KERNEL DICTIONARY LEARNING

Hien Van Nguyen[1], *Vishal M. Patel*[1], *Nasser M. Nasrabadi*[2], *and Rama Chellappa*[1]

[1]UMIACS, University of Maryland, College Park, MD
[2]U.S. Army Research Laboratory, Adelphi, MD
{hien,pvishalm,rama}@umiacs.umd.edu   nasser.m.nasrabadi.civ@mail.mil

ABSTRACT

In this paper, we present dictionary learning methods for sparse and redundant signal representations in high dimensional feature space. Using the kernel method, we describe how the well-known dictionary learning approaches such as the method of optimal directions and K-SVD can be made nonlinear. We analyze these constructions and demonstrate their improved performance through several experiments on classification problems. It is shown that nonlinear dictionary learning approaches can provide better discrimination compared to their linear counterparts and kernel PCA, especially when the data is corrupted by noise.

Index Terms**—** Kernel methods, dictionary learning, method of optimal directions, K-SVD.

1. INTRODUCTION

Sparse and redundant signal representations have recently drawn much interest in vision, signal and image processing [1]. This is due in part to the fact that signals and images of interest can be sparse or compressible in some dictionary. The dictionary can be either based on a mathematical model of the data or it can be learned directly from the data. It has been observed that learning a dictionary directly from the training data rather than using a predetermined dictionary (i.e. wavelet) usually leads to a more compact representation and hence can provide improved results in many practical image processing applications such as restoration and classification [1].

Several algorithms have been developed for the task of learning dictionaries. Two of the most well-known algorithms are the method of optimal directions (MOD) [2] and the K-SVD algorithm [3]. Given a set of examples $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_n]$, the goal of the K-SVD and MOD algorithms is to find a dictionary $\mathbf{D}$ and a sparse matrix $\mathbf{X}$ that minimize the following representation error

$$(\hat{\mathbf{D}}, \hat{\mathbf{X}}) = \arg\min_{\mathbf{D},\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 \text{ subject to } \|\mathbf{x}_i\|_0 \leq T_0 \ \forall i,$$

where $\mathbf{x}_i$ represent the columns of $\mathbf{X}$ and the ℓ_0 sparsity measure $\|.\|_0$ counts the number of nonzero elements in the representation. Here, $\|\mathbf{A}\|_F$ denotes the Frobenius norm. Both MOD and K-SVD are iterative methods and they alternate between sparse-coding and dictionary update steps.

The representation obtained by learning a dictionary can be further enhanced by exploiting the nonlinearities present in the data [4], [5]. For instance, in [6] it is shown that if the nonlinear sparsity is properly exploited then one can accurately recover *nonlinearly K-sparse* signals from approximately $2K$ measurements, which is far fewer than the number of measurements usually required for signals that are sparse in an orthonormal basis. In this paper, using kernel methods, we develop dictionary learning algorithms that take into account the nonlinear structure of data. Our dictionary learning methods yield representations that are more compact than kernel PCA and able to handle non-linearity better than its linear counterparts. Fig. 1, presents an important comparison in the representation power of kernel PCA and a learned kernel dictionary. A comparison of the mean-squared-error (MSE) of an image from the USPS dataset when approximated from m kernel PCA components and m kernel dictionary atoms (denoted by kernel KSVD) shows that the MSE decays much faster when a learned non-linear dictionary is used. This example shows that the image is *nonlinearly sparse* and learning a dictionary in the high dimensional feature space can provide better representation of data.

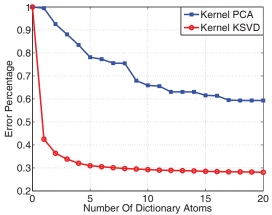


Fig. 1. Comparison of error percentage using kernel K-SVD and kernel PCA.

