The Pre-Image Problem in Kernel Methods

James T. Kwok JAMESK@CS.UST.HK
Ivor W. Tsang IVOR@CS.UST.HK
Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Abstract

In this paper, we address the problem of finding the pre-image of a feature vector in the feature space induced by a kernel. This is of central importance in some kernel applications, such as on using kernel principal component analysis (PCA) for image denoising. Unlike the traditional method in (Mika et al., 1998) which relies on nonlinear optimization, our proposed method directly finds the location of the pre-image based on distance constraints in the feature space. It is non-iterative, involves only linear algebra and does not suffer from numerical instability or local minimum problems. Performance of this method is evaluated on performing kernel PCA and kernel clustering on the USPS data set.

1. Introduction

In recent years, there has been a lot of interest in the study of kernel methods (Schölkopf & Smola, 2002; Vapnik, 1998). The basic idea is to map the data in the input space $\mathcal{X}$ to a feature space via some nonlinear map φ, and then apply a linear method there. It is now well-known that the computational procedure depends only on the inner products[1] $\varphi(\mathbf{x}_i)'\varphi(\mathbf{x}_j)$ in the feature space (where $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$), which can be obtained efficiently from a suitable *kernel* function $k(\cdot,\cdot)$. Besides, kernel methods have the important computational advantage that no nonlinear optimization is involved. Thus, the use of kernels provides elegant nonlinear generalizations of many existing linear algorithms. A well-known example in supervised learning is the support vector machines (SVMs). In unsupervised learning, the kernel idea has also led to methods such as kernel-based clustering algorithms (Girolami, 2002), kernel independent component analysis and kernel principal component analysis (PCA) (Schölkopf et al., 1998b).

While the mapping φ from input space to feature space is of primary importance in kernel methods, the reverse mapping from feature space back to input space (the *pre-image* problem) is also useful. Consider for example the use of kernel PCA for pattern denoising. Given some noisy patterns, kernel PCA first applies linear PCA on the φ-mapped patterns in the feature space, and then performs denoising by projecting them onto the subspace defined by the leading eigenvectors. These projections, however, are still in the feature space and have to be mapped back to the input space in order to recover the denoised patterns. Another example is in visualizing the clustering solution of a kernel-based clustering algorithm. Again, this involves finding the pre-images of, say, the cluster centroids in the feature space. More generally, methods for finding pre-images can be used as *reduced set methods* to compress a kernel expansion (which is a linear combination of many feature vectors) into one with fewer terms. They can offer significant speed-ups in many kernel applications (Burges, 1996; Schölkopf et al., 1998a).

However, the exact pre-image typically does not exist (Mika et al., 1998), and one can only settle for an approximate solution. But even this is non-trivial as the dimensionality of the feature space can be infinite. (Mika et al., 1998) cast this as a nonlinear optimization problem, which, for particular choices of kernels (such as the Gaussian kernel[2]), can be solved by a fixed-point iteration method. However, as mentioned in (Mika et al., 1998), this method suffers from nu-

[1] In this paper, vector/matrix transpose (in both the input and feature spaces) is denoted by the superscript $'$.

[2] We will show in Section 4.1.2 that an analogous iteration formula can also be derived for polynomial kernels.

Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

merical instabilities. Moreover, as in any nonlinear optimization problem, one can get trapped in a local minimum and the pre-image obtained is thus sensitive to the initial guess.

While the inverse of φ typically does not exist, there is usually a simple relationship between feature-space distance and input-space distance for many commonly used kernels (Williams, 2001). In this paper, we use this relationship together with the idea in multidimensional scaling (MDS) (Cox & Cox, 2001) to address the pre-image problem. The proposed method is noniterative and involves only linear algebra.

Our exposition in the sequel will focus on the pre-image problem in kernel PCA, though it can be applied equally well to other kernel methods. The rest of this paper is organized as follows. Brief introduction to the kernel PCA is given in Section 2. Section 3 then describes our proposed method. Experimental results are presented in Section 4, and the last section gives some concluding remarks.

2. Kernel PCA

2.1. PCA in the Feature Space

In this Section, we give a short review on the kernel PCA. While standard expositions (Mika et al., 1998; Schölkopf et al., 1998b) usually focus on the simpler case where the φ-mapped patterns have been centered, we will carry out this centering explicitly in the following.

Given a set of patterns $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^d$. Kernel PCA performs the traditional linear PCA in the feature space. Analogous to linear PCA, it also involves an eigen decomposition $\mathbf{HKH} = \mathbf{U\Lambda U}'$, where $\mathbf{K}$ is the kernel matrix with $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$,

$$\mathbf{H} = \mathbf{I} - \frac{1}{N}\mathbf{1}\mathbf{1}' \tag{1}$$

is the centering matrix, $\mathbf{I}$ is the $N \times N$ identity matrix, $\mathbf{1} = [1, 1, \ldots, 1]'$ is an $N \times 1$ vector, $\mathbf{U} = [\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N]$ with $\boldsymbol{\alpha}_i = [\alpha_{i1}, \ldots, \alpha_{iN}]'$ and $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$. Denote the mean of the φ-mapped patterns by $\bar{\varphi} = \frac{1}{N}\sum_{i=1}^N \varphi(\mathbf{x}_i)$ and define the "centered" map $\tilde{\varphi}$ as:

