario, which is when feature extraction is followed by classification. We apply k-Nearest Neighbor (KNN) and SVM classification and use a selection of UCI data sets (listed at the top of Table 2). The examples in these data sets are preprocessed in an identical manner to that used in the previous experiment. SMA and SMC are compared to KPLS,[5] sparse KPLS, KB, and KBLF.[6] The LIBSVM package [38] is used for computing SVM models.

For most of these tests, we use fivefold CV repeated three times (with random permutations of the data) and fivefold CV for model selection. KB, Least Absolute Deviation (LAD) loss BLF, and the RBF feature extraction algorithms are considerably slower than the other methods; hence, they are evaluated using fivefold CV repeated two times with threefold CV for model selection.

The parameters for the feature extraction methods are selected using the same values as the ones used in the previous experiment. KB, however, could iterate further than the rank of the data and was additionally allowed up to 1,000 iterations in steps of 100. The SVM penalty parameter is selected from 0.125 to 128 in exponentially increasing steps and the number of neighbors k for the KNN method is selected from $\{1, 3, 5, 7, 9\}$. As a useful comparison to the RBF feature extraction methods, we also apply the SVM and KNN classifiers to the original data using the RBF kernel, with σ selected from 0.125 to 16 in exponentially increasing steps.

Tables 4 and 5 summarize the results of this experiment, and we first discuss the primal results. A broad trend is that feature extraction results in larger improvements in the error with the KNN method than with the SVM. This can be explained by the good generalization implied by maximizing the margin for the SVM [39], whereas KNN is more sensitive to noise. As with the regression results, SMC and SMA have comparable errors to many of the other feature extraction methods despite using only a single example for each projection vector. They are particularly effective on the Ionosphere data set, improving the error obtained with the KNN method from 0.140 to 0.106 and 0.105, respectively,

5. It may seem unusual to use regressive PLS in a classification task in light of the discriminative PLS method; however, in the single label case, the methods are identical.

6. Sparse KBLF is not included in the comparison since [28] shows that it performs worse than an SVM.

TABLE 4
Errors and Standard Deviations (in Parentheses) of the Extracted Features with the KNN Algorithm

	Ionosphere	Sonar	SPECTF	WDBC
All features	0.140 (0.02)	**0.150** (0.05)	0.244 (0.04)	0.034 (0.01)
PLS	0.110 (0.02)	0.179 (0.07)	0.230 (0.05)	0.032 (0.01)
SMA	0.106 (0.04)	0.215 (0.07)	0.253 (0.05)	0.047 (0.02)
SMC	0.105 (0.03)	0.203 (0.05)	0.235 (0.04)	0.045 (0.02)
Exp BLF	0.096 (0.03)	0.218 (0.07)	**0.206** (0.04)	0.049 (0.01)
Log BLF	0.087 (0.03)	0.215 (0.06)	**0.206** (0.06)	0.044 (0.02)
LAD BLF	0.099 (0.04)	0.241 (0.07)	0.219 (0.04)	0.043 (0.01)
Exp KB	0.134 (0.04)	0.237 (0.05)	0.238 (0.05)	0.046 (0.01)
Log KB	0.110 (0.03)	0.241 (0.07)	0.223 (0.04)	**0.027** (0.02)
Sp. KPLS	**0.084** (0.03)	0.180 (0.06)	0.236 (0.04)	0.044 (0.02)
RBF Features	0.141 (0.04)	0.133 (0.06)	0.235 (0.03)	0.033 (0.01)
RBF PLS	**0.050** (0.02)	**0.107** (0.03)	0.223 (0.04)	**0.029** (0.01)
RBF SMA	0.053 (0.02)	0.168 (0.05)	**0.215** (0.07)	0.036 (0.02)
RBF SMC	0.057 (0.03)	0.173 (0.06)	0.243 (0.03)	0.052 (0.02)
Sp. RBF PLS	0.073 (0.04)	0.195 (0.07)	0.260 (0.04)	0.044 (0.02)