Background and problem formulation: Let $\Phi : \mathbb{R}^N \rightarrow F$ be a non-linear mapping from $\mathbb{R}^N$ into a higher dimensional feature space F. Since the feature space F can be very high dimensional, in the kernel methods, Mercer kernels are usually employed to carry out the mapping implicitly. A Mercer kernel is a function $\kappa(\mathbf{x}, \mathbf{y})$ that for all data $\{\mathbf{y}_i\}$ gives rise to a positive semidefinite matrix $\mathbb{K}_{ij} = \kappa(\mathbf{y}_i, \mathbf{y}_j)$. It can be shown that using κ instead of dot product in input space corresponds to mapping the data with some mapping Φ into a feature space F. That is, $\kappa(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$. Some commonly used kernels include polynomial kernels $\kappa(\mathbf{x}, \mathbf{y}) = \langle (\mathbf{x}, \mathbf{y} \rangle + c)^d$ and Gaussian kernels $\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{c})$, where c and d are the parameters. Thus, any algorithm that can be formulated in terms of dot products can be carried out in some feature space F without mapping the data explicitly by substituting a chosen kernel.

In this paper, we will use the following model for the dictionary $\mathbf{D}$: $\mathbf{D} = \mathbf{B}\mathbf{A}$, where $\mathbf{B}$ is some predefined base dictionary and $\mathbf{A}$ is the atom representation dictionary [7]. The base dictionary $\mathbf{B}$ can be chosen such that it incorporates some prior knowledge about the data. This model provides adaptivity via modification of the matrix $\mathbf{A}$. Let $\Phi(\mathbf{Y})$ denote the matrix whose columns are obtained by embedding the input signals $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_n]$ into some feature

This work was partially supported by a MURI from the Army Research Office under the Grant W911NF-09-1-0383.

978-1-4673-0046-9/12/$26.00 ©2012 IEEE

space using the mapping Φ. That is, $\Phi(\mathbf{Y}) = [\Phi(\mathbf{y}_1), \cdots, \Phi(\mathbf{y}_n)]$. Furthermore, we denote the learned dictionary in the feature space as $\Phi(\mathbf{D})$. Since dictionary atoms lie within the subspace spanned by the input data, we can write $\Phi(\mathbf{D}) = \Phi(\mathbf{Y})\mathbf{A}$, where $\mathbf{A}$ is the atom representation dictionary and $\Phi(\mathbf{Y})$ is the base dictionary.

Our goal is to find the best dictionary $\Phi(\mathbf{D})$ via $\mathbf{A}$ to represent the data in the feature space $\{\Phi(\mathbf{y}_i)\}_{i=1}^n$ as sparse compositions by solving the following optimization problem

$$\arg\min_{\mathbf{A},\mathbf{X}} \ \|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 \ \ s.t \ \ \|\mathbf{x}_i\|_0 \leq T_0, \forall i. \quad (1)$$

The objective function in (1) can be rewritten as in Eq. (2), which explicitly depends on the kernel matrix $\mathbb{K}$, but not the mapping Φ

$$\|\Phi(\mathbf{Y})-\Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 = \mathbf{tr}((\mathbf{I}-\mathbf{A}\mathbf{X})^T\mathbb{K}(\mathbf{Y},\mathbf{Y})(\mathbf{I}-\mathbf{A}\mathbf{X})), \ (2)$$

where $\mathbb{K}(\mathbf{Y},\mathbf{Y}) \in \mathbb{R}^{n\times n}$ is a positive semidefinite matrix whose elements are computed from the Mercer kernel

$$[\mathbb{K}(\mathbf{Y},\mathbf{Y})]_{ij} = [\langle\Phi(\mathbf{Y}), \Phi(\mathbf{Y})\rangle]_{ij} = \kappa(y_i, y_j).$$

Equipped with the above notation, in the following section, we present two algorithms for learning a dictionary in the feature space.

2. KERNEL DICTIONARY LEARNING

Just as in the case of K-SVD and MOD, our method of learning dictionaries involve two stages: sparse coding and dictionary update. In what follows, we describe them in detail.

