J Sign Process Syst (2011) 65:289–299
DOI 10.1007/s11265-010-0482-9

# A Closed-form Solution for the Pre-image Problem in Kernel-based Machines

**Paul Honeine · Cédric Richard**

Received: 15 January 2010 / Revised: 15 January 2010 / Accepted: 29 March 2010 / Published online: 17 April 2010


**Abstract** The pre-image problem is a challenging research subject pursued by many researchers in machine learning. Kernel-based machines seek some relevant feature in a reproducing kernel Hilbert space (RKHS), optimized in a given sense, such as kernel-PCA algorithms. Operating the latter for denoising requires solving the pre-image problem, i.e. estimating a pattern in the input space whose image in the RKHS is approximately a given feature. Solving the pre-image problem is pioneered by Mika's fixed-point iterative optimization technique. Recent approaches take advantage of prior knowledge provided by the training data, whose coordinates are known in the input space and implicitly in the RKHS, a first step in this direction made by Kwok's algorithm based on multidimensional scaling (MDS). Using such prior knowledge, we propose in this paper a new technique to learn the pre-image, with the elegance that only linear algebra is involved. This is achieved by establishing a coordinate system in the RKHS with an isometry with the input space, i.e. the inner products of training data are preserved using both representations. We suggest representing any feature in this coordinate system, which gives us information regarding its pre-image in the input space. We show that this approach provides a natural pre-image technique in kernel-based machines since, on one hand it involves only linear algebra operations, and on the other it can be written directly using the kernel values, without the need to evaluate distances as with the MDS approach. The performance of the proposed approach is illustrated for denoising with kernel-PCA, and compared to state-of-the-art methods on both synthetic datasets and realdata handwritten digits.

**Keywords** Kernel-based machines · Pre-image problem · Linear algebra · Kernel-PCA · Nonlinear denoising

## 1 Introduction

In the last decade or so, kernel-based machines have enjoyed increasing popularity, providing a breakthrough in both statistical learning theory and low computational complexity of nonlinear algorithms. Pioneered by Vapnik's Support Vector Machines (SVM) [20], this concept attracted significant attention due to the ever-expanding challenges in machine learning. Since then, many nonlinear algorithms have been developed, for supervised learning (or classification) such as kernel Fisher discriminant analysis [13] and least-squares SVM [18], and for unsupervised learning (with unlabelled data) with kernel principal component analysis (kernel-PCA) [17] and support vector domain description [19]. The main idea behind nonlinear algorithms in kernel-based machines is the *kernel trick* [1]. This concept gives rise to nonlinear algorithms based

This is an extended version of the paper [8], winner of the Best Paper Award at the IEEE Machine Learning For Signal Processing workshop.

P. Honeine (✉)
Institut Charles Delaunay (FRE CNRS 2848), LM2S, Université de Technologie de Troyes, 10010 Troyes, France
e-mail: paul.honeine@utt.fr

C. Richard
Laboratoire Fizeau (UMR CNRS 6525), Observatoire de la Côte d'Azur, Université de Nice Sophia-Antipolis, 06108 Nice, France
e-mail: cedric.richard@unice.fr

on classical linear ones, under the only requirement that the algorithm can be expressed only in terms of inner products between data. Then, data from the input space are (nonlinearly) mapped into a feature space. This mapping is achieved implicitly by substituting the inner product operator by a positive definite kernel, thus without much additional computational cost. This is the essence of the kernel trick. In order to provide the unified functional framework, common to many communities, this kernel is called the *reproducing kernel* while the induced feature space is the *reproducing kernel Hilbert space* (RKHS).

With the ever-increasing demands in machine learning, new challenges require computing the inverse map. For instance, while the kernel trick provides an elegant approach to apply denoising or compression techniques in the RKHS, we need to go back into the input space for the final result. This is the case in denoising an image (or a signal), the reconstructed image belongs to the input space of training images. However, getting back to the input space from the RKHS is not obvious in general, as most features of the latter may not have an exact pre-image in the former. This is the pre-image problem in kernel-based machines, as one seeks an approximate solution. Solving this problem has received a growing amount of attention, with the most breakthrough given in [14] and [11]. In the former work, Mika et al. present the problem and its ill-posedness, and derive a fixed-point iterative scheme to find an approximate solution. Hence, there is no guarantee that this leads to a global optimum, and may be unstable. In the latter work, Kwok et al. determine a relationship between the distances in the RKHS and the distances in the input data, based on a set of training data. Applying a multidimensional scaling technique (MDS) leads to an inverse map estimate and thus to the pre-image. This approach opens the door to a range of other techniques taking prior knowledge from training data in both spaces, such as manifold learning [6] and out-of-sample methods [2, 5].

In this paper, we propose a novel approach to solve the pre-image problem. To achieve this, we learn a coordinate system, not necessarily orthogonal, in the RKHS having an isometry with the input space. In other words, the inner products of the training data are (approximately) equal in both representations. Thus, by representing any feature of the RKHS in this coordinate system, we get an estimate of the inner products between the training data and its counterpart in the input space. It turns out that this approach is natural to kernel-based machines, and essentially requires only linear algebra, with any off-the-shelf linear solver. The proposed method is universal in the sense of being independent, in its formulation, of both the type of the adopted kernel and of the feature under investigation. Moreover, it extends naturally to get the pre-images of a set of features, since the coordinate system is computed only once.

The rest of the paper is organized as follows. In the next Section, we briefly present the framework behind kernel-based machines, with an illustration on kernel-PCA for denoising. In Section 3, the pre-image problem is described, and previous work on solving the problem are examined. The proposed method is described in Section 4, with connections to other methods. Experiments on synthetic and real datasets are presented in Section 5. Section 6 ends this paper with a brief conclusion.

