# Kernel PCA Pattern Reconstruction *via* Approximate Pre-Images

Bernhard Schölkopf, Sebastian Mika,
Alex Smola, Gunnar Rätsch, & Klaus-Robert Müller
GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany
{bs, mika, smola, raetsch, klaus}@first.gmd.de

**Abstract**

Algorithms based on Mercer kernels construct their solutions in terms of expansions in a high-dimensional feature space $F$. Previous work has shown that all algorithms which can be formulated in terms of dot products in $F$ can be performed using a kernel without explicitly working in $F$. The list of such algorithms includes support vector machines and nonlinear kernel principal component extraction. So far, however, it did not include the reconstruction of patterns from their largest nonlinear principal components, a technique which is common practice in linear principal component analysis.

The present work proposes an idea for approximately performing this task. As an illustrative example, an application to the de-noising of data clusters is presented.

## 1 Kernels and Feature Spaces

A Mercer kernel is a function $k(\mathbf{x}, \mathbf{y})$ which for all data sets $\{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\} \subset \mathbf{R}^N$ gives rise to a positive (not necessarily definite) matrix $K_{ij} := k(\mathbf{x}_i, \mathbf{x}_j)$ [4]. One can show that using $k$ instead of a dot product in $\mathbf{R}^N$ corresponds to mapping the data into a possibly high-dimensional space $F$ by a (usually nonlinear) map $\Phi$, and taking the dot product there, i.e. $k(\mathbf{x}, \mathbf{y}) = (\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$ [1].

Support vector (SV) machines construct linear functions on $F$, corresponding to nonlinear functions on $\mathbf{R}^N$ [1]. Kernel principal component analysis (kernel PCA) extracts linear principal components in $F$, corresponding to nonlinear feature extractors on $\mathbf{R}^N$ [6]. Kernels that have proven useful in these algorithms include Gaussian kernels

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\,\sigma^2)\right), \tag{1}$$

and polynomial kernels $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$, which compute dot products in feature spaces spanned by all possible products of input dimensions [1, 5].

In both cases, the corresponding $\Phi$ is nonlinear, and the dimensionality of the linear span of the $\Phi$-images of our data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_\ell\}$ can exceed the dimensionality of their span in input space, provided $\ell$ is sufficiently large. Thus, we cannot expect that there exists a pre-image under $\Phi$ for each vector $\Psi \in F$ that can be expressed as a linear combination of the vectors $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_\ell)$, i.e. a $\mathbf{z}$ such that $\Phi(\mathbf{z}) = \Psi$. Examples of the latter include, as we shall see below, kernel PCA Eigenvectors, and linear combinations thereof. For the cases where the pre-image does exist, we can provide a means of constructing it: to this end, suppose we have a vector in $F$ (e.g. a principal axis in kernel PCA, or the normal vector of a hyperplane describing a SV classifier) given in terms of an expansion of images of input data, with an unknown pre-image $\mathbf{x}_0$ under $\Phi$ in input space $\mathbf{R}^N$, i.e. $\Phi(\mathbf{x}_0) = \sum_{j=1}^{\ell} \alpha_j \Phi(\mathbf{x}_j)$; hence, for any $\mathbf{x} \in \mathbf{R}^N$,

$$k(\mathbf{x}_0, \mathbf{x}) = \sum_{j=1}^{\ell} \alpha_j k(\mathbf{x}_j, \mathbf{x}). \tag{2}$$