TABLE 5
Errors and Standard Deviations (in Parentheses) of the Extracted Features with the SVM Algorithm

	Ionosphere	Sonar	SPECTF	WDBC
All features	0.134 (0.03)	0.231 (0.08)	0.205 (0.03)	0.025 (0.01)
PLS	0.132 (0.03)	0.224 (0.07)	0.201 (0.04)	0.028 (0.01)
SMA	0.133 (0.04)	0.224 (0.08)	0.230 (0.05)	0.028 (0.01)
SMC	**0.123** (0.03)	0.231 (0.07)	0.213 (0.03)	0.034 (0.02)
Exp BLF	0.138 (0.03)	0.234 (0.06)	0.224 (0.06)	0.040 (0.01)
Log BLF	0.133 (0.03)	0.228 (0.06)	0.223 (0.05)	0.032 (0.01)
LAD BLF	0.124 (0.04)	0.239 (0.05)	0.228 (0.04)	0.031 (0.01)
Exp KB	0.131 (0.03)	**0.205** (0.03)	**0.200** (0.04)	0.033 (0.01)
Log KB	0.129 (0.03)	0.217 (0.05)	0.206 (0.03)	0.023 (0.01)
Sp. KPLS	0.136 (0.04)	0.246 (0.06)	0.209 (0.04)	**0.021** (0.01)
RBF Features	**0.056** (0.03)	0.145 (0.07)	0.200 (0.03)	0.031 (0.01)
RBF PLS	0.069 (0.02)	**0.122** (0.03)	0.219 (0.03)	**0.027** (0.01)
RBF SMA	0.057 (0.03)	0.146 (0.06)	**0.194** (0.03)	0.030 (0.02)
RBF SMC	0.057 (0.03)	0.141 (0.04)	0.202 (0.03)	0.031 (0.01)
Sp. RBF PLS	**0.056** (0.03)	0.137 (0.09)	0.291 (0.10)	**0.027** (0.01)

and SMC provides the lowest error with the linear SVM. Comparing SMC to PLS shows that the addition of the sparsity constraint does not have a large impact on the error. A possible explanation is that the distribution of examples in these data sets allows a single example to result in a covariance close to that obtained using a linear combination of the examples. Notice that the features produced by KB often lead to low error rates; however, this is frequently at the expense of a dimensionality higher than that of the original data.

When considering the RBF features spaces, the low errors obtained on the Ionosphere and Sonar data sets with both the KNN and SVM imply a nonlinear relationship between the features and labels. The RBF PLS approach improves over using the plain RBF features with the KNN and also for the Sonar and WDBC data sets with the SVM. KPLS also frequently has the lowest error rate when compared to the other RBF feature extraction methods. Again, in most cases, SMA and SMC are only slightly worse than RBF PLS, and also comparable to sparse KPLS.

6.3 Text Retrieval

We demonstrate the scalability of the sparse algorithms by running them on the Reuters Corpus Volume 1 news database. The full database consists of about 800,000 news articles (the period of a whole year), but only 20,000 examples are considered from the first three months of the Economics branch. As a preprocessing step, the Porter stemmer has been used to reduce all words to their stems which yields 136,469 distinct terms. Labels are assigned according to whether the articles are about the topic "Government Finance," with approximately 37 percent of the examples positively labeled. Features are extracted on this data set by sparse KPLS, SMC, and SMA and then used to train a linear SVM. In this case, we do not generate results using PLS, KB, and BLF since the computational and memory requirements needed to run these methods are prohibitively high.

TABLE 6
Results on the Reuters Data Set

	Average Precision	Projections	Sparsity	SVs
SVM	**0.847** ± 0.0096	-	-	8864
SMA	**0.848** ± 0.0068	366	366	9964
SMC	0.823 ± 0.009	400	400	9953
Sparse KPLS	0.776 ± 0.032	400	13334	6690

Sparsity is the number of kernel evaluations required for the projection of a new test example and SV is the number of support vectors used.

