The Pre-Image Problem in Kernel Methods

James Tin-Yau Kwok and Ivor Wai-Hung Tsang

***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 [1] 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 noniterative, involves only linear algebra and does not suffer from numerical instability or local minimum problems. Evaluations on performing kernel PCA and kernel clustering on the USPS data set show much improved performance.**

***Index Terms*—Kernel principal component analysis (PCA), multidimensional scaling (MDS), pre-image.**

I. Introduction

In RECENT years, there has been a lot of interest in the study of kernel methods [2]–[4]. 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 nonconvex 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 [5], [6], kernel independent component analysis [7], and kernel principal component analysis (PCA) [8].

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, and this can offer significant speed-ups in many kernel applications [9], [10].

However, the exact pre-image typically does not exist [1], and one can only settle for an approximate solution. But even this is nontrivial as the dimensionality of the feature space can be infinite. Schölkopf *et al.* [10] (and later in [1]) 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 [1], this method suffers from numerical 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. On the other hand, a method for computing the pre-images using only linear algebra has also been proposed [9], though it only works for polynomial kernels of degree two.

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 [11]. In this paper, we use this relationship together with the idea in multidimensional scaling (MDS) [12] to address the pre-image problem. The resultant procedure is noniterative and involves only linear algebra.

Our exposition in the sequel will focus on the pre-image problem in kernel PCA. However, this can be applied equally well to other kernel methods, such as kernel k-means clustering, as will be experimentally demonstrated in Section IV. The rest of this paper is organized as follows. Brief introduction to the kernel PCA is given in Section II. Section III then describes our proposed method. Experimental results are presented in Section IV, and the last section gives some concluding remarks. A preliminary version of this paper has appeared in [13].

II. Kernel PCA

A. PCA in the Feature Space

In this section, we give a short review on the kernel PCA. For clarity, centering of the φ-mapped patterns will be explicitly performed 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 corresponding to the kernel $k(\cdot,\cdot)$. Analogous to linear PCA, it involves the following eigen decomposition

$$\mathbf{HKH} = \mathbf{U}\Lambda\mathbf{U}'$$

Manuscript received March 21, 2003; revised March 17, 2004. This work was supported in part by the Research Grants Council, Hong Kong Special Administrative Region, under Grants HKUST2033/00E and HKUST6195/02E.

The authors are with the Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: jamesk@cs.ust.hk; ivor@cs.ust.hk).

Digital Object Identifier 10.1109/TNN.2004.837781

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

[2]In Section IV-A2, an analogous iteration formula for polynomial kernels is derived.

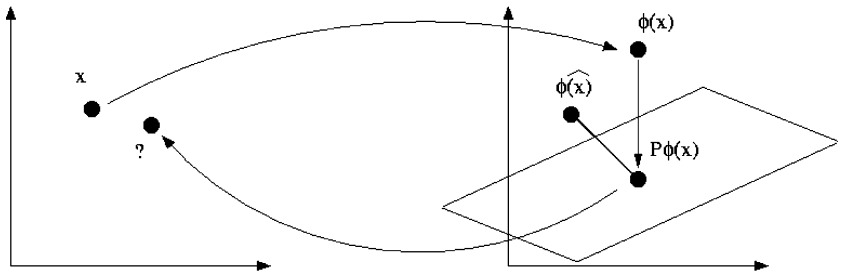

Fig. 1.Pre-image problem in kernel PCA.

where $\mathbf{K}$ is the kernel matrix with entries $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$

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

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 $\boldsymbol{\alpha}_i = [\alpha_{i1}, \ldots, \alpha_{iN}]'$ is the matrix containing the eigenvectors and $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_N)$ contains the corresponding eigenvalues. Denote the mean of the φ-mapped patterns by $\bar{\varphi} = (1/N)\sum_{i=1}^{N} \varphi(\mathbf{x}_i)$ 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 [8]

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

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 $\beta_k = (1/\sqrt{\lambda_k})\boldsymbol{\alpha}_k' \tilde{k}_{\mathbf{x}}$. Finally, the projection $P_K\varphi(\mathbf{x})$ of $\varphi(\mathbf{x})$ onto the subspace spanned by the first K eigenvectors[3] is

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

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

B. 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 (Fig. 1). As mentioned in Section I, 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.* addressed this problem by minimizing the squared distance between $\varphi(\hat{\mathbf{x}})$ and $P\varphi(\mathbf{x})$ [1]

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

where Ω includes terms independent of $\hat{\mathbf{x}}$. This, however, is a nonlinear optimization problem. As mentioned in Section I, 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:

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

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

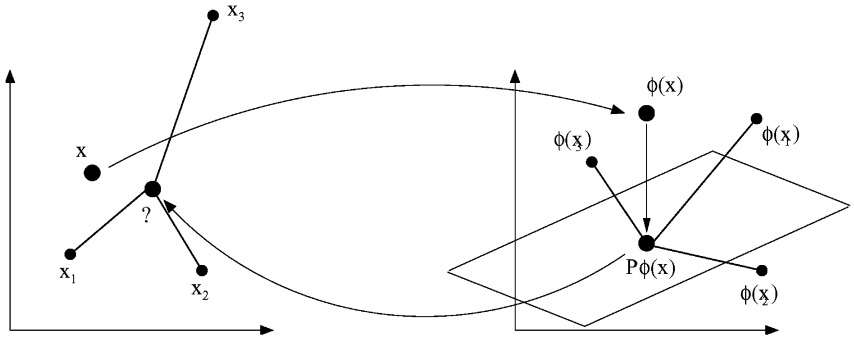

Fig. 2. Basic idea of the proposed method.

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

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 Sections III-B and C.

III. 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 $d(\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j))$ between their φ-mapped images. Moreover, for many commonly used kernels, there is a simple relationship[5] between $d(\mathbf{x}_i, \mathbf{x}_j)$ and $\tilde{d}(\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j))$ [11]. between $d(\mathbf{x}_i, \mathbf{x}_j)$ and $\tilde{d}(\varphi(\mathbf{x}_i), \varphi(\mathbf{x}_j))$ [11]. The idea of the proposed method is then as follows (Fig. 2). Let the pattern to be denoised be $\mathbf{x}$. As mentioned in Section I, 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 MDS[6] [12], 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.[7] 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).

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 IV.

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 III-A. Section III-B uses the distance relationship to obtain the corresponding distances in the input space. Section III-C uses these distances to constrain the final embedding of the pre-image.

A. 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 $\varphi(\mathbf{x}_i)$ 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

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

and

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

[4]The apparent difference with the equations in [1] is because we explicitly perform centering of the φ-mapped patterns here.

[5]An analogous relationship between the dot product in the feature space and the dot product in the input space is first pointed out in [14].

[6]Interested readers may also refer to [12] for a connection between PCA and MDS, and to [11] for a connection between kernel PCA and kernel MDS.

[7]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).

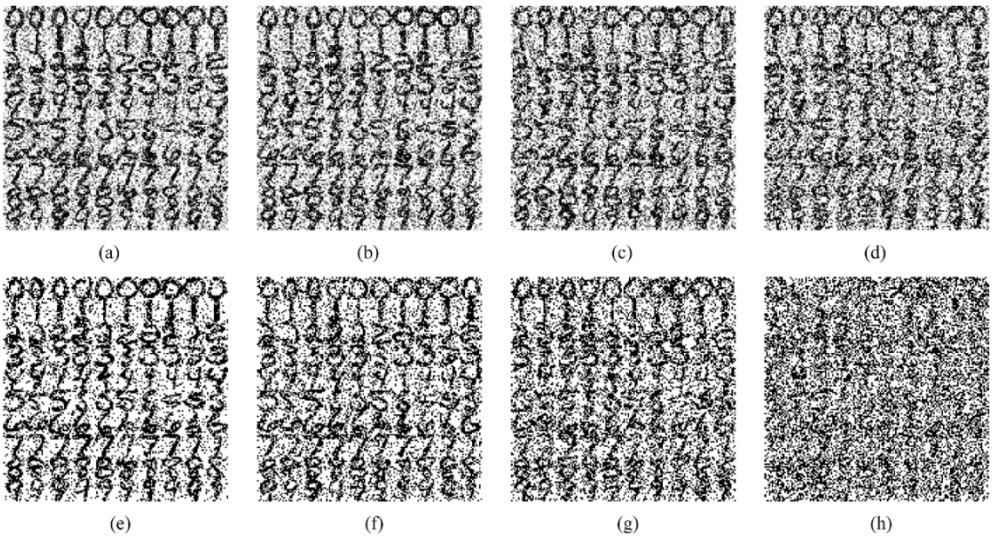

Fig. 3. Typical test images corrupted by Gaussian noise (top) and "salt and pepper" noise (bottom). (a) $\sigma^2 = 0.25$. (b) $\sigma^2 = 0.3$. (c) $\sigma^2 = 0.4$. (d) $\sigma^2 = 0.5$. (e) $p = 0.3$. (f) $p = 0.4$. (g) $p = 0.5$. (h) $p = 0.7$.