$$\tilde{\varphi}(\mathbf{x}) = \varphi(\mathbf{x}) - \bar{\varphi}.$$

The kth orthonormal eigenvector of the covariance matrix in the feature space can then be shown to be

$$\mathbf{V}_k = \sum_{i=1}^N \frac{\alpha_{ki}}{\sqrt{\lambda_k}}\tilde{\varphi}(\mathbf{x}_i) = \frac{1}{\sqrt{\lambda_k}}\tilde{\boldsymbol{\varphi}}\boldsymbol{\alpha}_k,$$

where $\tilde{\boldsymbol{\varphi}} = [\tilde{\varphi}(\mathbf{x}_1), \tilde{\varphi}(\mathbf{x}_2), \ldots, \tilde{\varphi}(\mathbf{x}_N)]$. Denote the projection of the φ-image of a pattern $\mathbf{x}$ onto the kth component by β_k. Then,

$$\begin{aligned}\beta_k &= \tilde{\varphi}(\mathbf{x})'\mathbf{V}_k = \frac{1}{\sqrt{\lambda_k}}\sum_{i=1}^N \alpha_{ki}\tilde{\varphi}(\mathbf{x})'\tilde{\varphi}(\mathbf{x}_i) \\ &= \frac{1}{\sqrt{\lambda_k}}\sum_{i=1}^N \alpha_{ki}\tilde{k}(\mathbf{x}, \mathbf{x}_i), \end{aligned} \tag{2}$$

where

$$\begin{aligned}\tilde{k}(\mathbf{x}, \mathbf{y}) &= \tilde{\varphi}(\mathbf{x})'\tilde{\varphi}(\mathbf{y}) \\ &= (\varphi(\mathbf{x}) - \bar{\varphi})'(\varphi(\mathbf{y}) - \bar{\varphi}) \\ &= k(\mathbf{x}, \mathbf{y}) - \frac{1}{N}\sum_{i=1}^N k(\mathbf{x}, \mathbf{x}_i) - \frac{1}{N}\sum_{i=1}^N k(\mathbf{x}_i, \mathbf{y}) \\ &\quad + \frac{1}{N^2}\sum_{i,j=1}^N k(\mathbf{x}_i, \mathbf{x}_j) \\ &= k(\mathbf{x}, \mathbf{y}) - \frac{1}{N}\mathbf{1}'\mathbf{k}_\mathbf{x} - \frac{1}{N}\mathbf{1}'\mathbf{k}_\mathbf{y} + \frac{1}{N^2}\mathbf{1}'\mathbf{K}\mathbf{1}, \end{aligned}$$

and $\mathbf{k}_\mathbf{x} = [k(\mathbf{x}, \mathbf{x}_1), \ldots, k(\mathbf{x}, \mathbf{x}_N)]'$. Denote

$$\begin{aligned}\tilde{k}_\mathbf{x} &= [\tilde{k}(\mathbf{x}, \mathbf{x}_1), \ldots, \tilde{k}(\mathbf{x}, \mathbf{x}_N)]' \\ &= \mathbf{k}_\mathbf{x} - \frac{1}{N}\mathbf{1}\mathbf{1}'\mathbf{k}_\mathbf{x} - \frac{1}{N}\mathbf{K}\mathbf{1} + \frac{1}{N^2}\mathbf{1}\mathbf{1}'\mathbf{K}\mathbf{1} \\ &= \mathbf{H}\left(\mathbf{k}_\mathbf{x} - \frac{1}{N}\mathbf{K}\mathbf{1}\right), \end{aligned} \tag{3}$$

then (2) can be written more compactly as $\beta_k = \frac{1}{\sqrt{\lambda_k}}\boldsymbol{\alpha}_k'\tilde{k}_\mathbf{x}$.

The projection $P_K\varphi(\mathbf{x})$ of $\varphi(\mathbf{x})$ onto the subspace spanned by the first K eigenvectors[3] is then

$$\begin{aligned}P_K\varphi(\mathbf{x}) &= \sum_{k=1}^K \beta_k\mathbf{V}_k + \bar{\varphi} = \sum_{k=1}^K \frac{1}{\lambda_k}(\boldsymbol{\alpha}_k'\tilde{k}_\mathbf{x})(\tilde{\boldsymbol{\varphi}}\boldsymbol{\alpha}_k) + \bar{\varphi} \\ &= \tilde{\boldsymbol{\varphi}}\mathbf{M}\tilde{k}_\mathbf{x} + \bar{\varphi}, \end{aligned} \tag{4}$$

where $\mathbf{M} = \sum_{k=1}^K \frac{1}{\lambda_k}\boldsymbol{\alpha}_k\boldsymbol{\alpha}_k'$ is symmetric.