Assume moreover that the kernel $k(\mathbf{x}, \mathbf{y})$ is an invertible function $f_k$ of $(\mathbf{x} \cdot \mathbf{y})$, e.g. $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d$ with odd $d$, or $k(\mathbf{x}, \mathbf{y}) = \sigma((\mathbf{x} \cdot \mathbf{y}) + \Theta)$ with a strictly monotonic sigmoid function $\sigma$ and a threshold $\Theta$. Given any a priori chosen orthonormal basis of input space $\{\mathbf{e}_1, \dots, \mathbf{e}_N\}$, we can use (2) to expand $\mathbf{x}_0$ as

$$\begin{aligned} \mathbf{x}_0 &= \sum_{i=1}^{N} (\mathbf{x}_0 \cdot \mathbf{e}_i)\mathbf{e}_i = \sum_{i=1}^{N} f_k^{-1}(k(\mathbf{x}_0, \mathbf{e}_i))\mathbf{e}_i \\ &= \sum_{i=1}^{N} f_k^{-1} \left( \sum_{j=1}^{\ell} \alpha_j k(\mathbf{x}_j, \mathbf{e}_i) \right) \mathbf{e}_i. \end{aligned} \tag{3}$$

Using the polarization identity, a similar expansion can be given for radial basis function kernels $k(\mathbf{x}, \mathbf{y}) = f_k\left(\|\mathbf{x} - \mathbf{y}\|^2\right)$; for further details, cf. [5].

The assumption, however, that there exists a pre-image $\mathbf{x}_0$, is a very strong one. To see this, we take a look at one possible representation of $F$. One can show [5] that $\Phi$ can be thought of as a map $\Phi(\mathbf{x}) = k(\mathbf{x}, .)$ into a Hilbert space $\mathcal{H}_k$ of functions $\sum_i \alpha_i k(\mathbf{x}_i, .)$ with a dot product satisfying $\langle k(\mathbf{x}, .), k(\mathbf{y}, .)\rangle = k(\mathbf{x}, \mathbf{y})$. By virtue of this property, $\mathcal{H}_k$ is called a *reproducing kernel Hilbert space* (e.g. [4]). For instance, for the Gaussian kernel (1), $\mathcal{H}_k$ thus contains all linear superpositions of Gaussian bumps on $\mathbf{R}^N$ (plus limit points), whereas by definition of $\Phi$ only single bumps $k(\mathbf{x}, .)$ do have pre-images under $\Phi$.

The remainder of the paper is organized as follows. We briefly review the kernel PCA algorithm, describe the problem of reconstructing patterns from nonlinear principal components, along with an algorithm to solve it. To illustrate the feasibility, we then consider an example of de-noising Gaussian clusters, and conclude with a discussion.

## 2 Kernel PCA and Reconstruction

Principal Component Analysis (PCA) (e.g. [3]) is a basis transformation to diagonalize an estimate of the covariance matrix of the data $\mathbf{x}_k$, $k = 1, \dots, \ell$, $\mathbf{x}_k \in \mathbf{R}^N$, $\sum_{k=1}^{\ell} \mathbf{x}_k = 0$, defined as $C = \frac{1}{\ell} \sum_{j=1}^{\ell} \mathbf{x}_j \mathbf{x}_j^\top$. The new coordinates in the Eigenvector basis, i.e. the orthogonal projections onto the Eigenvectors, are called *principal components*. To generalize this setting to a nonlinear one [6], assume for the moment that our data mapped into feature space,

$\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_\ell)$, are centered, i.e. $\sum_{k=1}^{\ell} \Phi(\mathbf{x}_k) = 0$.[1] To do PCA for the covariance matrix

$$\bar{C} = \frac{1}{\ell} \sum_{j=1}^{\ell} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^\top, \tag{4}$$

we have to find Eigenvalues $\lambda \geq 0$ and Eigenvectors $\mathbf{V} \in F \backslash \{0\}$ satisfying $\lambda \mathbf{V} = \bar{C}\mathbf{V}$. Substituting (4), we note that all solutions $\mathbf{V}$ with $\lambda \neq 0$ lie in the span of $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_\ell)$. This implies that we may consider the equivalent system

$$\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot \bar{C}\mathbf{V}) \text{ for all } k = 1, \ldots, \ell, \tag{5}$$