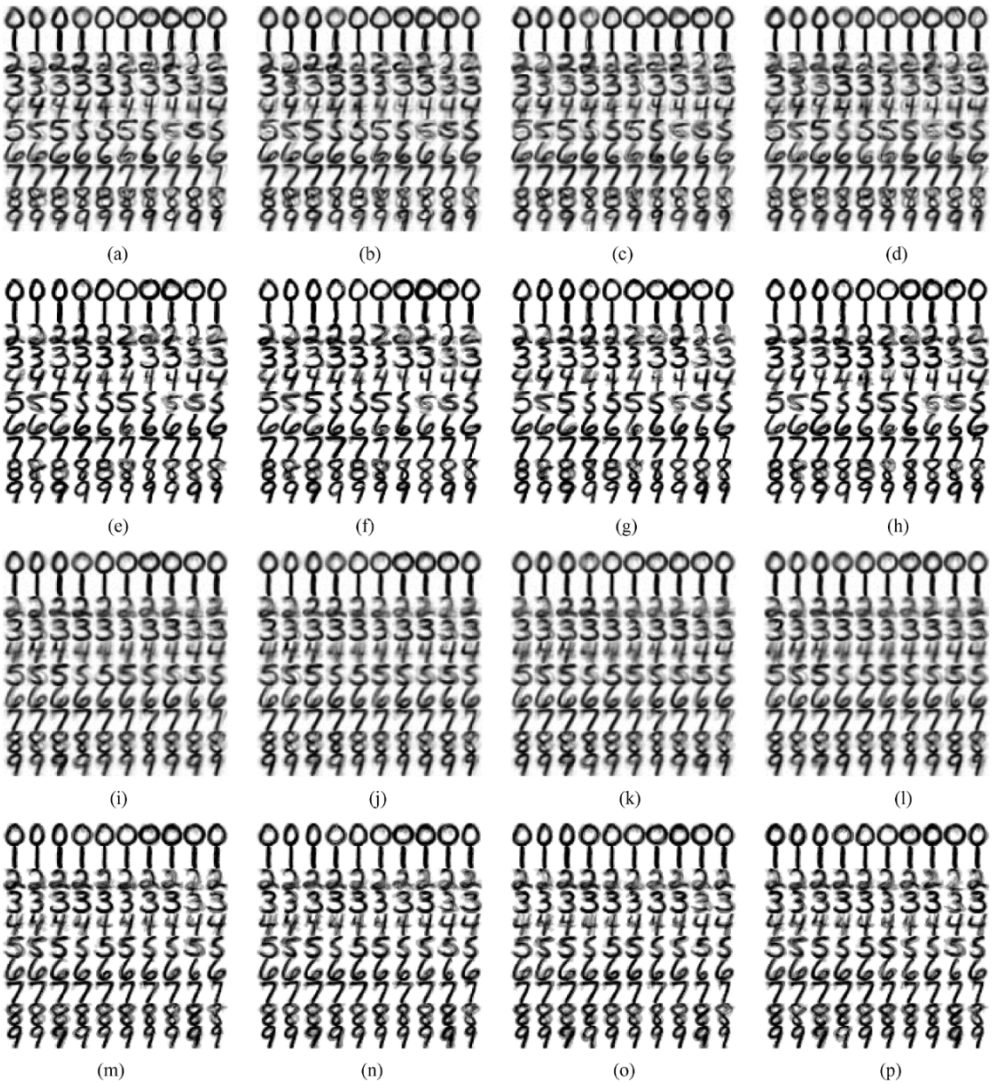

Fig. 4. Denoised results corresponding to the Gaussian kernel and the Gaussian noise. Top two rows: 300 training images. Bottom two rows: 60 training images. (a) $\sigma^2 = 0.25$, [1]. (b) $\sigma^2 = 0.3$, [1]. (c) $\sigma^2 = 0.4$, [1]. (d) $\sigma^2 = 0.5$, [1]. (e) $\sigma^2 = 0.25$, ours. (f) $\sigma^2 = 0.3$, ours. (g) $\sigma^2 = 0.4$, ours. (h) $\sigma^2 = 0.5$, ours. (i) $\sigma^2 = 0.25$, [1]. (j) $\sigma^2 = 0.3$, [1]. (k) $\sigma^2 = 0.4$, [1]. (l) $\sigma^2 = 0.5$, [1]. (m) $\sigma^2 = 0.25$, ours. (n) $\sigma^2 = 0.3$, ours. (o) $\sigma^2 = 0.4$, ours. (p) $\sigma^2 = 0.5$, ours.