2.2. Iterative Scheme for Finding the Pre-Image

As $P\varphi(\mathbf{x})$ is in the feature space, we have to find its pre-image $\hat{\mathbf{x}}$ in order to recover the denoised pattern (Figure 1). As mentioned in Section 1, the exact pre-image may not even exist, and so we can only recover an $\hat{\mathbf{x}}$ where $\varphi(\hat{\mathbf{x}}) \simeq P\varphi(\mathbf{x})$. (Mika et al., 1998) addressed this problem by minimizing the squared distance between $\varphi(\hat{\mathbf{x}})$ and $P\varphi(\mathbf{x})$:

$$\|\varphi(\hat{\mathbf{x}}) - P\varphi(\mathbf{x})\|^2 = \|\varphi(\hat{\mathbf{x}})\|^2 - 2P\varphi(\mathbf{x})'\varphi(\hat{\mathbf{x}}) + \Omega, \tag{5}$$

[3]For simplicity, $P_K\varphi(\mathbf{x})$ will often be denoted as $P\varphi(\mathbf{x})$ in the sequel.

where Ω includes terms independent of $\hat{\mathbf{x}}$. This, however, is a nonlinear optimization problem. As mentioned in Section 1, it will be plagued by the problem of local minimum and is sensitive to the initial guess of $\hat{\mathbf{x}}$.

For particular choices of kernels, such as Gaussian kernels of the form $k(\mathbf{x}, \mathbf{y}) = \exp(\|\mathbf{x} - \mathbf{y}\|^2/c)$, this nonlinear optimization can be solved by a fixed-point iteration method. On setting the derivative of (5) to zero, the following iteration formula is obtained:

$$\hat{\mathbf{x}}_{t+1} = \frac{\sum_{i=1}^{N} \tilde{\gamma}_i \exp(-\|\hat{\mathbf{x}}_t - \mathbf{x}_i\|^2/c)\mathbf{x}_i}{\sum_{i=1}^{N} \tilde{\gamma}_i \exp(-\|\hat{\mathbf{x}}_t - \mathbf{x}_i\|^2/c)}. \quad (6)$$

Here[4], $\gamma_i = \sum_{k=1}^{K} \beta_k \alpha_{ki}$ and $\tilde{\gamma}_i = \gamma_i + \frac{1}{N}(1 - \sum_{j=1}^{N} \gamma_j)$. However, as mentioned in (Mika et al., 1998), this iteration scheme is numerically unstable and one has to try a number of initial guesses for $\hat{\mathbf{x}}$.

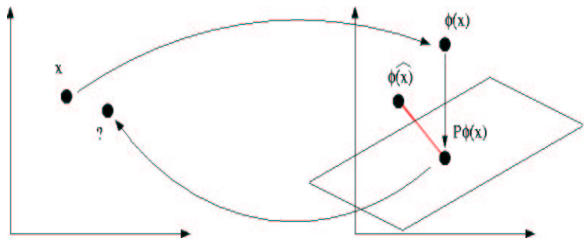

Figure 1. The pre-image problem in kernel PCA.

Notice from (6) that the pre-image obtained is in the span of $\mathbf{x}_i$'s. Besides, because of the exponential $\exp(-\|\hat{\mathbf{x}}_t - \mathbf{x}_i\|^2/c)$, the contributions of $\mathbf{x}_i$'s typically drop rapidly with increasing distance from the pre-image. These observations will be useful in Section 3.

3. Finding the Pre-Image Based on Distance Constraints

For any two points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the input space, we can obtain their Euclidean distance $d(\mathbf{x}_i, \mathbf{x}_j)$. Analogously, we can also obtain the feature-space distance $\tilde{d}(\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j))$ between their φ-mapped images. Moreover, for many commonly used kernels, there is a simple relationship between $d(\mathbf{x}_i, \mathbf{x}_j)$ and $\tilde{d}(\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j))$ (Williams, 2001). The idea of the proposed method is then as follows (Figure 2). Let the pattern to be denoised be $\mathbf{x}$. As mentioned in Section 1, the corresponding $\varphi(\mathbf{x})$ will be projected to $P\varphi(\mathbf{x})$ in the feature space. For each training pattern $\mathbf{x}_i$, this $P\varphi(\mathbf{x})$ will be at a distance $\tilde{d}(P\varphi(\mathbf{x}), \varphi(\mathbf{x}_i))$ from each $\varphi(\mathbf{x}_i)$ in the feature space. Using the distance relationship mentioned above, we can obtain the corresponding input-space distance between the desired pre-image $\hat{\mathbf{x}}$ and each of the $\mathbf{x}_i$'s. Now, in multidimensional scaling (MDS)[5] (Cox & Cox, 2001), one attempts to find a representation of the objects that preserves the dissimilarities between each pair of them. Here, we will use this MDS idea to embed $P\varphi(\mathbf{x})$ back to the input space. When the exact pre-image exists, it would have exactly satisfied these input-space distance constraints[6]. In cases where the exact pre-image does not exist, we will require the approximate pre-image to satisfy these constraints approximately (to be more precise, in the least-square sense).

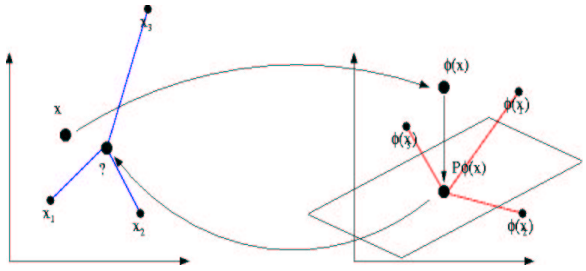

Figure 2. Basic idea of the proposed method.