The process used to conduct this experiment is similar to that of the previous one. The data is not centered since the examples are sparse and centering in general removes sparsity. The tests are run using a single repetition of threefold CV with an inner threefold CV loop, used on 2,000 randomly sampled training examples, for model selection. For each method, we record the average precision since it is a standard measure in information retrieval. It is defined as the cumulative precision after each relevant document is retrieved divided by the number of relevant documents, where precision is the proportion of relevant documents to all the documents retrieved. Average precision emphasizes returning more relevant documents high up in the ranking.

At the model selection stage, the parameters are chosen as follows: Each feature extraction algorithm is run from 100 to 400 iterations in steps of 100. Sparse KPLS is run using both the MI heuristic with a kernel cache size of 300 and the MR heuristic, with ν chosen from $\{0.2, 0.4, 0.8\}$. SMA and SMC are run using 500 kernel columns for the selection of β_j. The SVM is trained using a linear kernel and its penalty parameter is chosen from 0.125 to 128 in exponentially increasing values. Since the class labels are imbalanced, LIBSVM is parameterized so as to weight different classes with different values. The weight for each class is fixed as the percentage of examples of the opposite class.

Table 6 shows that SMC and SMA outperform sparse KPLS, both in terms of average precision and the number of kernel evaluations required to project a new example. Recall that, for SMA and SMC, the latter quantity is simply the number of iterations since the projection of a new test example requires only the selected training examples and the precomputed matrix $((\mathbf{T}'\mathbf{T})^{-1}\mathbf{T}'\mathbf{KB})^{-1}$. Notice that SMA achieves a similar average precision to the raw SVM using a much smaller dimensionality. We might hope for an improvement over the SVM results; however, [40] shows that few features in text categorization data sets are irrelevant. With sparse KPLS, we believe in this case that the heuristics used for selecting the nonzero elements in the dual directions were ineffective which resulted in model selection making a preference for nonsparse directions.

The number of support vectors may appear to be useful in computing the total number of kernel evaluations

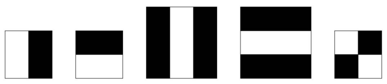


Fig. 5. Illustration of the different kinds of features proposed by Viola and Jones. Each feature is the sum of the pixel values within the white regions subtracted from the sum of the pixel values within the black regions.

required for the classification of a new example. However, once a new example is projected into the sparse subspace, one can work in the primal space and the number of support vectors is no longer relevant. In this case, an efficient primal space algorithm can be used, for example, the one described in [41]. This has a complexity of $O(s\ell)$, where s is the average number of nonzero features per example. When applied to sparse data, the overall complexity of SMA or SMC followed by this SVM classifier is $O(ck^2\ell + ck\ell p)$, i.e., linear in the number of examples.

6.4 Face Detection

Facial recognition has seen a lot of attention in recent years [42], with applications such as Human Computer Interaction (HCI), law enforcement, and security. An important stage in any face recognition system is face detection, i.e., answering the question about whether a particular image contains a face. If a raw image is used as the input to a pattern recognition algorithm, the number of features is often large and feature extraction is a useful process. PCA and KPCA have enjoyed success in face recognition since, in face data sets, most of the variation can be captured in relatively few projection directions [43]. In this final experiment, we apply KPCA, SMA, and SMC followed by SVM classification to the MIT CBCL Face Data Set 1 [44]. The aim is to observe how effectively the examples are ranked in each subspace using a Receiver Operating Characteristic (ROC) curve.

The MIT CBCL face data set consists of a training set composed of 2,429 faces and 4,548 nonfaces and a test set with 472 faces and 23,573 nonfaces. The test set is particularly challenging since it contains those nonface images from the CMU Test Set 1 [45] which are most similar to faces. All images are captured in grayscale at a resolution of 19×19 pixels, but rather than use pixel values as features, we use those proposed by Viola and Jones in [46] since they give excellent performance in conjunction with their face detection system. Viola and Jones describe three kinds of features computed for each image. A two-rectangle feature is the difference between the sum of pixels within two adjacent rectangular regions of same size and shape. A three-rectangle feature computes the sum of two outside rectangles subtracted from the sum of a center rectangle, and a four-rectangle feature is the difference between diagonal pairs of rectangles. These features, illustrated in Fig. 5, are computed for each image over many scales yielding 30,798 features per image.