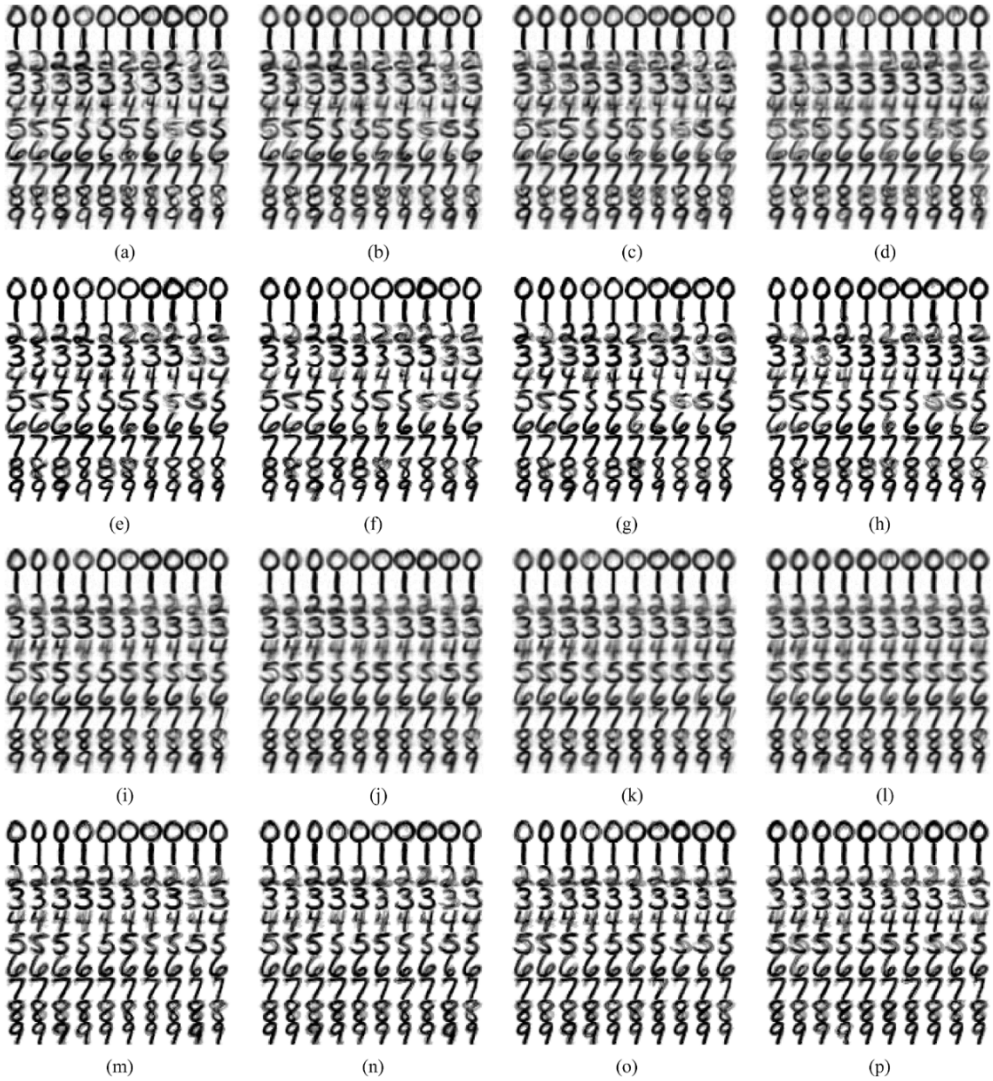

Fig. 5. Denoised results corresponding to the Gaussian kernel and the "salt and pepper" noise. Top two rows: 300 training images. Bottom two rows: 60 training images. (a) $p = 0.3$, [1]. (b) $p = 0.4$, [1]. (c) $p = 0.5$, [1]. (d) $p = 0.7$, [1]. (e) $p = 0.3$, ours. (f) $p = 0.4$, ours. (g) $p = 0.5$, ours. (h) $p = 0.7$, ours. (i) $p = 0.3$, [1]. (j) $p = 0.4$, [1]. (k) $p = 0.5$, [1]. (l) $p = 0.7$, [1]. (m) $p = 0.3$, ours. (n) $p = 0.4$, ours. (o) $p = 0.5$, ours. (p) $p = 0.7$, ours.