Notice that instead of finding the pre-image of $P\varphi(\mathbf{x})$ in kernel PCA, this procedure can also be used to find the pre-image of any feature vector in the feature space. For example, we can use this to find the pre-images of the cluster centroids obtained from some kernel clustering algorithm, as will be demonstrated in Section 4.

The following sections describe these steps in more detail. Computation of the feature-space distances $\tilde{d}(P\varphi(\mathbf{x}), \varphi(\mathbf{x}_i))$ is described in Section 3.1. Section 3.2 uses the distance relationship to obtain the corresponding distances in the input space. Finally, Section 3.3 uses these distances to constrain the embedding of the pre-image.

3.1. Distances in the Feature Space

For any two patterns $\mathbf{x}$ and $\mathbf{x}_i$, the squared feature-space distance between the projection $P\varphi(\mathbf{x})$ and

[4]The apparent difference with the equations in (Mika et al., 1998) is because we explicitly perform centering of the φ-mapped patterns here.

[5]Interested readers may also refer to (Cox & Cox, 2001) for a connection between PCA and MDS, and to (Williams, 2001) for a connection between kernel PCA and kernel MDS.

[6]One can visualize these $\mathbf{x}_i$'s as range sensors (or global positioning system satellites) that help to pinpoint the location of an object (i.e., the pre-image).

$\varphi(\mathbf{x}_i)$ is given by:

$$\tilde{d}^2(P\varphi(\mathbf{x}), \varphi(\mathbf{x}_i)) = \|P\varphi(\mathbf{x})\|^2 + \|\varphi(\mathbf{x}_i)\|^2 - 2P\varphi(\mathbf{x})'\varphi(\mathbf{x}_i). \quad (7)$$

Now, from (3) and (4), we have

$$\begin{aligned}
&\|P\varphi(\mathbf{x})\|^2 \\
&= \left(\sum_{k=1}^{K} \beta_k \mathbf{V}_k + \bar{\varphi}\right)' \left(\sum_{k=1}^{K} \beta_k \mathbf{V}_k + \bar{\varphi}\right) \\
&= \tilde{k}'_{\mathbf{x}} \mathbf{M} \tilde{k}_{\mathbf{x}} + \frac{1}{N^2}\mathbf{1}'\mathbf{K}\mathbf{1} \\
&\quad +2\left(\frac{1}{N}\mathbf{1}'\mathbf{K} - \frac{1}{N^2}\mathbf{1}'\mathbf{K}\mathbf{1}\mathbf{1}'\right)\mathbf{M}\tilde{k}_{\mathbf{x}} \\
&= \left(\mathbf{k}_{\mathbf{x}} + \frac{1}{N}\mathbf{K}\mathbf{1}\right)' \mathbf{H}'\mathbf{M}\mathbf{H}\left(\mathbf{k}_{\mathbf{x}} - \frac{1}{N}\mathbf{K}\mathbf{1}\right) + \frac{1}{N^2}\mathbf{1}'\mathbf{K}\mathbf{1},
\end{aligned}$$

and

$$\begin{aligned}
&P\varphi(\mathbf{x})'\varphi(\mathbf{x}_i) \\
&= (\tilde{\boldsymbol{\varphi}}\mathbf{M}\tilde{k}_{\mathbf{x}} + \bar{\varphi})'\varphi(\mathbf{x}_i) \\
&= \mathbf{k}'_{\mathbf{x}_i}\mathbf{H}'\mathbf{M}\mathbf{H}(\mathbf{k}_{\mathbf{x}} - \frac{1}{N}\mathbf{K}\mathbf{1}) + \frac{1}{N}\mathbf{1}'\mathbf{k}_{\mathbf{x}_i}. \quad (8)
\end{aligned}$$

Thus, (7) becomes:

$$\begin{aligned}
&\tilde{d}^2(P\varphi(\mathbf{x}), \varphi(\mathbf{x}_i)) \\
&= \left(\mathbf{k}_{\mathbf{x}} + \frac{1}{N}\mathbf{K}\mathbf{1} - 2\mathbf{k}_{\mathbf{x}_i}\right)' \mathbf{H}'\mathbf{M}\mathbf{H}\left(k_{\mathbf{x}} - \frac{1}{N}\mathbf{K}\mathbf{1}\right) \\
&\quad + \frac{1}{N^2}\mathbf{1}'\mathbf{K}\mathbf{1} + K_{ii} - \frac{2}{N}\mathbf{1}'\mathbf{k}_{\mathbf{x}_i}, \quad (9)
\end{aligned}$$

where $K_{ii} = k(\mathbf{x}_i, \mathbf{x}_i)$.

3.2. Distances in the Input Space

Given the feature-space distances between $P\varphi(\mathbf{x})$ and the φ-mapped training patterns (Section 3.1), we now proceed to find the corresponding input-space distances, which will be preserved when $P\varphi(\mathbf{x})$ is embedded back to the input space (Section 3.3). Recall that the distances with neighbors are the most important in determining the location of any point (Section 2.2). Hence, in the following, we will only consider the (squared) input-space distances between $P\varphi(\mathbf{x})$ and its n nearest neighbors[7], i.e.,

$$\mathbf{d}^2 = [d_1^2, d_2^2, \ldots, d_n^2]'. \quad (10)$$

This in turn can offer significant speed-up, especially during the singular value decomposition step in Section 3.3. Moreover, this is also in line with the ideas in metric multidimensional scaling (Cox & Cox, 2001), in which smaller dissimilarities are given more weight, and in locally linear embedding (Roweis & Saul, 2000), where only the local neighborhood structure needs to be preserved.