Sparse coding: In this stage, the matrix $\mathbf{A}$ is assumed to be fixed. With this, we seek for the sparse codes contained in the matrix $\mathbf{X}$. Note that, the penalty term in (1) can be re-written as

$$\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2 = \sum_{j=1}^{n} \|\Phi(\mathbf{y}_i) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{x}_i\|_2^2.$$

Hence, the problem in (1) can be reformulated as solving n different problems of the following form

$$\min_{\mathbf{x}_i} \ \|\Phi(\mathbf{y}_i) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{x}_i\|_2^2 \ \ s.t \ \ \|\mathbf{x}_i\|_0 \leq T_0, \quad (3)$$

for $i = 1, \cdots, n$. The above problem can be solved by any pursuit algorithms [8, 9], with the modification that signals are now in the feature space with a very high dimension. In the following section, we show how the well-known orthogonal matching pursuit algorithm (OMP) [9, 10] can be generalized using kernels to solve (3).

Kernel Orthogonal Matching Pursuit (KOMP): Given a signal $\mathbf{z} \in \mathbb{R}^N$ and the kernel dictionary represented via $\mathbf{A}$, we seek a sparse combinations of dictionary atoms that approximate the signal in the feature space: $\Phi(\mathbf{z}) = \Phi(\mathbf{Y})\hat{\mathbf{z}}_s + \mathbf{r}_s$. Here, $\hat{\mathbf{z}}_s \in \mathbb{R}^n$ indicates the current estimate of the signal $\mathbf{z}$, and $\mathbf{r}_s$ is the current residual.

The pseudo-code for KOMP is given in the Fig. 2. Let I_s denote the set of indices of selected atoms. In the first step, the residual is projected onto the remaining dictionary atoms:

$$\begin{aligned}\mathbf{r}_s^T(\Phi(\mathbf{Y})\mathbf{a}_i) &= \big(\Phi(\mathbf{z}) - \Phi(\mathbf{Y})\hat{\mathbf{z}}_s\big)^T\big(\Phi(\mathbf{Y})\mathbf{a}_i\big) \\ &= \big(\mathbb{K}(\mathbf{z},\mathbf{Y}) - \mathbb{K}(\mathbf{Y},\mathbf{Y})\hat{\mathbf{z}}_s^T\big)\mathbf{a}_i, \quad i \notin I_s \quad (4)\end{aligned}$$

where, with a slight abuse of notation, we denote

$$\mathbb{K}(\mathbf{z},\mathbf{Y}) = [\kappa(\mathbf{z},\mathbf{y_1}), \kappa(\mathbf{z},\mathbf{y_2}), \ldots, \kappa(\mathbf{z},\mathbf{y_n})].$$

```
Input: A signal $\mathbf{z}$, a kernel function $\kappa$, $\mathbf{A}$, and a sparsity level $T_0$.
Task: Find a coefficient vector $\mathbf{x} \in \mathbb{R}^K$ with at most $T_0$ non-zero coefficients such
that $\Phi(\mathbf{Y})\mathbf{A}\mathbf{x}$ approximates $\Phi(\mathbf{z})$.
Initialize: $s = 0, I_0 = \emptyset, \mathbf{x}_0 = \mathbf{0}, \hat{\mathbf{z}}_0 = \mathbf{0}$
Procedure:
    1. $\tau_i = \Big(\mathbb{K}(\mathbf{z},\mathbf{Y}) - \hat{\mathbf{z}}_s^T\mathbb{K}(\mathbf{Y},\mathbf{Y})\Big)\mathbf{a}_i, \ \forall i \notin I_{s-1}$
    2. $i_{max} = \arg\max_i|\tau_i|, \ \forall i \notin I_{s-1}$
    3. Update the index set $I_s = I_{s-1} \bigcup i_{max}$
    4. $\mathbf{x}_s = \Big(\mathbf{A}_{I_s}^T\mathbb{K}(\mathbf{Y},\mathbf{Y})\mathbf{A}_{I_s}\Big)^{-1}(\mathbb{K}(\mathbf{z},\mathbf{Y})\mathbf{A}_{I_s})^T$
    5. $\hat{\mathbf{z}}_s = \mathbf{A}_{I_s}\mathbf{x}_s$
    6. $s \leftarrow s + 1$; Repeat steps 1–6 $T_0$ times
Output: Sparse vector $\mathbf{x} \in \mathbb{R}^K$ satisfying $\mathbf{x}(I_s(j)) = \mathbf{x}_s(j), \forall j \in I_s$ and zero
elsewhere.
```

Fig. 2. The KOMP algorithm.