and that there exist coefficients $\alpha_1, \ldots, \alpha_\ell$ such that

$$\mathbf{V} = \sum_{i=1}^{\ell} \alpha_i \Phi(\mathbf{x}_i). \tag{6}$$

Substituting (4) and (6) into (5), and defining an $\ell \times \ell$ matrix $K$ by $K_{ij} := (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = (k(\mathbf{x}_i, \mathbf{x}_j))$, we arrive at a problem which is cast in terms of dot products: solve

$$\ell \lambda \boldsymbol{\alpha} = K \boldsymbol{\alpha} \tag{7}$$

for nonzero Eigenvalues $\lambda$, and coefficient Eigenvectors $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_\ell)^\top$. We normalize the solutions $\boldsymbol{\alpha}^k$ by requiring that the corresponding vectors in $F$ be normalized, i.e. $(\mathbf{V}^k \cdot \mathbf{V}^k) = 1$, which translates into $\lambda_k(\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k) = 1$. For principal component extraction, we compute projections of the image of a test point $\Phi(\mathbf{x})$ onto the Eigenvectors $\mathbf{V}^k$ in $F$ according to

$$\beta_k := (\mathbf{V}^k \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{\ell} \alpha_i^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{\ell} \alpha_i^k k(\mathbf{x}_i, \mathbf{x}). \tag{8}$$

Note that for feature extraction, we thus have to evaluate $\ell$ kernel functions in input space rather than a dot product in a very high–dimensional space. Moreover, kernel PCA can be carried out for all kernels described in Sec. 1, no matter whether we know the corresponding map $\Phi$ or not. The nonlinearity is taken into account implicitly when computing the matrix elements of $K$ and when computing the projections (8), the remainder of the algorithm is simple linear algebra.

Since kernel PCA is nothing but a PCA in $F$, i.e. an orthogonal basis transform in $F$, $\Phi(\mathbf{x})$ can always be reconstructed from its principal components (e.g. [3]). To state this formally, we define a projection operator $P_n$ (assuming the Eigenvectors are sorted such that the Eigenvalue size decreases with $k$) by

$$P_n \Phi(\mathbf{x}) = \sum_{k=1}^{n} \beta_k \mathbf{V}^k. \tag{9}$$

[1] Otherwise, we have to go through the same algebra using $\tilde{\Phi}(\mathbf{x}_i) := \Phi(\mathbf{x}_i) - (1/\ell) \sum_{j=1}^{\ell} \Phi(\mathbf{x}_j)$ (for details, see [6]).

Kernel PCA has the property that (i) if $n$ is sufficiently large to take into account all principal components belonging to Eigenvectors with nonzero Eigenvalues, we have $P_n\Phi(\mathbf{x}_i) = \Phi(\mathbf{x}_i)$, and (ii) if we truncate the expansion for smaller $n$, the overall squared error $\sum_{i=1}^{\ell} \|P_n\Phi(\mathbf{x}_i) - \Phi(\mathbf{x}_i)\|^2$ is minimal among all those obtainable by projecting onto $n$ directions. Typically, however, we are interested in approximate representations of the data in *input space* rather than in $F$. To this end, we are looking for a $\mathbf{z} \in \mathbf{R}^N$ such that

$$\rho(\mathbf{z}) = \|P_n\Phi(\mathbf{x}) - \Phi(\mathbf{z})\|^2 \tag{10}$$

is minimized. Before discussing how to do this, we state two applications:

**De-noising.** Given $\mathbf{x}$, we map it into $\Phi(\mathbf{x})$, discard higher components to obtain $P_n\Phi(\mathbf{x})$, and then compute $\mathbf{z}$. Here, the hope is that the main structure in the data set is captured in the first $n$ directions, and the remaining components mainly pick up the noise — in this sense, $\mathbf{z}$ can be thought of as a de-noised version of $\mathbf{x}$.