We first consider isotropic kernels[8] of the form $k(\mathbf{x}_i, \mathbf{x}_j) = \kappa(\|\mathbf{x}_i - \mathbf{x}_j\|^2)$. There is a simple relationship between the feature-space distance $\tilde{d}_{ij}$ and the input-space distance d_{ij} (Williams, 2001):

$$\begin{aligned}
\tilde{d}_{ij}^2 &= \tilde{d}^2(\mathbf{x}_i, \mathbf{x}_j) = K_{ii} + K_{jj} - 2\kappa(\|\mathbf{x}_i - \mathbf{x}_j\|^2) \\
&= K_{ii} + K_{jj} - 2\kappa(d_{ij}^2),
\end{aligned}$$

and hence,

$$\kappa(d_{ij}^2) = \frac{1}{2}(K_{ii} + K_{jj} - \tilde{d}_{ij}^2). \quad (11)$$

Typically, κ is invertible. For example, for the Gaussian kernel $\kappa(z) = \exp(-\beta z)$ where β is a constant, we have $d_{ij}^2 = -\frac{1}{\beta}\log(\frac{1}{2}(K_{ii} + K_{jj} - \tilde{d}_{ij}^2))$.

Similarly, for dot product kernels of the form $k(\mathbf{x}_i, \mathbf{x}_j) = \kappa(\mathbf{x}_i'\mathbf{x}_j)$, there is again a simple relationship between the dot product $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ in the feature space and the dot product $s_{ij} = \mathbf{x}_i'\mathbf{x}_j$ in the input space (Williams, 2001):

$$K_{ij} = \varphi(\mathbf{x}_i)'\varphi(\mathbf{x}_j) = k(\mathbf{x}_i'\mathbf{x}_j) = \kappa(s_{ij}). \quad (12)$$

Moreover, κ is often invertible. For example, for the polynomial kernel $\kappa(z) = z^p$ where p is the polynomial order, $s_{ij} = K_{ij}^{\frac{1}{p}}$ when p is odd. Similarly, for the sigmoid kernel $\kappa(z) = \tanh(vz - c)$ where $v, c \in \mathbb{R}$ are parameters, $s_{ij} = (\tanh^{-1}(K_{ij}) + c)/v$. The corresponding squared distance in the input space is then

$$d_{ij}^2 = s_{ii}^2 + s_{jj}^2 - 2s_{ij}. \quad (13)$$

Thus, in summary, we can often use (9), (11) for isotropic kernels, or (8), (12), (13) for dot product kernels, to construct the input-space distance vector $\mathbf{d}^2$ in (10).

3.3. Using the Distance Constraints

For the n neighbors $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^d$ obtained in Section 3.2, we will first center them at their centroid $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$ and define a coordinate system in their span. First, construct the $d \times n$ matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$. Using the $n \times n$ centering matrix $\mathbf{H}$ in (1), $\mathbf{H}\mathbf{X}'$ will center the $\mathbf{x}_i$'s at the centroid (i.e. the column sums of $\mathbf{H}\mathbf{X}'$ are zero). Assuming that

[7] In this paper, we use the n nearest neighbors in the feature space. Alternatively, we can use the neighbors in the input space with similar results.

[8] A kernel is isotropic if $k(\mathbf{x}_i, \mathbf{x}_j)$ depends only on the distance $\|\mathbf{x}_i - \mathbf{x}_j\|^2$.

the training patterns span a q-dimensional space (i.e., $\mathbf{X}$ is of rank q), we can obtain the singular value decomposition (SVD) of the $d \times n$ matrix $(\mathbf{HX}')' = \mathbf{XH}$ as:

$$\mathbf{XH} = \mathbf{U\Lambda V}' = \mathbf{UZ},$$

where $\mathbf{U} = [\mathbf{e}_1, \ldots, \mathbf{e}_q]$ is a $d \times q$ matrix with orthonormal columns $\mathbf{e}_i$ and $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_n]$ is a $q \times n$ matrix with columns $\mathbf{z}_i$ being the projections of $\mathbf{x}_i$ onto the $\mathbf{e}_j$'s. Note that the computational complexity for performing SVD on an $d \times n$ matrix is $O(kd^2n + k'n^3)$, where k and k' are constants. Hence, using only the n neighbors instead of all N training patterns can offer a significant speed-up. Besides, the squared distance of $\mathbf{x}_i$ to the origin, which is still at the centroid, is equal to $\|\mathbf{z}_i\|^2$. Again, collect these into an n-dimensional vector, as $\mathbf{d}_0^2 = [\|\mathbf{z}_1\|^2, \ldots, \|\mathbf{z}_n\|^2]'$.