The algorithm then selects a new dictionary atom in the remaining set that gives largest projection coefficient in Eq. (4). This selection guarantees the biggest reduction of approximation error.

Let $\mathbf{A}_{I_s}$ indicates the set of dictionary atoms whose indices are from the set I_s. We want to project the signal $\Phi(\mathbf{z})$ onto the subspace spanned by the selected dictionary atoms $\Phi(\mathbf{Y})\mathbf{A}_{I_s}$. The projection coefficients are simply obtained as follows:

$$\begin{aligned}\mathbf{x}_s &= \big((\Phi(\mathbf{Y})\mathbf{A}_{I_s})^T(\Phi(\mathbf{Y})\mathbf{A}_{I_s})\big)^{-1}\big(\Phi(\mathbf{Y})\mathbf{A}_{I_s}\big)^T\Phi(\mathbf{z}) \\ &= \big(\mathbf{A}_{I_s}^T\mathbb{K}(\mathbf{Y},\mathbf{Y})\mathbf{A}_{I_s}\big)^{-1}\big(\mathbb{K}(\mathbf{z},\mathbf{Y})\mathbf{A}_{I_s}\big)^T \quad (5)\end{aligned}$$

Note that the computation in Eq. (5) can be efficiently implemented in a recursive manner as in [9]. Once the coefficients $\mathbf{x}_s$ are found, the approximating signals $\hat{\mathbf{z}}_s$ are updated as in the step 5 of Fig. 2. The procedure is repeated until T_0 atoms are selected.

Dictionary Update: Once the sparse codes for each training data are calculated, the dictionary can be updated such that the error, $\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2$ is minimized. Taking the derivative of this error with respect to $\mathbf{A}$ and after some manipulations, we obtain the relation $\mathbf{A} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$ which leads to the following update: $\mathbf{A}_{k+1} = \mathbf{X}_k^T(\mathbf{X}_k\mathbf{X}_k^T)^{-1} = \mathbf{X}_k^\dagger$.

This way of updating the dictionary is essentially the idea behind the MOD method [2]. As discussed in [3], one of the major drawbacks of the MOD method is that it suffers from the high complexity of matrix inversion during the dictionary update stage. Several other methods have also been proposed that focus on reducing this complexity. One such algorithm is K-SVD [3]. Following the procedure of K-SVD, in what follows, we describe a more sophisticated way of updating the dictionary.

Kernel K-SVD: Let $\mathbf{a}_k$ and $\mathbf{x}_T^j$ denote the k-th column of $\mathbf{A}$ and the j-th row of $\mathbf{X}$, respectively. The error $\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\mathbf{A}\mathbf{X}\|_F^2$ can be re-written as:

$$\begin{aligned}&\left\|\Phi(\mathbf{Y}) - \Phi(\mathbf{Y})\sum_{j=1}^{K}\mathbf{a}_j\mathbf{x}_T^j\right\|_F^2 \\ &= \left\|\Phi(\mathbf{Y})\left(\mathbf{I} - \sum_{j\neq k}\mathbf{a}_j\mathbf{x}_T^j\right) - \Phi(\mathbf{Y})(\mathbf{a}_k\mathbf{x}_T^k)\right\|_F^2 \\ &= \|\Phi(\mathbf{Y})\mathbf{E}_k - \Phi(\mathbf{Y})\mathbf{M}_k\|_F^2 \quad (6)\end{aligned}$$

where,

$$\mathbf{E}_k = \left(\mathbf{I} - \sum_{j\neq k} \mathbf{a}_j \mathbf{x}_T^j\right); \quad \mathbf{M}_k = (\mathbf{a}_k \mathbf{x}_T^k). \tag{7}$$

$\mathbf{E}_k$ indicates the error between the approximated signals and the true signals when removing the k-th dictionary atom. $\mathbf{M}_k$ indicates the contribution of the k-th dictionary atom to the estimated signals.