TABLE I
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	our method	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

TABLE II
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	our method	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

Thus, (7) becomes

$$\begin{aligned}\tilde{d}^2(P\varphi(\mathbf{x}), \varphi(\mathbf{x}_i)) &= \left(\mathbf{k}_\mathbf{x} + \frac{1}{N}\mathbf{K1} - 2\mathbf{k}_{\mathbf{x}_i}\right)' \mathbf{H}'\mathbf{M}\mathbf{H}\left(\mathbf{k}_\mathbf{x} - \frac{1}{N}\mathbf{K1}\right) \\ &\quad + \frac{1}{N^2}\mathbf{1}'\mathbf{K1} + K_{ii} - \frac{2}{N}\mathbf{1}'\mathbf{k}_{\mathbf{x}_i} \end{aligned} \tag{9}$$

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

B. Distances in the Input Space

Given the feature-space distances between $P\varphi(\mathbf{x})$ and the φ-mapped training patterns (Section III-A), 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 III-C). Recall that the distances with neighbors are the most important in determining the location of any point

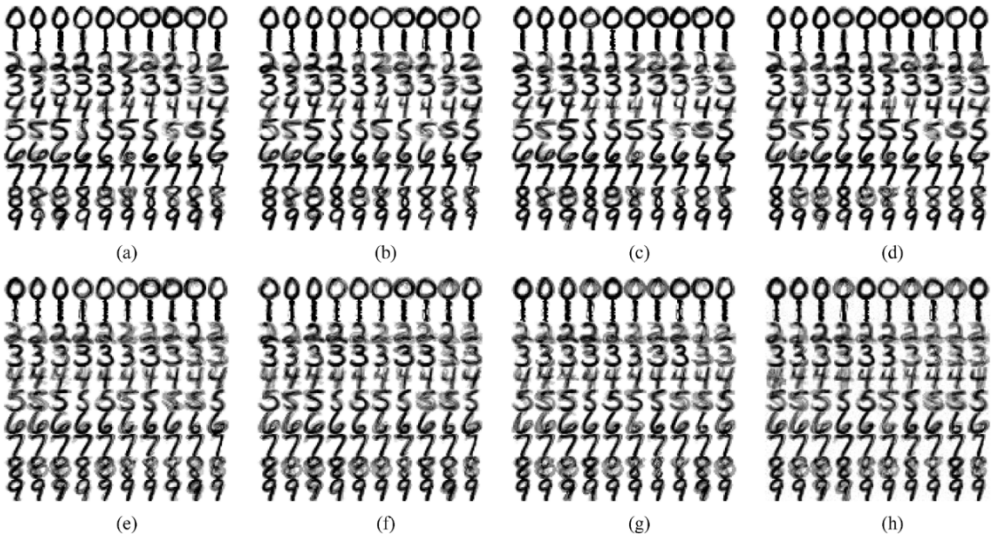

Fig. 6. Denoised results using the proposed method with the polynomial kernel (Gaussian noise). Top: 300 training images.Bottom: 60 training images. (a) $\sigma^2 = 0.25$. (b) $\sigma^2 = 0.3$. (c) $\sigma^2 = 0.4$. (d) $\sigma^2 = 0.5$. (e) $\sigma^2 = 0.25$. (f) $\sigma^2 = 0.3$. (g) $\sigma^2 = 0.4$. (h) $\sigma^2 = 0.5$.

(Section II-B). Hence, in the following, we will only consider the (squared) input-space distances between $P\varphi(\mathbf{x})$ and its n nearest neighbors,[8] i.e.,

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

This in turn can offer significant speed-up, especially during the singular value decomposition step in Section III-C. Moreover, this is also in line with the ideas in metric MDS [12], in which smaller dissimilarities are given more weight, and in locally linear embedding [15], where only the local neighborhood structure needs to be preserved.

We first consider isotropic kernels[9] 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} [11]

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

and, hence,

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

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

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 [11]

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

Moreover, κ is often invertible. For example, for the polynomial kernel $\kappa(z) = z^p$ where p is the polynomial order, $s_{ij} = K_{ij}^{(1/p)}$ 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

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

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

C. Using the Distance Constraints

For the n neighbors $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^d$ obtained in Section III-B, we will first center them at their centroid $\bar{\mathbf{x}} = (1/n)\sum_{i=1}^n \mathbf{x}_i$ 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), $\mathbf{HX}'$ will center the $\mathbf{x}_i$'s at the centroid (i.e., the column sums of $\mathbf{HX}'$ are zero). Assuming that 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

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

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 [16]. 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 II-B that the approximate pre-image $\hat{\mathbf{x}}$ obtained in [1] 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 III, its location will be obtained by

[8] 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.