**Compression.** Given $\beta_k$ and $\mathbf{V}^k$, $k = 1, \ldots, n$, but not $\mathbf{x}$, compute $\mathbf{z}$ as an approximate reconstruction for $\mathbf{x}$. This is useful if $n$ is smaller than the dimensionality of the input data.

If $P_n\Phi(\mathbf{x}) \neq \Phi(\mathbf{x})$, we cannot guarantee the existence of an exact pre-image (and indeed, in the de-noising case, we do not even want an exact pre-image), i.e. the minimal $\rho$ might be nonzero. Therefore, we opted for an approach different from the one described in Sec. 1, and simply used gradient methods to minimize (10) over $\mathbf{z}$ (cf. [2, 6]). Substituting (6) and (9) into (10), we can express $\rho$ (and thus $\frac{d\rho}{dz}$) in terms of the kernel $k$: collecting terms independent of $\mathbf{z}$ in $C$, the objective function $\rho(\mathbf{z})$ takes the form

$$\rho(\mathbf{z}) = k(\mathbf{z}, \mathbf{z}) - 2\sum_{k=1}^{n} \beta_k \sum_{i=1}^{\ell} \alpha_i^k k(\mathbf{x}_i, \mathbf{z}) + C. \tag{11}$$

## 3 Experiments

In the present work, we focus on the application to de-noising, which differs from compression by the fact that we are allowed to make use of the original data. We took advantage of this in the minimization of (10), by using conjugate gradient descent with the respective example $\mathbf{x}$ as a starting point. All experiments reported were carried out with Gaussian kernels (1), however, similar results were obtained with polynomial kernels. We generated an artificial data set from three point sources at (-0.5,-0.1), (0,0.7), (0.5,0.1) (100 points each) with Gaussian noise ($\sigma = 0.1$), and performed kernel PCA on it (Fig. 1).

Using the resulting Eigenvectors, we extracted nonlinear principal components from a set of test points generated from the same model, and reconstructed the points from varying numbers of principal components. Figure 2 shows that discarding higher-order components leads to removal of the noise — the points move towards their respective sources.

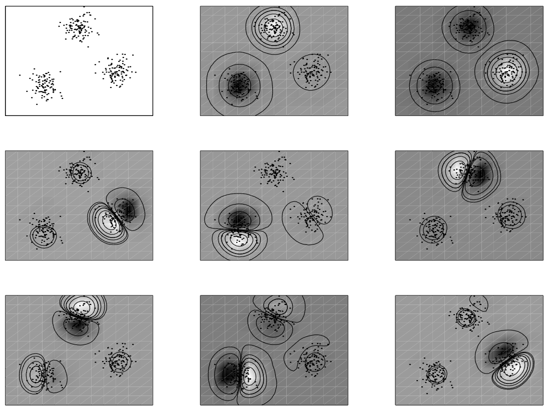

Figure 1: Kernel PCA toy example (see text): lines of constant feature value for the first 8 nonlinear principal components extracted with $k(\mathbf{x},\mathbf{y}) = \exp\left(-\|\mathbf{x}-\mathbf{y}\|^2/0.1\right)$. The first 2 principal component (top middle/right) separate the three clusters. Components 3–5 split the clusters. Components 6–8 split them again, orthogonal to the above splits.