## 2 Kernel-based Machines, with Application to Nonlinear Denoising Using Kernel-PCA

### 2.1 Kernel-based Machines

Let $\mathcal{X}$ be a compact of $\mathbb{R}^p$, equipped with the natural Euclidean inner product defined for any $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$ by $\boldsymbol{x}_i^\top \boldsymbol{x}_j = \sum_{\ell=1}^{p} \boldsymbol{x}_{i,\ell} \boldsymbol{x}_{j,\ell}$, with $\boldsymbol{x}_{\cdot,\ell}$ the $\ell$-th entry of vector $\boldsymbol{x}_\cdot$. Let $\kappa(\cdot,\cdot)$ be a positive (semi-)definite kernel on $\mathcal{X} \times \mathcal{X}$, where the positive (semi-)definiteness is defined by the property

$$\sum_{i,j} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

for all $\alpha_i, \alpha_j \in \mathbb{R}$ and $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$. The Moore-Aronszajn theorem [3] states that for every positive definite kernel, there exists a unique reproducing kernel Hilbert space (RKHS), and vice versa. With this one-to-one correspondence between RKHS and positive definite kernels, the latter will be called reproducing kernels hereafter. Let $\mathcal{H}$ be the RKHS associated with $\kappa$, and let $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ be the endowed inner product. This means that any arbitrary function $\psi(\cdot)$ in $\mathcal{H}$ can be evaluated at any $\boldsymbol{x}_j \in \mathcal{X}$ with

$$\psi(\boldsymbol{x}_j) = \langle \psi(\cdot), \kappa(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}}. \tag{1}$$

This expression shows that the kernel is the representer of evaluation. Moreover, replacing in this expression $\psi(\cdot)$ by $\kappa(\cdot, \boldsymbol{x}_i)$ yields to the popular property

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \kappa(\cdot, \boldsymbol{x}_i), \kappa(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}}, \tag{2}$$

for all $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$. This is the reproducing property from which the name of reproducing kernel is derived. Denoting by $\phi(\cdot)$ the map that assigns to each input $\boldsymbol{x} \in \mathcal{X}$ the kernel function $\kappa(\cdot, \boldsymbol{x})$, the reproducing property

**Table 1** Commonly used reproducing kernels in machine learning, with parameters $\beta > 0$, $q \in \mathbb{N}$, and $\sigma > 0$.

| | |
|---|---|
| Polynomial | $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle + \beta)^q$ |
| Laplace | $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|/\sigma\right)$ |
| Gaussian | $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2\right)$ |

(2) implies that $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}$. The kernel then evaluates the inner product of any pair of elements of $\mathcal{X}$ mapped into $\mathcal{H}$, without any explicit knowledge of either the mapping function $\phi(\cdot)$ or the RKHS $\mathcal{H}$. This is the well-known kernel trick. Examples of commonly used reproducing kernels are given in Table 1.

In combination with the kernel trick, the representer theorem provides a powerful theoretical foundation for kernel-based machines. Initially derived in [10] and recently generalized in [16], results of this theorem include SVM and kernel-PCA, where one seeks to maximize the separating margin between classes or the variance of projected data, respectively. This theorem states that any function $\varphi^*(\cdot)$ of a RKHS $\mathcal{H}$ minimizing a regularized cost functional of the form

$$\sum_{i=1}^{n} J(\varphi(\boldsymbol{x}_i), y_i) + g(\|\varphi\|_{\mathcal{H}}^2),$$

with predicted output $\varphi(\boldsymbol{x}_i)$ for input $\boldsymbol{x}_i$, and eventually the desired output $y_i$, and $g(\cdot)$ a strictly monotonically increasing function on $\mathbb{R}_+$, can be written as a kernel expansion in terms of available data

$$\varphi^*(\cdot) = \sum_{i=1}^{n} \gamma_i \, \kappa(\cdot, \boldsymbol{x}_i). \tag{3}$$

This theorem shows that even in an infinite dimensional RKHS, as with the Gaussian kernel, we only need to work in the subspace spanned by the $n$ kernel functions of the training data, $\kappa(\cdot, \boldsymbol{x}_1), \ldots, \kappa(\cdot, \boldsymbol{x}_n)$.

## 2.2 Kernel-PCA for Denoising

An elegant kernel-based machine is the kernel-PCA [17], a nonlinear extension of one of the most used dimension reduction and denoising technique, the principal component analysis (PCA).

With PCA, one seeks principal axes that capture the highest variance in the data, that is, useful information as opposed to noise, and thus projecting data onto the space spanned by these relevant axes yields a denoising scheme. These principal axes are the eigenvectors $\varphi_k$ associated with the largest eigenvalues $\lambda_k$ of the covariance matrix $\boldsymbol{R}$ of data, i.e. solving the eigen-problem $\boldsymbol{R}\varphi_k = \lambda_k \varphi_k$. There exists another formulation of the PCA algorithm, using only inner products of the training data. By substituting the inner product operator with any valid reproducing kernel, we get an implicit nonlinear mapping of the data into a RKHS. This is the kernel-PCA, where each of the resulting principal functions takes the representer form (3), with

$$\varphi_k^*(\cdot) = \sum_{i=1}^{n} \gamma_{i,k} \, \kappa(\cdot, \boldsymbol{x}_i).$$

The weighting coefficients $\gamma_{i,k}$ are obtained from the eigen-decomposition of the so-called Gram matrix $\boldsymbol{K}$, whose entries are $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, for $i, j = 1, \ldots, n$, by solving

$$\boldsymbol{K}\boldsymbol{\gamma} = n\lambda \, \boldsymbol{\gamma}.$$

In order to have a PCA interpretation in feature space, two issues should be carried out. First, data is implicitly centered in feature space by substituting in this expression $\boldsymbol{K}$ with $(\mathbf{1} - \mathbf{1}_n)\boldsymbol{K}(\mathbf{1} - \mathbf{1}_n)$, with $\mathbf{1}_n$ the $n$-by-$n$ matrix of entries $1/n$ and $\mathbf{1}$ the identity matrix; second, principal functions are normalized to 1, by scaling expansion coefficients such that $\sum_{i=1}^{n} \gamma_{i,k}^2 = 1/\lambda_k$.