The following procedure is used for this experiment. A reduced training set is formed using 3,000 examples sampled from the original training set. The Viola and Jones features are used in the RBF feature space with $\sigma = 0.25$, as preliminary tests on the training set showed that this gave good performance with an SVM. Model selection is performed using threefold CV repeated twice using a sample of 1,500 examples from the training set. The parameters which result in the highest value for the Area Under the ROC Curve (AUC)

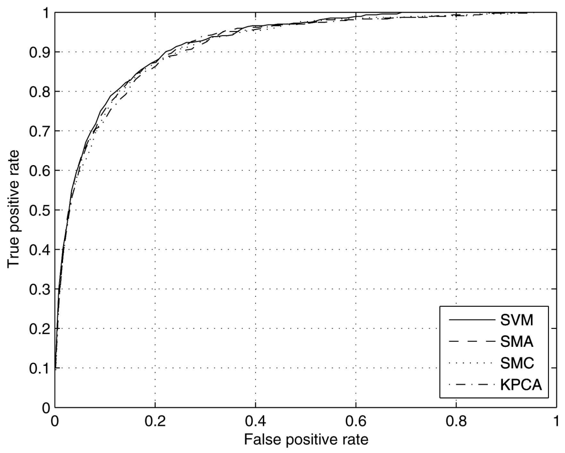


Fig. 6. ROC curves for the MIT CBCL Face Data Set 1.

measure are selected at this stage. KPCA, SMA, and SMC are iterated from 100 to 1,000 times in steps of 100 and the SVM penalty parameter is varied from 0.25 to 64. SMA and SMC are run using 500 kernel matrix columns for the selection of the dual projection directions.

To evaluate the learned models, a ROC curve is recorded for the predictions made on the test set. This curve shows the quality of the ranking of examples using different classifier thresholds. It plots the false positive rate against the true positive rate, where the false positive rate is the fraction of negative examples that are incorrectly predicted as positive and the true positive rate is the fraction of positive examples that are correctly classified.

Fig. 6 shows the resulting ROC curves for the SVM, KPCA, SMA, and SMC. There is little to differentiate the curves and all cover approximately 90 percent of the true positives with a false positive rate of 0.22 to 0.25. This is an improvement over the results in [47], which use the full training set. While this may appear to indicate that there is no advantage in any of the methods, the number of projections required for KPCA, SMA, and SMC are 1,000, 700, and 900, respectively. Clearly, SMA and SMC improve over KPCA as they require fewer features to match the performance of the SVM. Moreover, the number of kernel evaluations for the classification of a new test example using the SVM is 1,022, compared to 700 and 900 using SMA and SMC, respectively.

7 Conclusions

This paper has presented a general framework for feature extraction based on the PLS deflation. Within this framework, one chooses a projection direction, using a user-defined criterion, and then deflates. The framework, while not entirely novel, brings together a number of existing results and supplies additional insights into several feature extraction methods.

By choosing projection directions that are scalar multiples of a single residual example, one can easily derive new sparse feature extraction algorithms. Under the framework, this approach was adopted to formulate SMA and SMC, which are based on maximizing covariance and kernel target alignment, respectively. The resulting algorithms are simple to implement, and can be trained in linear time in the number of examples. Furthermore, the projection of a new test example requires only k kernel evaluations, where k is the output dimensionality.

Empirical results on a sample of real-world data sets show the features produced by SMA and SMC compare well with other feature extraction methods in both regression and classification scenarios. Using 20,000 examples from the Reuters Corpus Volume 1 data set, SMA was shown to match the performance of the original 136,469 features in conjunction with an SVM. Furthermore, on an example face detection problem, SMA and SMC require fewer features than KPCA to equal the performance of an SVM across a range of thresholds.