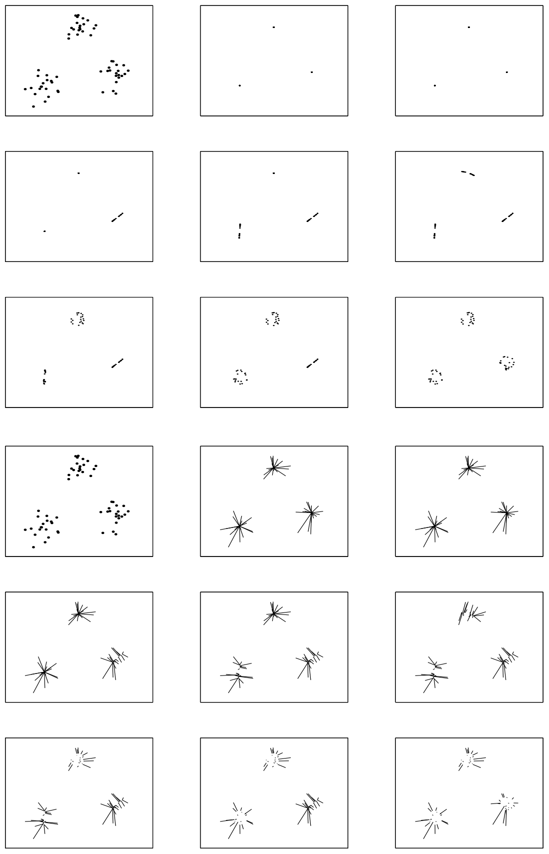

Figure 2: Kernel PCA denoising by reconstruction from projections onto the Eigenvectors of Fig. 1. We generated 20 new points from each Gaussian, represented them in feature space by their first $n = 1, 2, \ldots, 8$ nonlinear principal components, and computed approximate pre-images, shown in the upper 9 pictures (top left: original data, top middle: $n = 1$, top right: $n = 2$, etc.). Note that by discarding higher order principal components (i.e. using a small $n$), we remove the noise inherent in the nonzero variance $\sigma^2$ of the Gaussians. The lower 9 pictures show how the original points "move" in the denoising. Unlike the corresponding case in linear PCA, where where we obtain lines (see Figure 3), in kernel PCA clusters shrink to points.

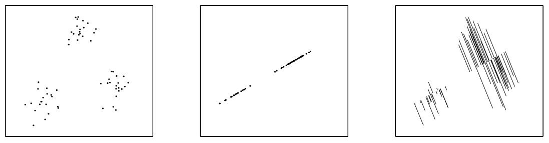

Figure 3: Reconstructions and point movements for linear PCA, based on the first principal component.

## 4 Discussion

We have studied the problem of finding pre-image of vectors in feature spaces, and shown how it is related to the problem of reconstructing data from its nonlinear principal components as extracted by the kernel PCA algorithm. We have proposed an algorithm to perform the reconstruction, and presented a toy example in order to illustrate that the basic idea is sound. A thorough experimental evaluation on real-world data is currently being carried out.

Clearly, there are other possibilities for constructing the mapping from the nonlinear feature representation back to the data. For instance, we could have trained a network to approximate this mapping, which would yield an overall architecture similar to a bottleneck network with (8) serving as the mapping into the bottleneck. Alternatively, we could have thrown all theoretical worries overboard, pretending that exact pre-images did exist, and hoping that (3) would give us a sensible approximation even though the former is not the case.

Open questions and problems include the choice of a suitable kernel for a given noise reduction problem, possibly in conjunction with the regularization properties of the kernel, faster reconstruction algorithms, the application of the approach to compression, and the study of the optimal cut-off, i.e. the unsolved problem of where the structure ends, and where the noise begins (see e.g. [3]).

**Acknowledgments.** BS and AS were supported by a grant from the Australian Research Council, and carried out part of the work at the Australian National University, Canberra. AS was supported by the DFG (# Ja 379/71).

## References

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.

[2] C. J. C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings, 13th Intl. Conf. on Machine Learning*, pages 71–77, San Mateo, CA, 1996. Morgan Kaufmann.

[3] K. I. Diamantaras and S. Y. Kung. *Principal Component Neural Networks.* Wiley, New York, 1996.

[4] S. Saitoh. *Theory of Reproducing Kernels and its Applications.* Longman Scientific & Technical, Harlow, England, 1988.

[5] B. Schölkopf. *Support Vector Learning.* Oldenbourg Verlag, Munich, 1997.

[6] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299 – 1319, 1998.