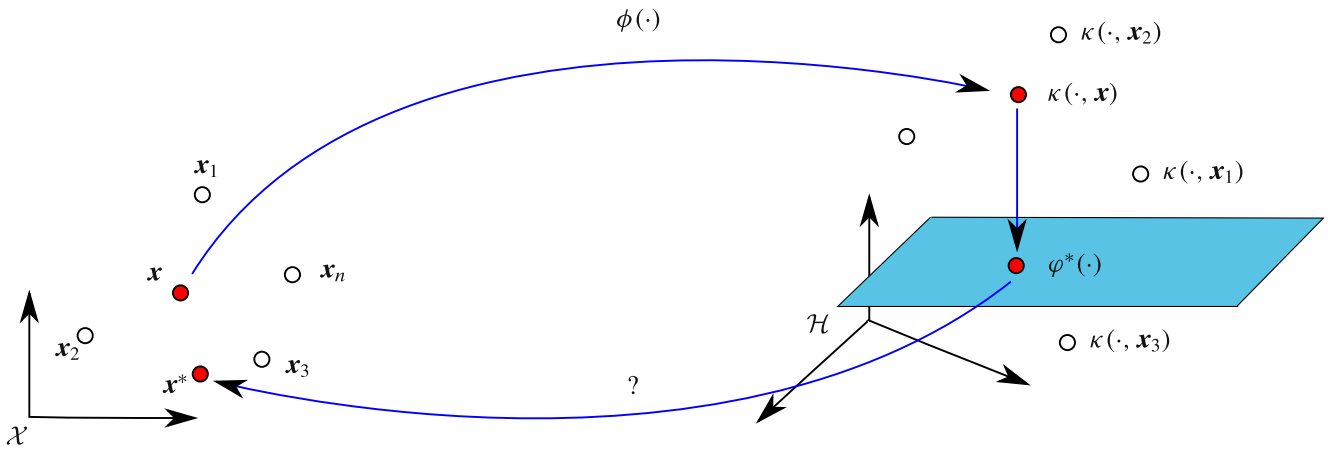


**Figure 1** Illustration of the pre-image problem in kernel-based machines.

In the same spirit of the conventional PCA, one constructs a subspace of $\mathcal{H}$ spanned by the most relevant principal functions. Using kernel-PCA for denoising any given $\boldsymbol{x} \in \mathcal{X}$, we project the associated kernel function $\kappa(\cdot, \boldsymbol{x})$ onto that subspace. Since this subspace is spanned by most relevant principal functions, each of the form (3), any function from this subspace takes the same form, i.e. can be written as a kernel expansion in terms of available data. Let $\varphi^*(\cdot)$ be this projection, with

$$\varphi^*(\cdot) = \sum_{i=1}^{n} \gamma_i \kappa(\cdot, \boldsymbol{x}_i),$$

which is assumed to be noise-free by virtue of the PCA interpretation. From this denoised feature, we need to get its counterpart in the input space, e.g. a denoised image in the image space. As illustrated in Fig. 1, this requires the estimation of the pattern $\boldsymbol{x}^*$ from $\kappa(\cdot, \boldsymbol{x})$, by solving the pre-image problem.

## 3 A Brief Review of the Pre-image Problem

For supervised learning, one seeks a prediction value associated to any input such as in regression problems, while in classification this value is compared to a threshold, which yields a decision rule. While every optimal function $\varphi^*(\cdot)$ takes the form (3), we obtain its evaluation at any $\boldsymbol{x}$ with $\sum_{i=1}^{n} \gamma_i \kappa(\boldsymbol{x}_i, \boldsymbol{x})$, thus requiring only computing values of the kernel. For pattern recognition with unsupervised learning, one is often interested in the feature in the feature space, or more precisely in its counterpart in the input space.

Estimating the input whose map is an arbitrary function in the RKHS is an ill-posed problem. To show this, recall that the dimensionality of the feature space can be very high, and even infinite with some kernels such as the Gaussian kernel. Thus, (most) features $\varphi^*(\cdot) \in \mathcal{H}$ might not have an existing pre-image in $\mathcal{X}$, i.e. a $\boldsymbol{x}^*$ such that $\kappa(\cdot, \boldsymbol{x}^*) = \varphi^*(\cdot)$. In order to circumvent this difficulty, one seeks an approximate solution, i.e. $\boldsymbol{x}^* \in \mathcal{X}$ whose map $\kappa(\cdot, \boldsymbol{x}^*)$ is as close as possible to $\varphi^*(\cdot)$. This is the pre-image problem in kernel-based machines. Methods for solving the pre-image problem are roughly classified into two categories: Fixed-point iterative methods and methods based on learning the *inverse map*.

The pre-image problem was initially studied by Mika et al. in [14]. They proposed to solve the optimization problem

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x} \in \mathcal{X}} \|\varphi^*(\cdot) - \kappa(\cdot, \boldsymbol{x})\|_{\mathcal{H}}^2, \tag{4}$$

where $\|\cdot\|_{\mathcal{H}}$ denotes the norm in the RKHS. For this purpose, a fixed-point iterative scheme is used to solve the pre-image problem. However, since this optimization problem is highly non-convex, such iterative technique suffers from numerical instabilities and local minima. The pre-image will highly depend on the initial guess and is likely to get stuck in a local minimum. A further improvement of the fixed-point iterative scheme is presented in [15], where the authors operate additional approximations by, roughly speaking, substituting the mapping $\kappa(\cdot, \boldsymbol{x})$ in (4) with its projection onto the subspace. It is worth noting that as an alternative to Mika's distance minimization, one may consider a collinearity maximization problem [2], with

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \mathcal{X}} \left\langle \frac{\varphi^*(\cdot)}{\|\varphi^*(\cdot)\|_{\mathcal{H}}}, \frac{\kappa(\cdot, \boldsymbol{x})}{\|\kappa(\cdot, \boldsymbol{x})\|_{\mathcal{H}}} \right\rangle_{\mathcal{H}}.$$

A key ingredient of these methods is a high dependence on the kernel type, since the fixed-point can only be applied to some specific kernels such as the Gaussian kernel, and only more recently extended to polynomial kernels in [11].

Recent approaches take advantage of prior knowledge provided by some available training data, whose *coordinates* are available in both the input and the feature spaces. This approach is initiated by an algorithm based on multidimensional scaling (MDS), presented by Kwok et al. [11]. This is achieved by computing distances between every pair of training data, in both spaces. For each pair, the classical Euclidean distance is used in the input space, as well as the distance in the RKHS which can be computed using kernel values. With these pairs of distances, a MDS technique is considered by performing a singular-value-decomposition.[1] This yields an *inverse map*, in the same spirit of the out-of-sample extension [2]. In order to make this method tractable in practice, only the neighboring data affect the pre-image estimation. Learning the inverse map is studied in [4] by solving a regression problem, while alternative approaches can be based on the manifold learning [6]. All these methods take advantage of prior knowledge, i.e. training data with information available in both input and feature spaces.

Using such prior knowledge, we propose in this paper to learn the inverse map without the need to

[1] This is done by operating on the distances, transforming them into inner products, and then apply eigen-decomposition into the resulting Gram matrix to get the coordinates. This nicely captures our guiding intuition of the problem in contrast with the MDS: we propose to work exclusively on the inner products, without the need to compute distances.

compute distances, and does not require sophisticated optimization schemes. Only conventional linear algebra are needed. Furthermore, it is universe, in the sense that it does not depend on the kernel type, as opposed to fixed-point iterative techniques.