ACKNOWLEDGMENTS

The authors would like to thank Michinari Momma and Koby Crammer for providing code for their algorithms.

REFERENCES

[1] B.E. Boser, I.M. Guyon, and V.N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," *Proc. Fifth Ann. Conf. Computational Learning Theory,* pp. 144-152, 1992.

[2] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik, "Feature Selection for SVMs," *Advances in Neural Information Processing Systems 13,* T.K. Leen, T.G. Dietterich, and V. Tresp, eds., pp. 668-674, MIT Press, 2000.

[3] J. Bi, K.P. Bennett, M.J. Embrechts, C.M. Breneman, and M. Song, "Dimensionality Reduction via Sparse Support Vector Machines," *J. Machine Learning Research,* vol. 3, pp. 1229-1243, 2003.

[4] J.H. Friedman and J.W. Tukey, "A Projection Pursuit Algorithm for Exploratory Data Analysis," *IEEE Trans. Computers,* vol. 23, no. 9, pp. 881-889, Sept. 1974.

[5] P.J. Huber, "Projection Pursuit," *The Annals of Statistics,* vol. 13, no. 2, pp. 435-475, 1985.

[6] A.J. Smola, O.L. Mangasarian, and B. Scholkopf, "Sparse Kernel Feature Analysis," technical report, Data Mining Inst., Univ. of Wisconsin-Madison, 1999.

[7] H. Hotelling, "Analysis of a Complex of Statistical Variables into Principle Components," *J. Educational Psychology,* vol. 24, pp. 417-441 and 498-520, 1933.

[8] H. Wold, "Estimation of Principal Components and Related Models by Iterative Least Squares," *Multivariate Analysis,* pp. 391-420, 1966.

[9] M. Momma and K.P. Bennett, "Constructing Orthogonal Latent Features for Arbitrary Loss," *Feature Extraction, Foundations and Applications,* I.M. Guyon, S.R. Gunn, M. Nikravesh, and L. Zadeh, eds., pp. 551-583, 2005.

[10] N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J.S. Kandola, "On Kernel-Target Alignment," *Advances in Neural Information Processing Systems 14,* T.G. Dietterich, S. Becker, and Z. Ghahramani, eds., pp. 367-373, 2001.

[11] C. Dhanjal, S.R. Gunn, and J. Shawe-Taylor, "Sparse Feature Extraction Using Generalised Partial Least Squares," *Proc. IEEE Int'l Workshop Machine Learning for Signal Processing,* pp. 27-32, 2006.

[12] W. Massy, "Principal Components Regression in Exploratory Statistical Research," *J. Am. Statistical Assoc.,* vol. 60, no. 309, pp. 234-256, 1965.

[13] B. Schölkopf, A.J. Smola, and K.-R. Müller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation,* vol. 10, no. 5, pp. 1299-1319, 1998.

[14] A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G.R.G. Lanckriet, "A Direct Formulation for Sparse PCA Using Semidefinite Programming," Technical Report UCB/CSD-04-1330, Electrical Eng. and Computer Science Dept., Univ. of California, Berkeley, June 2004.

[15] B. Moghaddam, Y. Weiss, and S. Avidan, "Spectral Bounds for Sparse PCA: Exact and Greedy Algorithms," *Advances in Neural Information Processing Systems 18,* Y. Weiss, B. Schölkopf, and J. Platt, eds., pp. 915-922, 2006.

[16] B. Moghaddam, Y. Weiss, and S. Avidan, "Generalized Spectral Bounds for Sparse LDA," *Proc. 23rd Int'l Conf. Machine Learning,* W.W. Cohen and A. Moore, eds., pp. 641-648, 2006.

[17] M. Barker and W. Rayens, "Partial Least Squares for Discrimination," *J. Chemometrics,* vol. 17, no. 3, pp. 166-173, 2003.