Recall from Section 2.2 that the approximate pre-image $\hat{\mathbf{x}}$ obtained in (Mika et al., 1998) is in the span of the training patterns, with the contribution of each individual $\mathbf{x}_i$ dropping exponentially with its distance from $\hat{\mathbf{x}}$. Hence, we will assume in the following that the required pre-image $\hat{\mathbf{x}}$ is in the span of the n neighbors. As mentioned in Section 3, its location will be obtained by requiring $d^2(\hat{\mathbf{x}}, \mathbf{x}_i)$ to be as close to those values obtained in (10) as possible, i.e.,

$$d^2(\hat{\mathbf{x}}, \mathbf{x}_i) \simeq d_i^2, \quad i = 1, \ldots, n.$$

In the ideal case, we should have exactly preserved these distances. However, as mentioned in Section 1, in general there is no exact pre-image in the input space and so a solution satisfying all these distance constraints may not even exist. Hence, we will settle for the least-square solution $\hat{\mathbf{z}}$. Following (Gower, 1968), this can be shown to satisfy:

$$-2\mathbf{Z}'\hat{\mathbf{z}} = (\mathbf{d}^2 - \mathbf{d}_0^2) - \frac{1}{n}\mathbf{1}\mathbf{1}'(\mathbf{d}^2 - \mathbf{d}_0^2).$$

Now, $\mathbf{Z1}' = \mathbf{0}$ because of the centering. Hence, the pre-image can be obtained as

$$\hat{\mathbf{z}} = -\frac{1}{2}(\mathbf{ZZ}')^{-1}\mathbf{Z}(\mathbf{d}^2 - \mathbf{d}_0^2) = -\frac{1}{2}\mathbf{\Lambda}^{-1}\mathbf{V}'(\mathbf{d}^2 - \mathbf{d}_0^2).$$

This $\hat{\mathbf{z}}$ is expressed in terms of the coordinate system defined by the $\mathbf{e}_j$'s. Transforming back to the original coordinate system in the input space, we thus have

$$\hat{\mathbf{x}} = \mathbf{U}\hat{\mathbf{z}} + \bar{\mathbf{x}}.$$

4. Experiment

4.1. Pre-Images in Kernel PCA

In this Section, we report denoising results on the USPS data set consisting of 16×16 handwritten digits[9]. For each of the ten digits, we randomly choose some examples (300 and 60 respectively) to form the training set, and 100 examples as the test set. Kernel PCA is performed on each digit separately.

Two types of additive noise are then added to the test set. The first one is the Gaussian noise $N(0, \sigma^2)$ with variance σ^2. The second type is the "salt and pepper" noise with noise level p, where $p/2$ is the probability that a pixel flips to black or white. The model selection problem for the number (K) of eigenvectors is sidestepped by choosing $K = \arg\min_n \|P_n\varphi(\mathbf{x}) - \varphi(\widetilde{\mathbf{x}})\|^2$ where $\mathbf{x}$ is the noisy image and $\widetilde{\mathbf{x}}$ is the original (clean) image. 10 neighbors is used in locating the pre-image. Moreover, comparison will be made with the traditional method in (Mika et al., 1998).

4.1.1. Gaussian Kernel

We first experiment with the Gaussian kernel $\exp(-\beta z)$, where we set $\frac{1}{\beta} = \frac{1}{N(N-1)}\sum_{i,j=1}^{N}\|\mathbf{x}_i - \mathbf{x}_j\|^2$. Figures 3 and 4 show some typical noisy test images for the two types of noise, and the corresponding denoised images. Tables 1 and 2 show the numerical comparisons using the signal-to-noise ratio (SNR). As can be seen, the proposed method produces better results both visually and quantitatively.

Table 1. SNRs (in dB) of the denoised images using the Gaussian kernel, at different number of training samples and different noise variances (σ^2) of the Gaussian noise.

number of training images	σ^2	SNR noisy images	SNR our method	SNR Mika *et al.*
300	0.25	2.32	6.36	5.90
	0.3	1.72	6.24	5.60
	0.4	0.91	5.89	5.17
	0.5	0.32	5.58	4.86
60	0.25	2.32	4.64	4.50
	0.3	1.72	4.56	4.39
	0.4	0.90	4.41	4.19
	0.5	0.35	4.29	4.06

4.1.2. Polynomial Kernel

Next, we perform experiment on the polynomial kernel $(\mathbf{xy}+1)^d$ with $d = 3$. Following the exposition of (Mika et al., 1998) in Section 2.2, (5) now becomes $(\hat{\mathbf{x}}'\hat{\mathbf{x}} + 1)^d - 2\sum_{i=1}^{N}\tilde{\gamma}_i(\hat{\mathbf{x}}'\mathbf{x}_i + 1)^d + \Omega$. On setting its derivative w.r.t. $\hat{\mathbf{x}}$ to zero, we obtain an iteration formula for the

[9]The USPS database can be downloaded from http://www.kernel-machines.org.