## 4 The Proposed Pre-image Method

Given a set of training data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, we seek the pre-image in $\mathcal{X}$ of some arbitrary $\varphi^*(\cdot)$ of the RKHS $\mathcal{H}$, denoted $\boldsymbol{x}^*$. The proposed method can be organized into two stages: learning the inverse map and operating a pre-image. To learn the inverse map, a coordinate system is constructed in the RKHS, having an isometry with the input space coordinates, where the isometry is defined with respect to the training data. In order to operate a pre-image, we represent $\varphi^*(\cdot)$ in this coordinate system which, by virtue of the isometry, gives the values of the inner products of its pre-image with the training data in the input space. From these values we obtain the pre-image $\boldsymbol{x}^*$.

### 4.1 Stage 1: Learn the Inverse Map

In this stage, we provide a coordinate system in the RKHS that is isometric with the input space. In order to achieve such isometry, we consider a set of $n$ training data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\} \in \mathcal{X}$. By virtue of the representer theorem, we only need to consider the subspace spanned by their kernel functions $\{\kappa(\cdot, \boldsymbol{x}_1), \kappa(\cdot, \boldsymbol{x}_2), \ldots, \kappa(\cdot, \boldsymbol{x}_n)\}$. Within this subspace, we define the set of $\ell$ coordinate functions, denoted $\{\psi_1(\cdot), \psi_2(\cdot), \ldots, \psi_\ell(\cdot)\}$ with $\ell \leq n$, and write

$$\psi_k(\cdot) = \sum_{i=1}^{n} \alpha_{k,i}\, \kappa(\cdot, \boldsymbol{x}_i),$$

for $k = 1, 2, \ldots, \ell$. For any kernel function $\kappa(\cdot, \boldsymbol{x})$, its coordinate on $\psi_k(\cdot)$ is given by

$$\langle \psi_k(\cdot), \kappa(\cdot, \boldsymbol{x}) \rangle_{\mathcal{H}} = \psi_k(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_{k,i}\, \kappa(\boldsymbol{x}_i, \boldsymbol{x}),$$

where (1) is used. Therefore, its representation in this coordinate system is obtained by the $\ell$ coordinates, written vector-wise as

$$\Psi_{\boldsymbol{x}} = [\psi_1(\boldsymbol{x}) \;\; \psi_2(\boldsymbol{x}) \;\; \cdots \;\; \psi_\ell(\boldsymbol{x})]^\top,$$

where the $k$-th entry depends on the $\alpha_{k,i}$, for $i = 1, \ldots, n$.

In order to estimate the coordinate functions, we propose an equivalence, between the inner products in this coordinate system and their counterparts in the canonic input space, using the model

$$\Psi_{\boldsymbol{x}_i}^\top \Psi_{\boldsymbol{x}_j} = \boldsymbol{x}_i^\top \boldsymbol{x}_j + \epsilon_{ij}, \tag{5}$$

for all the training set, i.e. $i, j = 1, 2, \ldots, n$, and where $\epsilon_{ij}$ corresponds to the lack-of-fit of the model. We insist on the fact that this model is not coupled with any constraint on the coordinate functions, as opposed to the orthogonality between the functions resulting from the kernel-PCA. The only requirement we impose is the isometry defined in (5). The minimization of the variance of $\epsilon_{ij}$, a lack-of-fit criterion, consists of solving the optimization problem

$$\min_{\psi_1, \ldots, \psi_\ell} \frac{1}{2} \sum_{i,j=1}^{n} \left( \boldsymbol{x}_i^\top \boldsymbol{x}_j - \Psi_{\boldsymbol{x}_i}^\top \Psi_{\boldsymbol{x}_j} \right)^2 + \lambda\, R(\psi_1, \ldots, \psi_\ell).$$

As suggested in machine learning literature, we include in this expression a regularization term, where $\lambda$ a tunable parameter controlling the tradeoff between the fitness to the model (5) and the smoothness of the solution. In order to penalize high norm functions, the regularization $R(\psi_1, \ldots, \psi_\ell) = \sum_{k=1}^{\ell} \|\psi_k\|_{\mathcal{H}}^2$ is used in this paper.

This optimization problem can be written in matrix form. This is done by a factorization of $\Psi_{\boldsymbol{x}}$ into a matrix of unknowns and a vector of available information, with

$$\Psi_{\boldsymbol{x}} = \boldsymbol{A}\, \boldsymbol{\kappa}_{\boldsymbol{x}},$$

where $\boldsymbol{\kappa}_{\boldsymbol{x}} = [\kappa(\boldsymbol{x}_1, \boldsymbol{x}) \;\; \kappa(\boldsymbol{x}_2, \boldsymbol{x}) \;\; \cdots \;\; \kappa(\boldsymbol{x}_n, \boldsymbol{x})]^\top$ and $\boldsymbol{A}$ is a $\ell \times n$ matrix of unknowns whose $(k, i)$-th entry is $\alpha_{k,i}$. This leads to the optimization problem

$$\widehat{\boldsymbol{A}} = \arg\min_{\boldsymbol{A}} \frac{1}{2} \sum_{i,j=1}^{n} \left( \boldsymbol{x}_i^\top \boldsymbol{x}_j - \boldsymbol{\kappa}_{\boldsymbol{x}_i}^\top \boldsymbol{A}^\top \boldsymbol{A}\, \boldsymbol{\kappa}_{\boldsymbol{x}_j} \right)^2 + \lambda \sum_{k=1}^{\ell} \sum_{i,j=1}^{n} \alpha_{k,i} \alpha_{k,j}\, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

By denoting $\|\cdot\|_F$ the Frobenius norm[2] of a matrix and $\mathrm{tr}(\cdot)$ its trace, this yields

$$\widehat{\boldsymbol{A}} = \arg\min_{\boldsymbol{A}} \frac{1}{2} \|\boldsymbol{P} - \boldsymbol{K}\, \boldsymbol{A}^\top \boldsymbol{A}\, \boldsymbol{K}\|_F^2 + \lambda\, \mathrm{tr}(\boldsymbol{A}^\top \boldsymbol{A} \boldsymbol{K}),$$

where $\boldsymbol{P}$and $\boldsymbol{K}$ are the Gram matrices with entries $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ and $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$, respectively. Taking the derivative of the above cost function with respect to $\boldsymbol{A}^\top \boldsymbol{A}$, rather than $\boldsymbol{A}$, and setting it to zero, we get

$$\widehat{\boldsymbol{A}}^\top \widehat{\boldsymbol{A}} = \boldsymbol{K}^{-1} \left( \boldsymbol{P} - \lambda\, \boldsymbol{K}^{-1} \right) \boldsymbol{K}^{-1}. \tag{6}$$

[2]The Frobenius norm of a matrix is the root of sum of squared (absolute) values of all its elements, or equivalently $\|\boldsymbol{M}\|_F^2 = \mathrm{tr}(\boldsymbol{M}^\top \boldsymbol{M})$.