[18] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis.* Cambridge Univ. Press, 2004.

[19] R. Rosipal and N. Kramer, "Overview and Recent Advances in Partial Least Squares," *Subspace, Latent Structure and Feature Selection Techniques,* pp. 34-51, 2006.

[20] R. Manne, "Analysis of Two Partial-Least-Squares Algorithms for Multivariate Calibration," *Chemometrics and Intelligent Laboratory Systems,* vol. 2, no. 1, pp. 187-197, 1987.

[21] M. Stone and R.J. Brooks, "Continuum Regression: Cross-Validated Sequentially Constructed Prediction Embracing Ordinary Least Squares, Partial Least Squares and Principal Components Regression," *J. Royal Statistical Soc. Series B (Methodological),* vol. 52, no. 2, pp. 237-269, 1990.

[22] R. Rosipal and L.J. Trejo, "Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space," *J. Machine Learning Research,* vol. 2, pp. 97-123, 2001.

[23] R. Rosipal, L.J. Trejo, and B. Matthews, "Kernel PLS-SVC for Linear and Nonlinear Classification," *Proc. 12th Int'l Conf. Machine Learning,* A. Prieditis and S.J. Russell, eds., pp. 640-647, 2003.

[24] Y. Freund and R. Schapire, "A Short Introduction to Boosting," *J. Japanese Soc. for Artificial Intelligence,* vol. 14, no. 5, pp. 771-780, Sept. 1999.

[25] K. Crammer, J. Keshet, and Y. Singer, "Kernel Design Using Boosting," *Advances in Neural Information Processing Systems 15,* S. Becker, S. Thrun, and K. Obermayer, eds., pp. 537-544, 2002.

[26] M. Momma and K.P. Bennett, "Sparse Kernel Partial Least Squares Regression," *Proc. 16th Ann. Conf. Computational Learning Theory,* B. Schölkopf and M.K. Warmuth, eds., pp. 216-230, 2003.

[27] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support Vector Regression Machines," *Advances in Neural Information Processing Systems 9,* M. Mozer, M.I. Jordan, and T. Petsche, eds., pp. 155-161, 1997.

[28] M. Momma, "Efficient Computations via Scalable Sparse Kernel Partial Least Squares and Boosted Latent Features," *Proc. 11th ACM Int'l Conf. Knowledge Discovery in Data Mining,* pp. 654-659, 2005.

[29] J. Arenas-García, K.B. Petersen, and L.K. Hansen, "Sparse Kernel Orthonormalized PLS for Feature Extraction in Large Data Sets," *Advances in Neural Information Processing Systems 19,* B. Schölkopf, J.C. Platt, and T. Hoffman, eds., pp. 33-40, 2006.

[30] K.J. Worsley, J.B. Poline, K.J. Friston, and A.C. Evans, "Characterizing the Response of PET and fMRI Data Using Multivariate Linear Models," *Neuroimage,* vol. 6, no. 4, pp. 305-319, 1997.

[31] G. Strang, *Introduction to Linear Algebra,* third ed. Wellesley Cambridge Press, 2003.

[32] J. Shawe-Taylor, C.K.I. Williams, N. Cristianini, and J.S. Kandola, "On the Eigenspectrum of the Gram Matrix and the Generalization Error of Kernel PCA," *IEEE Trans. Information Theory,* vol. 51, no. 7, pp. 2510-2522, 2005.

[33] M. Ledoux and M. Talagrand, *Probability in Banach Spaces: Isoperimetry and Processes.* Springer, May 1991.

[34] P.L. Bartlett and S. Mendelson, "Rademacher and Gaussian Complexities: Risk Bounds and Structural Results," *J. Machine Learning Research,* vol. 3, pp. 463-482, 2003.

[35] H.A. Guvenir and I. Uysal, *Bilkent University Function Approximation Repository,* http://funapp.cs.bilkent.edu.tr, 2000.