In this stage, we assume that only $(\mathbf{a}_k, \mathbf{x}_T^k)$ are variables and the rest are fixed, hence $\mathbf{E}_k$ is also constant for each k. Minimization of the above problem is equivalent to finding $(\mathbf{a}_k, \mathbf{x}_T^k)$ such that the rank-1 matrix $\Phi(\mathbf{Y})\mathbf{M}_k$ best approximates $\Phi(\mathbf{Y})\mathbf{E}_k$. The optimal solution can be obtained via SVD. However, there are two reasons that make direct SVD decomposition inappropriate. Firstly, it would yield a dense vector $\mathbf{x}_T^k$, which increases the number of non-zeros in the representation $\mathbf{X}$. Secondly, the matrix might have infinitely large row dimension, which is computationally prohibitive.

In order to minimize the objective function while keeping the sparsities of all the representations fixed, we work only with a subset of columns. Note that the columns of $\mathbf{M}_k$ associated with zero-value elements of $\mathbf{x}_T^k$ are all zero. These columns do not affect the minimization of the objective function. Therefore, we can shrink the matrices $\mathbf{E}_k$ and $\mathbf{M}_k$ by discarding these zero columns. An advantage of working with the reduced matrices is that only non-zero coefficients in $\mathbf{x}_T^k$ are allowed to vary and therefore the sparsities are preserved [3].

Define ω_k as the group of indices pointing to examples $\{\Phi(\mathbf{y}_i)\}$ that use the atom $(\Phi(\mathbf{Y})\mathbf{A})_k$: $\omega_k = \{i | 1 \leq i \leq K, \ \mathbf{x}_T^k(i) \neq 0\}$. Let Ω_k be a matrix of size $n \times |\omega_k|$, with ones on the $(\omega_k(i), i)$-th entries and zeros elsewhere. When multiplying with Ω_k, all zeros within the row vector $\mathbf{x}_T^k$ will be discarded resulting in the row vector $\mathbf{x}_R^k$ of the length $|\omega_k|$. The column-reduced matrices are obtained as $\mathbf{E}_k^R = \mathbf{E}_k\Omega_k; \quad \mathbf{M}_k^R = \mathbf{M}_k\Omega_k$.

We can now modify the cost function in (6) so that its solution has the same support with the original $\mathbf{x}_T^k$:

$$\left\|\Phi(\mathbf{Y})\mathbf{E}_k^R - \Phi(\mathbf{Y})\mathbf{M}_k^R\right\|_F^2 = \left\|\Phi(\mathbf{Y})\mathbf{E}_k^R - \Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x}_R^k\right\|_F^2. \tag{8}$$

Recall the fact that $\Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x}_R^k$ is a rank-1 matrix. Therefore, the optimal solution of (8) can be obtained by first decomposing $\Phi(\mathbf{Y})\mathbf{E}_k^R$ into rank-1 matrices using SVD, and then equating $\Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x}_R^k$ to the rank-1 matrix corresponding to the largest singular value. That is,

$$\Phi(\mathbf{Y})\mathbf{E}_k^R = \mathbf{U}\Sigma\mathbf{V}^T \tag{9}$$

$$\Phi(\mathbf{Y})\mathbf{a}_k\mathbf{x}_R^k = \sigma_1\mathbf{u}_1\mathbf{v}_1^T, \tag{10}$$

where $\mathbf{u}_1$ and $\mathbf{v}_1$ are the first columns of $\mathbf{U}$ and $\mathbf{V}$ corresponding to the largest singular value $\sigma_1 = \Sigma(1,1)$, respectively. A valid breakdown for the assignment (10) is given. The reason for putting the multiplier σ_1 in Eq. (11) instead of in Eq. (12) will become clearer when solving for $\mathbf{a}_k$. Basically, such assignment guarantees that the resulting dictionary atom on the feature space is normalized to unit-norm