In what follows, we show that only $\boldsymbol{A}^\top \boldsymbol{A}$ is required to find the pre-image, rather than $\boldsymbol{A}$. Fortunately, we do not need to compute the coefficients $\alpha_{k,j}$ to generate the coordinate system in the RKHS; only their inner products are required.

### 4.2 Stage 2: Operate a Pre-image

Since the model (5) is valid for all the training data, we apply it to do the pre-image, as discussed in this stage. Let $\varphi^*(\cdot)$ be any optimal function resulting from a kernel-based machine, with $\varphi^*(\cdot) = \sum_{i=1}^{n} \gamma_i \kappa(\cdot, \boldsymbol{x}_i)$ as given in (3). By virtue of the representer theorem, it belongs to the subspace spanned by the training kernel functions, and therefore can be expressed in terms of the provided coordinate system. The coordinate of $\varphi^*(\cdot)$ associated to the coordinate function $\psi_k(\cdot)$ is

$$\langle \varphi^*(\cdot), \psi_k(\cdot) \rangle_{\mathcal{H}} = \sum_{i,j=1}^{n} \alpha_{k,i}\, \gamma_j\, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j).$$

Each of these $\ell$ coordinates are computed and collected into one vector, denoted $\Psi_{\varphi^*}$ with some abuse of notation. Thus, we extend the model (5), and write

$$\Psi_{\boldsymbol{x}_i}^\top \Psi_{\varphi^*} = \boldsymbol{x}_i^\top \boldsymbol{x}^*,$$

for $i = 1, 2, \ldots, n$, where $\boldsymbol{x}^*$ is the pre-image to be estimated. This identity can be expressed matrix-wise with

$$\boldsymbol{K}\widehat{\boldsymbol{A}}^\top \widehat{\boldsymbol{A}}\, \boldsymbol{K}\boldsymbol{\gamma} = \boldsymbol{X}^\top \boldsymbol{x}^*$$

where $\boldsymbol{\gamma} = [\gamma_1\ \gamma_2\ \cdots\ \gamma_n]^\top$ and $\boldsymbol{X} = [\boldsymbol{x}_1\ \boldsymbol{x}_2\ \cdots\ \boldsymbol{x}_n]$. By injecting the provided system (6) into this expression, we get

$$\boldsymbol{X}^\top \boldsymbol{x}^* = (\boldsymbol{P} - \lambda\, \boldsymbol{K}^{-1})\, \boldsymbol{\gamma}. \tag{7}$$

This is a classical system of linear equations. Thus, the pre-image can be estimated by applying any off-the-shelf solver. For instance, one can solve the linear least-squares optimization problem

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \| \boldsymbol{X}^\top \boldsymbol{x} - (\boldsymbol{P} - \lambda\, \boldsymbol{K}^{-1})\, \boldsymbol{\gamma} \|^2, \tag{8}$$

where any iterative or non-iterative technique can be used, such as the pseudo-inverse or the eigen-decomposition,[3] in the spirit of the Nyström method. It is worth noting that the optimization scheme is applied here to the input space, as opposed to high dimensional RKHS with the fixed-point iteration schemes. Moreover, one needs only to consider solution from the span of the training data, in coherence with previous work on the pre-image problem [11, 14]. The proposed method is universal in the sense of being independent, in its formulation, of both the type of the adopted kernel and of the feature under investigation.

In order to better understand this result, consider the potential theoretical setting of linear independent training data. In this case, the minimization problem (8) has a unique solution, given by solving the normal equations $\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{x}^* = \boldsymbol{X}\big(\boldsymbol{P} - \lambda\, \boldsymbol{K}^{-1}\big)\, \boldsymbol{\gamma}$. By using the pseudo-inverse matrix algebra with the identity $(\boldsymbol{X}\boldsymbol{X}^\top)^{-1}\boldsymbol{X} = \boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$, we get

$$\boldsymbol{x}^* = \boldsymbol{X}\boldsymbol{P}^{-1}\big(\boldsymbol{P} - \lambda\, \boldsymbol{K}^{-1}\big)\, \boldsymbol{\gamma}. \tag{9}$$

### 4.3 Extension to a Set of Features

These expressions can be applied readily to a set of features in the RKHS to get their pre-images in the input space. This can be done straightforwardly by writing (7) as

$$\boldsymbol{X}^\top \boldsymbol{X}^* = (\boldsymbol{P} - \lambda\, \boldsymbol{K}^{-1})\, \boldsymbol{\Gamma},$$

where each column of matrix $\boldsymbol{\Gamma}$ represents the coefficient vector $\boldsymbol{\gamma}$, and each column of $\boldsymbol{X}^*$ the corresponding pre-image. From the solution (9), we see that the matrix

$$\boldsymbol{M} = \boldsymbol{X}\boldsymbol{P}^{-1}\big(\boldsymbol{P} - \lambda\, \boldsymbol{K}^{-1}\big)$$

needs to be computed only once, and then applied with

$$\boldsymbol{X}^* = \boldsymbol{M}\, \boldsymbol{\Gamma}.$$

This corresponds to a matrix completion scheme, or more specifically the kernel matrix regression approach, as given in [9, 21].

**Table 2** Values of the parameters for the synthetic datasets.

| | $n_{\text{train}}$ | $n_{\text{pre-image}}$ | $\nu$ | $n_{\text{eigen}}$ | $\sigma$ |
|---|---|---|---|---|---|
| Frame | 350 | 850 | 0.1 | 5 | 0.4 |
| Banana | 300 | 200 | 0.2 | 3 | 0.5 |
| Spiral | 70 | 250 | 0.3 | 10 | 0.3 |
| Sine | 420 | 330 | 0.5 | 10 | 0.4 |

[3] Doing eigen-decomposition gives the pre-image relative to the eigen-basis in the input space. A post-processing is required to set the pre-image relative to the training data; this is called the procrustes problem.