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

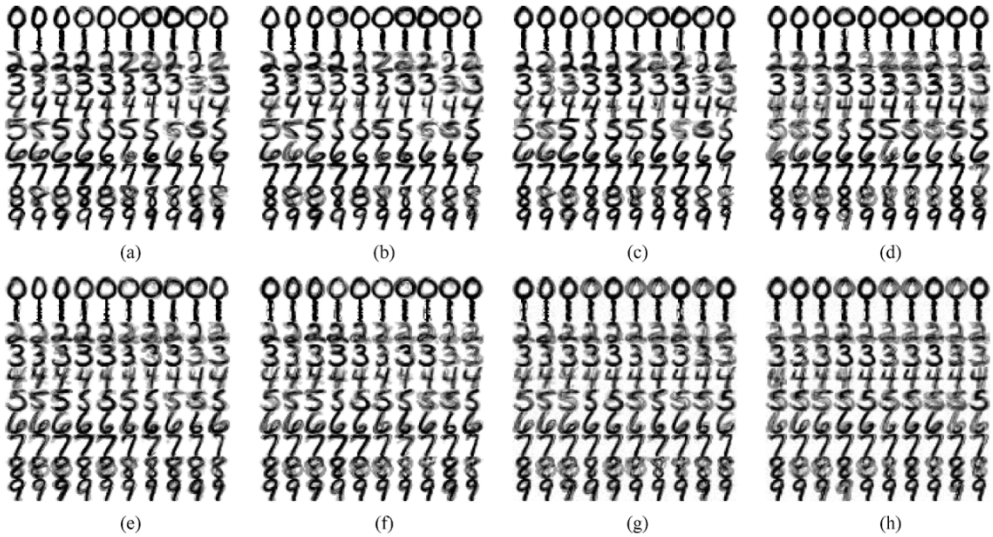

Fig. 7. Denoised results using the proposed method with the polynomial kernel ("salt and pepper" noise). Top: 300 training images. Bottom: 60 training images. (a) $p = 0.3$. (b) $p = 0.4$. (c) $p = 0.5$. (d) $p = 0.7$. (e) $p = 0.3$. (f) $p = 0.4$. (g) $p = 0.5$. (h) $p = 0.7$.

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 I, 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 [17], this can be shown to satisfy:

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

Now, $\mathbf{Z}\mathbf{1}\mathbf{1}' = \mathbf{0}$ because of the centering. Hence, the pre-image can be obtained as

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

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

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

IV. Experiment

A. Pre-Images in Kernel PCA

In this section, we report denoising results on the USPS data set consisting of 16×16 handwritten digits.[10] For each of the ten digits, we randomly choose some examples (N =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 (Fig. 3). As the model selection problem for the number (K) of eigenvectors is not the main focus of this paper, this is side-stepped by using information on the test set, and we

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

TABLE III
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 from 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

TABLE IV
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 from 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

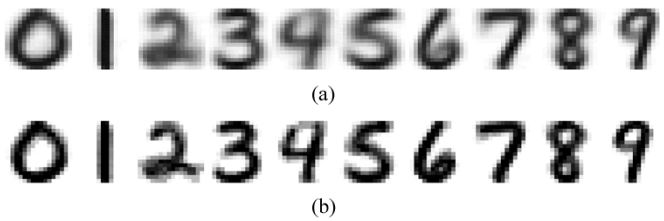

Fig. 8. Cluster centroids after performing kernel k-means clustering on the USPS data set. (a) Using input space averaging. (b) Using the proposed method.