$$\mathbf{x}_R^k = \sigma_1\mathbf{v}_1^T \tag{11}$$

$$\Phi(\mathbf{Y})\mathbf{a}_k = \mathbf{u}_1. \tag{12}$$

However, as mentioned before, we can not perform direct SVD decomposition on $\Phi(\mathbf{Y})\mathbf{E}_k^R$ as in (9) since this matrix can have infinitely large row dimension. A remedy for this issue comes from the fact that SVD decomposition is closely related to eigen decomposition of the Gram matrix, which is independent of the row dimension. It is easily seen that

$$\left(\Phi(\mathbf{Y})\mathbf{E}_k^R\right)^T\left(\Phi(\mathbf{Y})\mathbf{E}_k^R\right) = \left(\mathbf{E}_k^R\right)^T\mathbb{K}(\mathbf{Y},\mathbf{Y})(\mathbf{E}_k^R) = \mathbf{V} \bigtriangleup \mathbf{V}^T,$$

where $\bigtriangleup = \Sigma^T\Sigma \in \mathbb{R}^{n\times n}$. This gives us $\mathbf{v}_1$ as the first column of $\mathbf{V}$, and $\sigma_1 = \sqrt{\bigtriangleup(1,1)}$. Hence, $\mathbf{x}_R^k$ can be found using the relation in (11).

To solve for $\mathbf{a}_k$, we first observe that by right-multiplying both sides of (9) by $\mathbf{V}$ and considering only the first column, we get

$$\Phi(\mathbf{Y})\mathbf{E}_k^R\mathbf{v}_1 = \sigma_1\mathbf{u}_1. \tag{13}$$

The solution for $\mathbf{a}_k$ is obtained by substituting Eq. (12) into Eq. (13) $\Phi(\mathbf{Y})\mathbf{E}_k^R\mathbf{v}_1 = \sigma_1\Phi(\mathbf{Y})\mathbf{a}_k$. Hence, $\mathbf{a}_k = \sigma_1^{-1}\mathbf{E}_k^R\mathbf{v}_1$. One can easily verify that this updating procedure of $\mathbf{a}_k$ results in a dictionary atom of unit-norm on the feature space. The pseudo-code for kernel K-SVD algorithm is given in Fig. 3.

```
**Input:** A set of signals $\mathbf{Y}$, a kernel function $\kappa$.
**Task:** Find a dictionary via $\mathbf{A}$ to represent these signals as sparse decompositions in the feature space by solving Eq. (1).
**Initialize:** Set $T_0$ random elements of each column in $\mathbf{X}$ to be 1. Set iteration $J = 1$.
**Stage 1**: *Sparse coding*
Use the KOMP algorithm described in Fig. 2 to obtain sparse coefficient matrix $\mathbf{X}$ given a fixed dictionary $\mathbf{A}$.
**Stage 2**: *Dictionary update*
For each column $k = 1, 2, \ldots, K$ in $\mathbf{A}^{(J-1)}$, update it by
- Define the group of examples that use this atom, $\omega_k = \{i | 1 \leq i \leq N, \mathbf{x}_T^k(i) \neq 0\}$
- Define the representation error matrix, $\mathbf{E}_k$, by (7).
- Restrict $\mathbf{E}_k$ by choosing only the columns corresponding to $\omega_k$, and obtain $\mathbf{E}_k^R$ as $\mathbf{E}_k^R = \mathbf{E}_k\Omega_k$
- Apply SVD decomposition to get $(\mathbf{E}_k^R)^T\mathbb{K}(\mathbf{Y},\mathbf{Y})(\mathbf{E}_k^R) = \mathbf{V} \bigtriangleup \mathbf{V}^T$. Choose updated $\mathbf{a}_k = \sigma_1^{-1}\mathbf{E}_k^R\mathbf{v}_1$, where $\mathbf{v}_1$ is the first vector of $\mathbf{V}$ corresponding to the largest singular value $\sigma_1^2 = \bigtriangleup(1,1)$.
- Set $J = J + 1$
**Output:** $\mathbf{A}$ and $\mathbf{X}$.
```

Fig. 3. The kernel K-SVD algorithm.