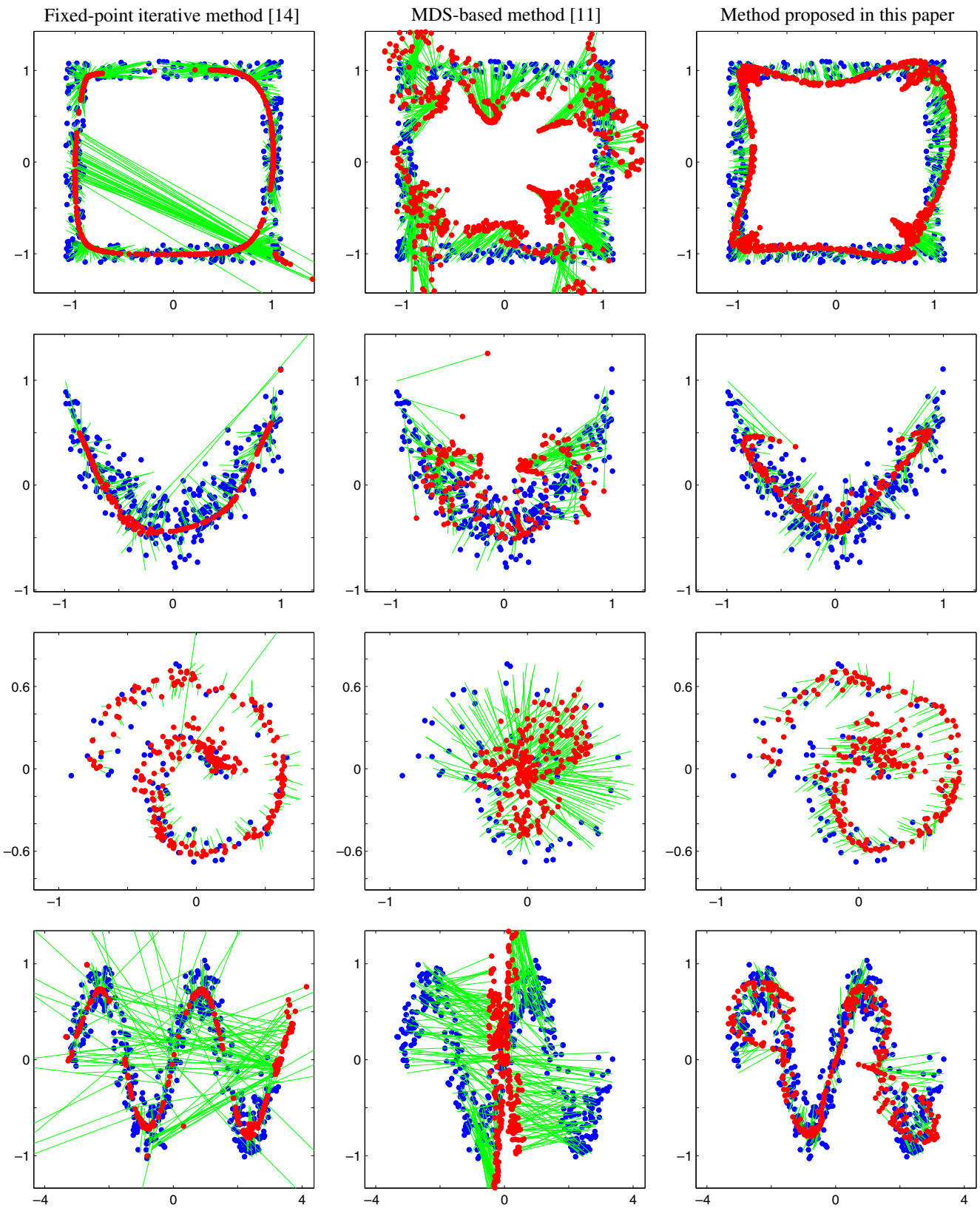


**Figure 2** Experimental results for the frame (*first row*), the banana (*second row*), the spiral (*third row*), and the sine (*fourth row*) datasets, using the fixed-point iterative (*left*), the MDS-based (*middle*), and the proposed (*right*) algorithms. Training data are represented by *blue dots*, estimated pre-images by *red dots*, and *green lines* illustrate the distance between these estimates and the initial noisy data (not shown).

## 5 Experiments

In this section, we compare the proposed method with two state-of-the art methods:[4] the fixed-point iterative technique [14] and the MDS-based approach [11]. For this purpose, the kernel-PCA for denoising is applied on synthetic and real datasets. The Gaussian kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2)$ is used, its bandwidth $\sigma$ fixed to the same value for the three methods.

### 5.1 Synthetic Datasets

We consider a family of four datasets in 2-D, each having a geometric form corrupted by a noise of bandwidth parameter $\nu$. The data are sampled uniformly randomly within this area. We generate $n_{\text{train}}$ data to train the $n_{\text{eigen}}$ eigenfunctions and to construct the coordinate system. Then, we apply these results on another set of $n_{\text{pre-image}}$ generated data, in order to denoise using the pre-image techniques. For each dataset, the parameters values are summarized in Table 2.

The *frame* dataset consists of a square with sides of length 2. Data are generated uniformly randomly on each side and corrupted by a noise uniformly distributed on the interval $[-\nu, \nu]$ normal to the side. The *banana* dataset is given by the parabola defined by the coordinates $(x, x^2 + \xi)$, with $x$ on the $x$-axis uniformly distributed on the interval $[-1, 1]$, and $\xi$ normally distributed with a standard deviation of $\nu$. The *spiral* is defined by the coordinates $(A(\varphi)\cos(\varphi), A(\varphi)\sin(\varphi))$, with $A(\varphi) = 0.07\varphi + \xi$, where $\varphi$ and $\xi$ are generated uniformly on the intervals $[0, 6\pi]$ and $[0, \nu]$, respectively. The *sine* dataset is defined by the coordinates $(\varphi, 0.8\sin(2\varphi))$, where $\varphi$ is generated uniformly on the interval $[0, 2\pi]$, and corrupted with an additive uniformly distributed noise in the range $[0, \nu]^2$. See [7] for more information.

The fixed-point iterative algorithm is set with a stopping criterion of maximum 100 iterations, reaching the limit of reasonable cpu time. The initial estimate is chosen from the valid model $\boldsymbol{x}^* = \sum_i \gamma_i \boldsymbol{x}_i$, with the weighting coefficients $\gamma_i$ generated uniformly on the interval $[-1, 1]$. The MDS-based algorithm operates using a global optimization scheme, which gives better results than the neighborhood setting. Since this algorithm is based on an eigen-decomposition technique, it results in a new coordinate system in the input space. Hence, we consider a procrustes technique to align it with the initial canonical one, by minimizing the mean-squares error.

In Fig. 2, we show the four datasets, with on the one hand the training data (blue dots), and on the other the denoised estimates (red dots) obtained from another set of noisy data (not shown here, yet given by the unmarked ends of green lines). Green lines show the distance between the denoised and the initial noisy data.