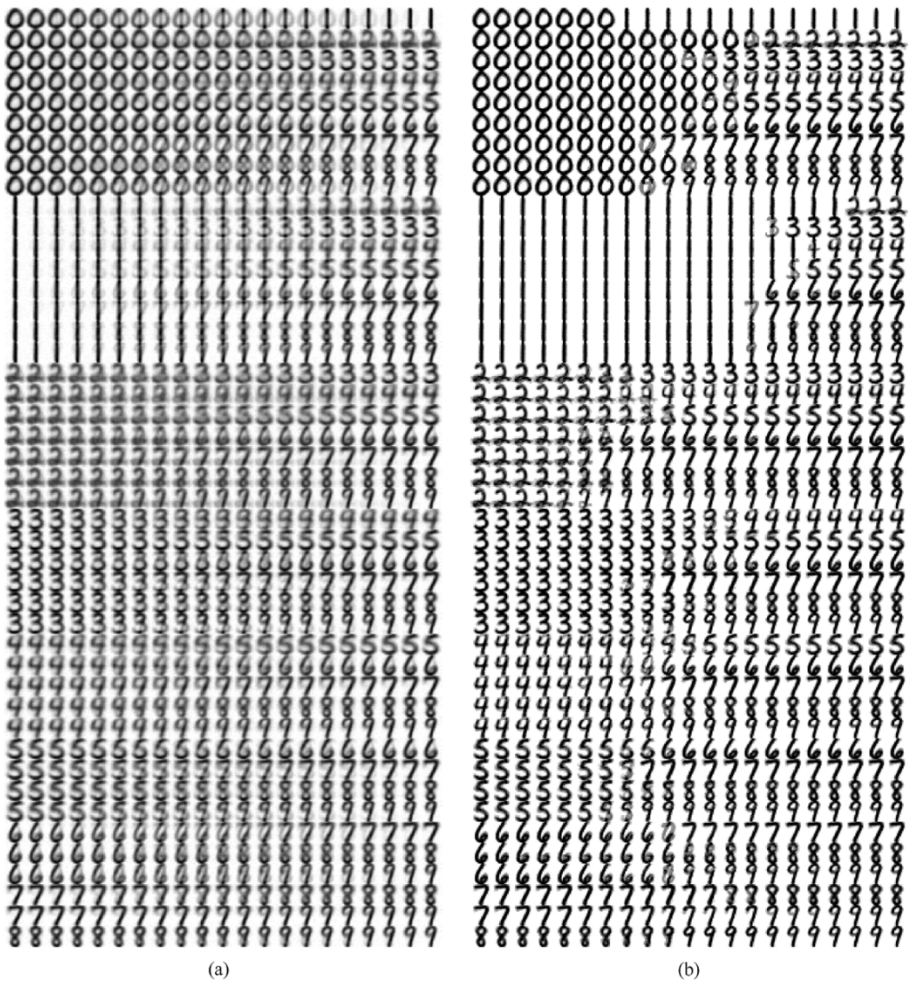

Fig. 9. Morphing results on the USPS digits. (a) Using cross fading. (b) Using the proposed pre-image method.

simply choose $K = \arg\min_n ||P_n\varphi(\mathbf{x}) - \varphi(\tilde{\mathbf{x}})||^2$, where $\mathbf{x}$ is the noisy image and $\tilde{\mathbf{x}}$ is the original (clean) image. Ten neighbors are used in locating the pre-image. Moreover, comparison will be made with the traditional method in [1].

1) Gaussian Kernel: We first experiment with the Gaussian kernel $\exp(-\beta z)$, where we set $(1/\beta) = (1)/(N(N-1))\sum_{i,j=1}^{N}||\mathbf{x}_i - \mathbf{x}_j||^2$ with $\mathbf{x}_i$'s being the training patterns. Figs. 4 and 5 show some typical denoised images based on 300 and 60 training images. Tables I and II show the corresponding numerical comparisons using the signal-to-noise ratio (SNR). As can be seen, the proposed method produces better results both visually and quantitatively.

2) Polynomial Kernel: Next, we perform experiment on the polynomial kernel $(\mathbf{xy}+1)^d$ with $d = 3$. Following the exposition of [1] in Section II-B, we develop an iteration scheme for polynomial kernels as follows. First, (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 polynomial kernel

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

However, this iteration scheme fails to converge in the experiments, even after repeated restarts. On the other hand, our proposed method is noniterative and can always obtain reasonable pre-images (Fig. 6 for the Gaussian noise and Fig. 7 for the "salt and pepper" noise), though its performance is slightly inferior to that of the Gaussian kernel. Tables III and IV show the corresponding resulting SNRs.

B. Pre-Images in Kernel k-Means Clustering

In this section, we perform the kernelized version of k-means clustering algorithm as described in [6] (the technical report version of [8]), and then use the proposed method to find pre-images of the cluster centroids. A total of 3000 images are randomly selected from the USPS data set. The same Gaussian kernel as used in Section IV-A-1 is employed, and k is set to ten, the number of digits. For comparison, cluster centroids are also obtained by averaging in the input space over all patterns belonging to the same cluster. Fig. 8 compares the resultant sets of pre-images. Clearly, the proposed method yields centroids that are visually more appealing.