[36] D.J. Newman, S. Hettich, C.L. Blake, and C.J. Merz, *UCI Repository of Machine Learning Databases,* http://www.ics.uci.edu/mlearn/MLRepository.html, 1998.

[37] T. Rose, M. Stevenson, and M. Whitehead, "The Reuters Corpus Volume 1—From Yesterday's News to Tomorrow's Language Resources," *Proc. Third Int'l Conf. Language Resources and Evaluation,* pp. 827-832, 2002.

[38] C.-C. Chang and C.-J. Lin, *LIBSVM: A Library for Support Vector Machines,* software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.

[39] V.N. Vapnik, *Statistical Learning Theory.* Wiley-Interscience, Sept. 1998.

[40] T. Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features," *Proc. 10th European Conf. Machine Learning,* C. Nédellec and C. Rouveirol, eds., pp. 137-142, 1998.

[41] T. Joachims, "Training Linear SVMs in Linear Time," *Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining,* T. Eliassi-Rad, L.H. Ungar, M. Craven, and D. Gunopulos, eds., pp. 217-226, 2006.
[42] W.-Y. Zhao, R. Chellappa, P.J. Phillips, and A. Rosenfeld, "Face Recognition: A Literature Survey," *ACM Computing Surveys,* vol. 35, no. 4, pp. 399-458, 2003.
[43] M.A. Turk and A.P. Pentland, "Eigenfaces for Recognition," *J. Cognitive Neuroscience,* vol. 3, no. 1, pp. 71-86, 1991.
[44] *MIT CBCL Face Database 1.* Center for Biological and Computational Learning, MIT, http://www.ai.mit.edu/projects/cbcl, 1996.
[45] H.A. Rowley, S. Baluja, and T. Kanade, "Neural Network-Based Face Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 1, pp. 23-38, Jan. 1998.
[46] P. Viola and M.J. Jones, "Robust Real-Time Face Detection," *Int'l J. Computer Vision,* vol. 57, no. 2, pp. 137-154, May 2004.
[47] B. Heisele, T. Poggio, and M. Pontil, "Face Detection in Still Gray Images," A.I. Memo 1687, Center for Biological and Computational Learning, Massachusetts Inst. of Technology, 2000.

Charanpal Dhanjal received the BSc degree in computer science from the University of Southampton in 2002 and the MSc degree in advanced computing from Imperial College, London, in 2003. He is currently working toward the PhD degree in the School of Electronics and Computer Science at the University of Southampton.

Steve R. Gunn received the degree (with first class honors) in electronic engineering from the University of Southampton in 1992 and the PhD degree from the University of Southampton in 1996. Between 1996 and 1998, he worked as a research fellow investigating intelligent data modeling algorithms. In 1998, he became a lecturer and, in 2002, a senior lecturer in the Information: Signals, Images, Systems (ISIS) research group. In 2007, he was awarded a personal chair in the ISIS research group at the University of Southampton. He is currently the operational coordinator of the Pattern Analysis, Statistical Modelling and Computational Learning (PASCAL) network.

John Shawe-Taylor received the MSc degree in the foundations of advanced information technology from Imperial College and the PhD degree in mathematics from Royal Holloway, University of London, in 1986. He was promoted to professor of computing science in 1996. He moved to the University of Southampton in 2003 to lead the ISIS research group. He was appointed as the director of the Centre for Computational Statistics and Machine Learning, University College London in July 2006. He has coordinated a number of European-wide projects investigating the theory and practice of Machine Learning, including the NeuroCOLT projects. He is currently the coordinator of the Framework VI Network of Excellence in Pattern Analysis, Statistical Modelling and Computational Learning (PASCAL) involving 57 partners. He has coauthored *Introduction to Support Vector Machines*, the first comprehensive account of this new generation of machine learning algorithms. A second book, *Kernel Methods for Pattern Analysis*, appeared in 2004. He has published more than 150 research papers. He is a member of the IEEE and the IEEE Computer Society.