The fixed-point iterative method suffers on one side from numerical instabilities, illustrated through many estimates falling outside the bounds of the images (following the long green lines), and on the other from local minima, illustrated with improper denoising (for instance, the upper border of the frame dataset ($y$-axis close to 1) are not denoised to the same area). It is obvious that the MDS-based approach is clearly inappropriate to any of the given datasets. The method presented in this paper gives good results with the four proposed datasets, with the smallest reconstruction error of all algorithms. It seems less sharper in denoising than the fixed-point iterative algorithm, without suffering from the drawbacks of the latter. However, it causes the estimates to fold over itself, in the same sense of manifold learning. This is illustrated for instance with the banana data, yet much less pronounced than the MDS-based results.

### 5.2 Real Datasets

We illustrated the efficiency of our method on denoising real datasets. We consider the handwritten digit "2", obtained from the MNIST database of handwritten digits [12]. The images are (almost) binary images of $28 \times 28$ pixels. Hence, from a machine learning point of view, each image is simply a point in the 784-dimensional space. The original images were corrupted by adding a zero-mean white Gaussian noise with variance $\nu = 0.1$. A set of $n_{\text{train}} = 1{,}000$ images are used to train the kernel-PCA with the $n_{\text{eigen}} = 100$ leading principal functions retained. We apply the Gaussian kernel to all three algorithms, with bandwidth set to $\sigma = 10^5$. The parameter settings are summarized in Table 3.

To illustrate the denoised ability of each algorithm, another set of $n_{\text{pre-image}} = 10$ images is considered under the same noise conditions. These images are illustrated in Fig. 3 (first row), with results from the fixed-point

**Table 3** Values of the parameters for the real digit dataset.

| $n_{\text{train}}$ | $n_{\text{pre-image}}$ | $\nu$ | $n_{\text{eigen}}$ | $\sigma$ |
|---|---|---|---|---|
| 1,000 | 10 | 0.1 | 100 | $10^5$ |

[4] Matlab codes for these algorithms are available from the Statistical Pattern Recognition Toolbox http://cmp.felk.cvut.cz/cmp/software/stprtool/.

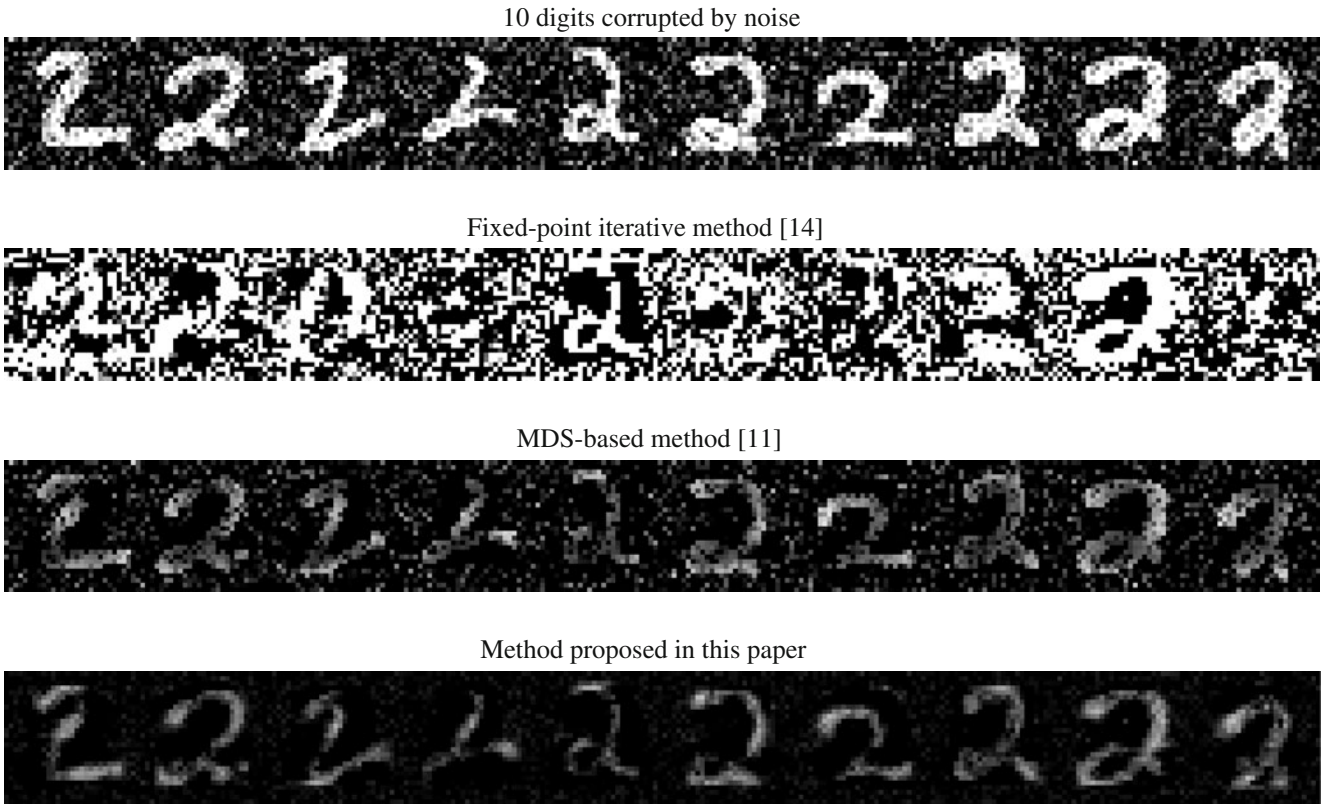


**Figure 3** Comparative analysis for denoising a set of ten "2" digits (*first row*), with the denoised images from the fixed-point iterative (*second row*), the MDS-based (*third row*), and the proposed (*fourth row*) algorithms.

iterative (second row), the MDS-based (third row) and the proposed (fourth row) algorithms. It is obvious that fixed-point iterative algorithm is inappropriate for such application, even with the number of maximum iterations set to 10,000 corresponding to an average total CPU time of up to one hour and a half. To take advantage of prior knowledge, the same training set is used for learning the inverse map. Realistic results can be obtained using the MDS-based algorithm, with five minutes and a half. The algorithm proposed in this paper achieves better denoised results, as illustrated in Fig. 3. For this simulation, the regularization parameter was set to $\lambda = 10^{-9}$, and the resulting average total CPU time is 1.3 s.[5]

## 6 Conclusion

In this paper, we presented a new method to solve the pre-image problem. As opposed to previous work, the proposed method neither suffers from numerical instability, nor requires computing the distances in the input and the RKHS spaces. We showed that using the inner product information in both spaces, we can provide a coordinate system in the RKHS to learn the inverse map. The efficiency of the proposed method were studied with experiments on both synthetic data and real handwritten digits, and compared to state-of-the-art methods. The major advantage of the proposed method resides on its simplicity in dealing with the optimization issue, thanks to conventional linear algebra.