Besides finding the cluster centroids, we can also visualize the transitions from one digit to another. For each pair of cluster centroids in the feature space, we draw a line connecting them and then obtain pre-images for the intermediate points uniformly spaced on this line. For comparison, we also apply the traditional method of cross fading [18] in the input space. As can be seen from Fig. 9, our proposed method again produces images that are visually better than those obtained from cross fading.

In general, note that the cluster centroids (and similarly for the intermediate morphing results) obtained 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.

V. 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 [1] 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 noniterative, 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 certain dot product kernels. Experimental results on denoising the USPS data set show significant improvements over [1].

In the future, we plan to apply this method to other kernel applications that also require computation of the pre-images. An example will be information retrieval applications with the use of kernels, in which we may want to find pre-images of the query vectors residing in the feature space [19]. Moreover, we have only addressed isotropic kernels and dot product kernels in this paper. 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 [20].

Acknowledgment

The authors would also like to thank the anonymous reviewers for their constructive comments on an earlier version of this paper.

References

[1] S. Mika, B. Schölkopf, A. Smola, K. R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de-noising in feature spaces," in *Advances in Neural Information Processing Systems 11*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds. San Mateo, CA: Morgan Kaufmann, 1998.

[2] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

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

[4] V. Vapnik, *Statistical Learning Theory*. New York: Wiley, 1998.

[5] M. Girolami, "Mercer kernel-based clustering in feature space," *IEEE Trans. Neural Networks*, vol. 13, pp. 780–784, May 2002.

[6] B. Schölkopf, A. Smola, and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," Max-Planck-Institut für biologische Kybernetik, Tech. Rep. 44, 1996.

[7] F. R. Bach and M. I. Jordan, "Kernel independent component analysis," Dept. Comput. Sci., Univ. California, Tech. Rep. UCB/CSD-01-1166, 2001.

[8] B. Schölkopf, A. Smola, and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computat.*, vol. 10, pp. 1299–1319, 1998.

[9] C. J. C. Burges, "Simplified support vector decision rules," in *Proceedings of the Thirteenth International Conference on Machine Learning*. San Francisco, CA, 1996, pp. 71–77.

[10] B. Schölkopf, P. Knirsch, A. J. Smola, and C. Burges, "Fast approximation of support vector kernel expansions, and an interpretation of clustering as approximation in feature spaces," in *Proc. DAGM Symp. Mustererkennung*, 1998, pp. 124–132.

[11] C. K. I. Williams, "On a connection between kernel PCA and metric multidimensional scaling," in *Advances in Neural Information Processing Systems 13*, T. Leen, T. Dietterich, and V. Tresp, Eds. Cambridge, MA: MIT Press, 2001.

[12] T. F. Cox and M. A. A. Cox, *Multidimensional Scaling, Monographs on Statistics and Applied Probability 88*, 2nd ed. London, U.K.: Chapman & Hall, 2001.

[13] J. T. Kwok and I. W. Tsang, "The pre-image problem in kernel methods," in *Proc. 20th Int. Conf. Machine Learning*, Washington, D.C., Aug. 2003, pp. 408–415.

[14] B. Schölkopf, "Support vector learning," Ph.D. dissertation, Tech. Univ. Berlin, Berlin, Germany, 1997.

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

[16] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.

[17] J. C. Gower, "Adding a point to vector diagrams in multivariate analysis," *Biometrika*, vol. 55, pp. 582–585, 1968.

[18] G. Wolberg, *Digital Image Warping*. Los Alamitos, CA: IEEE Comput. Soc. Press, 1990.

[19] J. Peng, private communication, 2002 Personal communications.

[20] J. Weston, O. Chapelle, A. Elisseeff, B. Schölkopf, and V. Vapnik, "Kernel dependency estimation," in *Advances in Neural Information Processing Systems 15*, S. Becker, S. Thrun, and K. Obermayer, Eds. Cambridge, MA: MIT Press, 2003.

James Tin-Yau Kwok received the Ph.D. degree in computer science from The Hong Kong University of Science and Technology (HKUST), Clear Water Bay, in 1996.

He then joined the Department of Computer Science, Hong Kong Baptist University, as an Assistant Professor. He returned to the HKUST in 2000 and is now an Assistant Professor in the Department of Computer Science. His research interests include kernel methods, machine learning, pattern recognition, and artificial neural networks.

Ivor Wai-Hung Tsang received the B.Eng. and M.Phil. degrees in computer science from the Hong Kong University of Science and Technology (HKUST), Clear Water Bay, in 2001 and 2003, respectively. He is currently working toward the Ph.D. degree at HKUST.

His scientific interests includes machine learning and kernel methods.