3. EXPERIMENTAL RESULTS

First we present two synthetic experiments to examine the effectiveness of a learned dictionary in the feature space. The following parameters are used to learn dictionaries using both K-SVD and kernel K-SVD: dictionaries are learned with 30 atoms, $T_0 = 3$, polynomial kernel of degree 2 is used, the maximum number of training iterations is set to 80.

The first synthetic experiment is done with two classes of data. In each class, 1500 data samples are randomly generated from a 2-dimensional circle $\{y = [y_1, y_2] \in \mathbb{R}^2 \mid y_1^2 + y_2^2 = r^2\}$. The radius r of the first circle (class 1) is half that of the second circle (class 2). The first figure in the left column of Fig. 4 shows the color-coded map of error ratio obtained by dividing the reconstruction errors of the second class by those of the first class for all points on the $\mathbb{R}^2$ plane. Since data samples from the two classes lie roughly on the same linear subspace, which is the entire plane in $\mathbb{R}^2$, dictionaries learned using K-SVD are indistinguishable for the two classes. This is clearly seen from this figure where error ratios are quite random even for points lying on the circles.

On the contrary, as can be seen from the first figure in the second row of Fig. 4, the error ratios corresponding to a dictionary learned in the feature space exhibit strong differences between the two classes. In particular, error ratios are very high for points lying close to the first class, and very small for points lying close to the second class. Moreover, points on the same circle have similar error ratios. This observation implies that kernel K-SVD correctly learns the nonlinear structure of the data and embeds this information into kernel dictionary atoms.

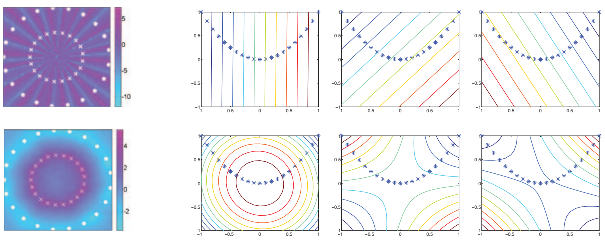


Fig. 4. Left: Comparison of error ratio for K-SVD and kernel K-SVD (common logarithm scale). Right: Comparison between contours of linear K-SVD and kernel K-SVD for three different dictionary atoms. In both figures, the first row corresponds to K-SVD and the second row corresponds to kernel K-SVD.

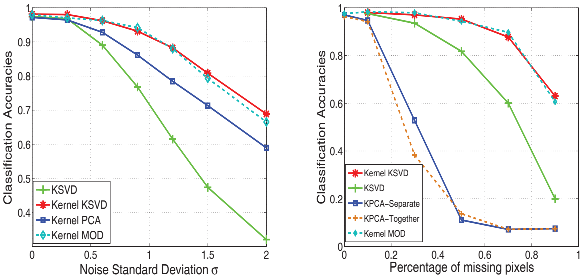
In the second synthetic experiment, we learn a dictionary from 1500 data samples generated from a 2-dimensional parabola $\{y = [y_1, y_2] \in \mathbb{R}^2 \mid y_2 = y_1^2\}$. Columns 2-4 in Fig. 4 show level curves of the projection coefficients for three different dictionary atoms. The level curves are obtained as follows. First, we project every point $y \in \mathbb{R}^2$ onto the selected dictionary atom to get the projection coefficients. Then, points with the same projection coefficients are grouped together and are shown with the same color map. Coefficients of the kernel K-SVD (Bottom row of columns 2-4 in Fig. 4) change most dramatically along the main directions of data's variation, while coefficients of the linear K-SVD do not. Again, this observation implies that our dictionary learning method can provide good representation for data with non-linear structures.

Digit Recognition: In recent years, there has been a great interest in applying dictionary learning methods for classification. To this end, we apply our approach on the real-world handwritten digits classification problem. We use the USPS database which contains ten classes of 256-dimensional handwritten digits. For each class, we randomly select 300 samples for training and 200 samples for testing. We use the following parameters for learning dictionaries: dictionaries are learned with 500 atoms, $T_0 = 5$, polynomial kernel of degree 4 is used, maximum number of training iterations are set to 80.