[5]CPU times are given only as an indication of the computations required for the various algorithms.

## References

1. Aizerman, M., Braverman, E., & Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control, 25*, 821–837.
2. Arias, P., Randall, G., & Sapiro, G. (2007). Connecting the out-of-sample and pre-image problems in kernel methods. In *IEEE Computer Society conference on computer vision and pattern recognition*. http://ampere.iie.edu.uy/publicaciones/2007/ARS07.

3. Aronszajn, N. (1950). Theory of reproducing kernels. *Transactions of the American Mathematical Society, 68*, 337–404.
4. Bakir, G., Weston, J., & Schölkopf, B. (2004). Learning to find pre-images. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *NIPS 2003* (Vol. 16, pp. 449–456). Cambridge, MA: MIT Press.
5. Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Roux, N. L., & Ouimet, M. (2004). Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In S. Thrun, L. Saul, & B. Schölkopf (Eds.), *Advances in neural information processing systems* (Vol. 16). Cambridge, MA: MIT Press.
6. Etyngier, P., SÈgonne, F., & Keriven, R. (2007). Shape priors using manifold learning techniques. In *11th IEEE international conference on computer vision*. Rio de Janeiro, Brazil.
7. Hoffmann, H. (2007). Kernel PCA for novelty detection. *Pattern Recognition, 40*, 863–874.
8. Honeine, P., & Richard, C. (2009). Solving the pre-image problem in kernel machines: A direct method. In *IEEE workshop on machine learning for signal processing*. Grenoble, France.
9. Honeine, P., Richard, C., Essoloh, M., & Snoussi, H. (2008). Localization in sensor networks—A matrix regression approach. In *5th IEEE sensor array and multichannel signal processing workshop (SAM)*. Darmstadt, Germany.
10. Kimeldorf, G., & Wahba, G. (1971). Some results on tchebycheffian spline functions. *Journal of Mathematical Analysis and Applications, 33*, 82–95.
11. Kwok, J. T., & Tsang, I. W. (2003). The pre-image problem in kernel methods. In *Machine learning, proceedings of the twentieth international conference (ICML 2003)* (pp. 408–415). Washington, DC: AAAI Press.
12. Lecun, Y., & Cortes, C. (1998). The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.
13. Mika, S. (2002). *Kernel fisher discriminants*. Ph.D. thesis, University of Technology, Berlin.
14. Mika, S., Schölkopf, B., Smola, A., Müller, K., Scholz, M., & Rätsch, G. (1999). Kernel pca and de-noising in feature spaces. In *Proceedings of the 1998 conference on advances in neural information processing systems II* (pp. 536–542). Cambridge, MA: MIT Press.
15. Rathi, Y., Dambreville, S., & Tannenbaum, A. (2006). Statistical shape analysis using kernel pca. In *IS&T/SPIE symposium on electronic imaging*.
16. Schölkopf, B., Herbrich, R., & Williamson, R. (2000). *A generalized representer theorem*. Tech. rep. NC2-TR-2000-81, Royal Holloway College, Univ. of London, UK.
17. Schölkopf, B., Smola, A., & Müller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation, 10*(5), 1299–1319.
18. Suykens, J., Gestel, T. V., Brabanter, J. D., Moor, B. D., & Vandewalle, J. (2002). *Least squares support vector machines*. Singapore: World Scientific.
19. Tax, D. (2001). *One-class classification; concept-learning in the absence of counter-examples*. Ph.D. thesis, Advanced School for Computing and Imaging—Delft University of Technology.
20. Vapnik, V. (1998). *Statistical learning theory*. New York: Wiley.
21. Yamanishi, Y., & Vert, J. P. (2007). *Kernel matrix regression*. Tech. rep. http://arxiv.org/abs/q-bio/0702054v1.


**Paul Honeine** was born in Beirut, Lebanon, on October 2, 1977. He received the Dipl.-Ing. degree in mechanical engineering in 2002 and the M.Sc. degree in industrial control in 2003, both from the Faculty of Engineering, the Lebanese University, Lebanon. In 2007, he received the Ph.D. degree in Systems Optimisation and Security from the University of Technology of Troyes, France, and was a Postdoctoral Research associate with the Systems Modeling and Dependability Laboratory, from 2007 to 2008.

Since September 2008, he has been an assistant Professor at the University of Technology of Troyes, France. His research interests include nonstationary signal analysis, nonlinear adaptive filtering, sparse representations, machine learning, and wireless sensor networks.

He is the co-author (with C. Richard) of the 2009 Best Paper Award at the IEEE Workshop on Machine Learning for Signal Processing.

**Cédric Richard** (S'98–M'01–SM'07) was born January 24, 1970 in Sarrebourg, France. I received the Dipl.-Ing. and the M.S. degrees in 1994 and the Ph.D. degree in 1998 from the University of Technology of Compiègne, France, all in Electrical and Computer Engineering. From 1999 to 2003, he was an Associate Professor at the University of Technology of Troyes, France. From 2003 to 2009, he was a Full Professor at the Institut Charles Delaunay (CNRS FRE 2848) at the UTT, and the supervisor of a group consisting of 60 researchers and Ph.D. In winter 2009, he was a Visiting Researcher with the Department of Electrical

Engineering, Federal University of Santa Catarina (UFSC), Florianòpolis, Brazil.

Since September 2009, Cédric Richard is a Full Professor at Fizeau Laboratory (CNRS UMR 6525, Observatoire de la Côte d'Azur), University of Nice Sophia-Antipolis, France. His current research interests include statistical signal processing and machine learning. Prof. Cédric Richard is the author of over 100 papers. He was the General Chair of the XXIth francophone conference GRETSI on Signal and Image Processing that was held in Troyes, France, in 2007. Since 2005, he is in charge of the Ph.D. students network of the federative CNRS research group ISIS on Information, Signal, Images and Vision. He is a member of GRETSI association board and of the EURASIP society, and Senior Member of the IEEE.

Cédric Richard serves as an Associate Editor of the IEEE Transactions on Signal Processing since 2006, and of the EURASIP Signal Processing Magazine since 2009. In 2009, he was nominated liaison local officer for EURASIP, and member of the Signal Processing Theory and Methods (SPTM) Technical Committee of the IEEE Signal Processing Society.

Paul Honeine and Cédric Richard received Best Paper Award for "Solving the pre-image problem in kernel machines: a direct method" at the 2009 IEEE Workshop on Machine Learning for Signal Processing (IEEE MLSP'09).