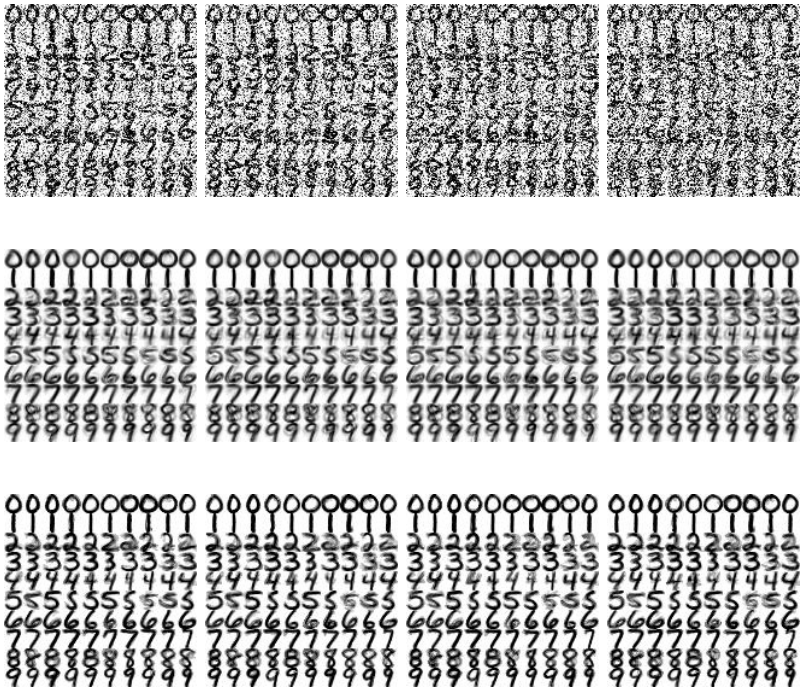

Figure 3. Typical test images corrupted by the Gaussian noise, and the corresponding denoised results with $N = 100$ and the Gaussian kernel (The noise variances for the columns from left to right are 0.2, 0.3, 0.4 and 0.5 respectively). Top: noisy images; Middle: results by (Mika et al., 1998); Bottom: results by the proposed method.

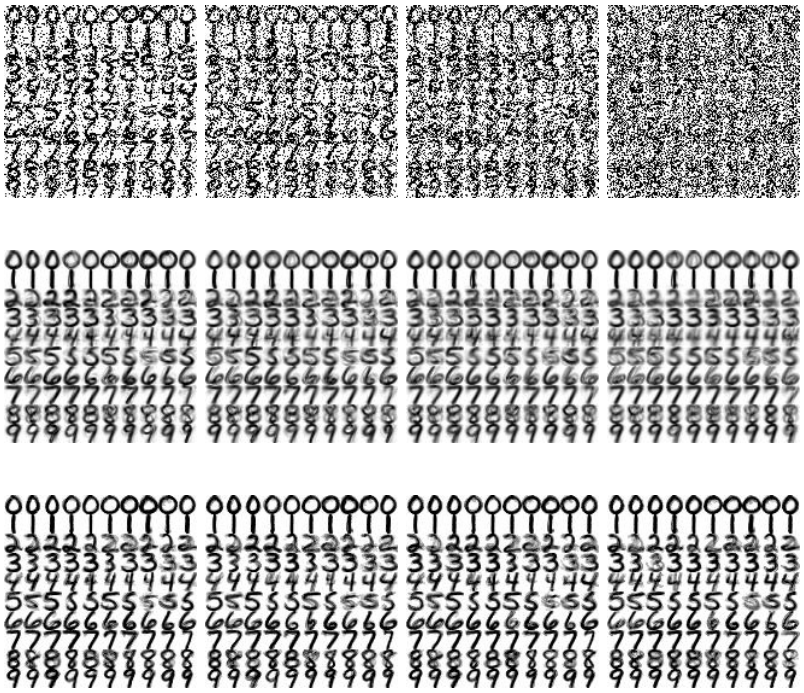

Figure 4. Typical test images corrupted by the "salt and pepper" noise, and the corresponding denoised results with $N = 100$ and the Gaussian kernel (The noise levels for the columns from left to right are 0.3, 0.4, 0.5 and 0.7 respectively). Top: noisy images; Middle: results by (Mika et al., 1998); Bottom: results by the proposed method.

Table 2. SNRs (in dB) of the denoised images using the Gaussian kernel, at different number of training samples and different noise levels (p) of the "salt and pepper" noise.

number of training images	p	SNR noisy images	SNR our method	SNR Mika *et al.*
300	0.3	1.27	6.43	5.98
	0.4	0.06	5.96	5.24
	0.5	-0.90	5.31	4.62
	0.6	-1.66	4.69	4.17
	0.7	-2.66	4.08	3.86
60	0.3	1.26	4.65	4.55
	0.4	0.24	4.45	4.24
	0.5	-0.89	4.13	3.93
	0.7	-2.99	3.52	3.48

polynomial kernel:

$$\hat{\mathbf{x}}_{t+1} = \sum_{i=1}^{N} \tilde{\gamma}_i (\frac{\hat{\mathbf{x}}_t' \mathbf{x}_i + 1}{\hat{\mathbf{x}}_t' \hat{\mathbf{x}}_t + 1})^{d-1} \mathbf{x}_i.$$

However, this iteration scheme fails to converge in the experiments, even after repeated restarts. On the other hand, our proposed method is non-iterative and can always obtain reasonable pre-images (Figure 5). Tables 3 and 4 show the resulting SNRs for the Gaussian noise and "salt and pepper" noise.