We use the generative approach to do classification. In particular, digits are classified to the classes that give the smallest reconstruction error. Let $\{\mathbf{A}_i\}_{i=1}^{c}$ denote the learned dictionaries for c classes. Given a query image $\mathbf{z}$, we first perform KOMP on each $\mathbf{A}_i$ to get the sparse code $\mathbf{x_i}$. The reconstruction error is then computed as:

$$\begin{aligned}\mathbf{r_i} &= ||\Phi(\mathbf{z}) - \Phi(\mathbf{Y})\mathbf{A}_i\mathbf{x_i}||^2 \\ &= \mathbb{K}(\mathbf{z},\mathbf{z}) - 2\mathbb{K}(\mathbf{z},\mathbf{Y})\mathbf{A}_i\mathbf{x_i} + \mathbf{x_i}^T\mathbf{A}_i^T\mathbb{K}(\mathbf{Y},\mathbf{Y})\mathbf{A}_i\mathbf{x_i}. \quad (14)\end{aligned}$$

For kernel PCA, we project the query image $\mathbf{z}$ onto the first 500 principal components and train a linear SVM classifier using these coefficients for classification. Note that in the case of K-SVD, kernel MOD and kernel K-SVD, dictionaries are trained separately for each class while kernel PCA uses all training samples to obtain projection coefficients. For fair comparisons, we have also obtained projection coefficients by training separate dictionaries for each class using kernel PCA.



(a) Gaussian noise

(b) Missing pixels

Fig. 5. Comparison of digit recognition accuracies for different methods in the presence of Gaussian noise and missing-pixel effects.

The first experiment presents the results for the situation where test samples are corrupted by random Gaussian noise with different standard deviations as shown in Fig. 5(a). Fig. 5(b) shows the results obtained when pixels are randomly removed from the test images. In both experiments, Kernel K-SVD and kernel MOD consistently outperform linear K-SVD and kernel PCA. As the distortion level increases the performance difference between kernel dictionaries and linear dictionaries become more dramatic.

4. DISCUSSION AND CONCLUSION

We have presented two non-linear dictionary learning algorithms that exploit sparsity of data in high dimensional feature space through an appropriate choice of kernel. It is shown that kernel methods improve the separating margin between dictionaries and allow better tolerance against different types of degradations. Experimental results indicate that exploiting nonlinear sparsity via learning dictionaries in the feature domain can provide better discrimination than the traditional linear approaches and kernel PCA.

5. REFERENCES

[1] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proceedings of the IEEE*, submitted 2009.

[2] K. Engan, S. O. Aase, and J. H. Husoy, "Method of optimal directions for frame design," *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 5, pp. 2443–2446, 1999.

[3] M. Aharon, M. Elad, and A. M. Bruckstein, "The k-svd: an algorithm for designing of overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, 2006.

[4] S. Gao, I. W. Tsang, and L.-T. Chia, "Kernel sparse representation for image classification and face recognition," in *European Conference on Computer Vision*, vol. 6314, 2010.

[5] X.-T. Yuan and S. Yan, "Visual classification with multi-task joint sparse representation," in *Comp. Vision and Pattern Recognition*, 2010.

[6] H. Qi and S. Hughes, "Using the kernel trick in compressive sensing: Accurate signal recovery from fewer measurements," in *IEEE Int. Conf. on Acoustics, Speech and Signal Proc.*, may 2011, pp. 3940 –3943.

[7] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *Signal Processing, IEEE Transactions on*, vol. 58, no. 3, pp. 1553 –1564, march 2010.

[8] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comp.*, vol. 20, no. 1, pp. 33–61, 1998.

[9] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," *1993 Conference Record of the 27th Asilomar Conference on Signals, Systems and Computers*, pp. 40–44, 1993.

[10] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Info. Theory*, vol. 50, no. 10, pp. 2231–2242, Oct. 2004.