Table 3. SNRs (in dB) of the denoised images using the polynomial kernel, at different number of training samples and different noise variances (σ^2) of the Gaussian noise.

number of training images	σ^2	SNR of our method
300	0.25	5.39
	0.3	5.08
	0.4	4.61
	0.5	4.24
60	0.25	4.33
	0.3	4.09
	0.4	3.74
	0.5	3.50

4.2. Pre-Images in Kernel k-Means Clustering

In this Section, we perform the kernelized version of k-means clustering algorithm on the USPS data set, and then use the proposed method to find the pre-images of the cluster centroids. We select a total of 3000 random images from the USPS data set and the same Gaussian kernel as in Section 4.1.1. Figure 6 shows the resultant pre-images. For comparison, the cluster centroids obtained by averaging in the input space are also shown.

Table 4. SNRs (in dB) of the denoised images using the polynomial kernel, at different number of training samples and different noise levels (p) of the "salt and pepper" noise.

number of training images	p	SNR of our method
300	0.3	5.84
	0.4	5.09
	0.5	4.28
	0.6	3.56
	0.7	3.10
60	0.3	4.66
	0.4	4.08
	0.5	3.49
	0.7	2.84

(a) Using input space averaging.

(b) Using the proposed pre-image method.

Figure 6. Cluster centroids obtained on the USPS data set.

Note that, in general, these cluster centroids may have to be interpreted with caution. As in ordinary k-means clustering, the (exact or approximate) pre-images of the cluster centroids may sometimes fall outside of the data distribution.

5. Conclusion

In this paper, we address the problem of finding the pre-image of a feature vector in the kernel-induced feature space. Unlike the traditional method in (Mika et al., 1998) which relies on nonlinear optimization and is iterative in nature, our proposed method directly finds the location of the pre-image based on distance constraints. It is non-iterative, involves only linear algebra and does not suffer from numerical instabilities or the local minimum problem. Moreover, it can be applied equally well to both isotropic kernels and dot product kernels. Experimental results on denoising the USPS data set show significant improvements over

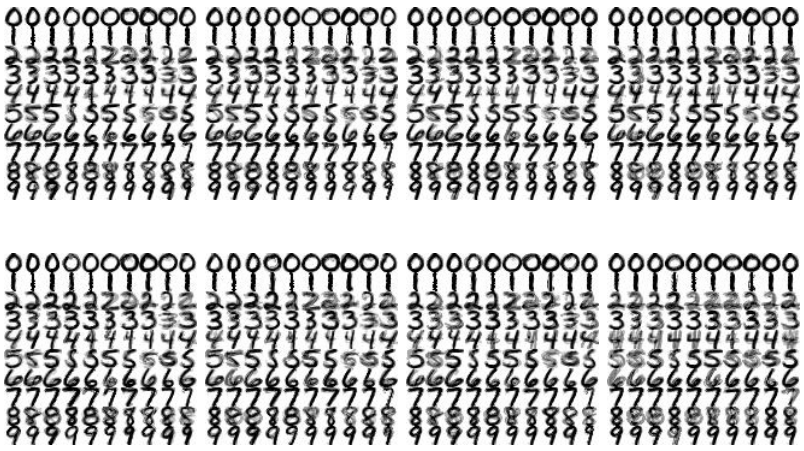

Figure 5. Denoised results using the proposed method with $N = 100$ and polynomial kernel. Top: Gaussian noise (noise variances from left to right are 0.2, 0.3, 0.4 and 0.5 respectively). Bottom: "salt and pepper" noise (noise levels from left to right are 0.3, 0.4, 0.5 and 0.7 respectively).

(Mika et al., 1998).

In the future, other classes of kernel functions will also be investigated, especially those that are defined on structured objects, in which the pre-image problem then becomes an important issue (Weston et al., 2003).

Acknowledgments

This research has been partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grants HKUST2033/00E and HKUST6195/02E.

References

Burges, C. (1996). Simplified support vector decision rules. *Proceedings of the Thirteenth International Conference on Machine Learning* (pp. 71–77). San Francisco, CA: Morgan Kaufmann.

Cox, T., & Cox, M. (2001). *Multidimensional scaling.* Monographs on Statistics and Applied Probability 88. Chapman & Hall / CRC. Second edition.

Girolami, M. (2002). Mercer kernel-based clustering in feature space. *IEEE Transactions on Neural Networks*, *13*, 780–784.

Gower, J. (1968). Adding a point to vector diagrams in multivariate analysis. *Biometrika*, *55*, 582–585.

Mika, S., Schölkopf, B., Smola, A., Müller, K., Scholz, M., & Rätsch, G. (1998). Kernel PCA and denoising in feature spaces. *Advances in Neural Information Processing Systems 11.* San Mateo, CA: Morgan Kaufmann.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, *290*, 2323–2326.

Schölkopf, B., Knirsch, P., Smola, A., & Burges, C. (1998a). Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces. *Proceedings of the DAGM Symposium Mustererkennung* (pp. 124–132).

Schölkopf, B., & Smola, A. (2002). *Learning with kernels.* MIT.

Schölkopf, B., Smola, A., & Müller, K. (1998b). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Vapnik, V. (1998). *Statistical learning theory.* New York: Wiley.

Weston, J., Chapelle, O., Elisseeff, A., Schölkopf, B., & Vapnik, V. (2003). Kernel dependency estimation. *Advances in Neural Information Processing Systems 15.* Cambridge, MA: MIT Press.

Williams, C. (2001). On a connection between kernel PCA and metric multidimensional scaling. *Advances in Neural Information Processing Systems 13.* Cambridge, MA: MIT Press.