Ukrainian Catholic University

Applied Sciences Faculty

Data Science Master Programme

Kernel Principal Component Analysis and its Applications

Project report

Authors:
Irynei Baran
Hanna Pylieva

February 18, 2018

Abstract

Principal component analysis (PCA) is a popular tool for linear dimensionality reduction and feature extraction. Kernel PCA is the nonlinear form of PCA, which better exploits the complicated spatial structure of high-dimensional features. In this project, we review the basic ideas of PCA, identify its limitations and possible modifications. Then we concentrate on one important extension of PCA - kernel PCA. We give the detailed explanation of the latter, followed by step-by-step instructions of the algorithm's implementation. We also provide the source code of kPCA developed by ourselves with comparison of its performance against linear PCA. Finally, we briefly specify the possible applications of kPCA and discuss the method's pros and cons.

1 Principal Component Analysis

Principal component analysis, or PCA, is a mathematical procedure which is widely used for dimensionality reduction and feature selection. Those applications are achieved by projecting the data orthogonally onto a linear space with lower dimension, known as the principal subspace or feature space, such that the variance of projected data is maximal (Bishop , 2009).

Consider a data set X containing N observations of D features ($D < N$). In order to visualize data or diminish number of features for modeling we want to reduce dimensionality of feature space to $M < D$. Whereas we are interested in most influential features to minimize data losses, that is why our goal is to maximize variance of projected data. The directions on which the data is projected called principal components. They are orthogonal and form coordinate system of subspace M (see on Figure 1).

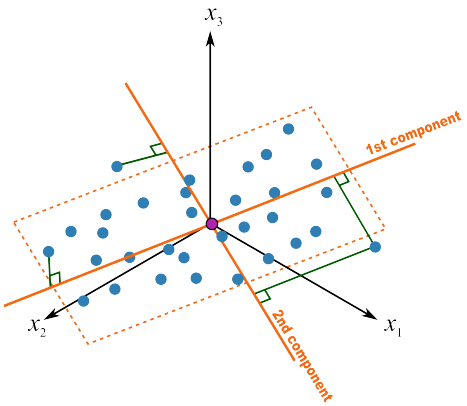

Figure 1: Principal components

1.1 Finding principal components

To begin with, consider the projection on M when $dim(M) = 1$. Let a unit vector $u_1 \in D$ be the direction of M. Then projection of an observation $x_n \in X$ onto M is $u_1^T x_n$ and the variance of projected data is

$$\frac{1}{N}\sum_{n=1}^{N}\{u_1^T x_n - u_1^T \bar{x}\} = u_1^T S u_1 \tag{1}$$

where $\bar{x} = \frac{1}{N}\sum_{n=1}^{N} x_i$ is the mean of sample set and S is the covariance matrix of data set X:

$$S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T \tag{2}$$

Maximization of (1) is kept in a unit circle as we chose u_1 s.t. $||u_1|| = u_1^T u_1 = 1$. So we need to find maximum of the next Lagrange function:

$$L(X, \lambda_1) = u_1^T S u_1 + \lambda_1(1 - u_1^T u_1) \tag{3}$$

By setting the derivative with respect to u_1 equal to zero we find that in stationary point u_1 needs to be an eigenvector of S:

$$S u_1 = \lambda_1 u_1 \tag{4}$$

Now when we left-multiply by u_1^T and make use of $u_1^T u_1 = 1$ we find out that the variance is given by

$$u_1^T S u_1 = \lambda_1 \tag{5}$$

and so the variance will be a maximum when we set u_1 equal to the eigenvector having the largest eigenvalue λ_1. This eigenvector id called the first principal component (Bishop , 2009).

Next principal components can be found following the same procedure and choosing each new direction such that it maximizes the projected variance amongst all possible directions orthogonal to those already considered.

1.2 Limitations of standard PCA

Although PCA is very useful from practical prospective it has some limitations.

1. Assumptions of linear dependency. PCA projects data orthogonally to reduce dimensionality. This works only if data has linear dependent variables: $y = kx + \epsilon$. Otherwise the method doesn't identify the direction where projected data has highest possible variance which can be seen on Figure 2 .

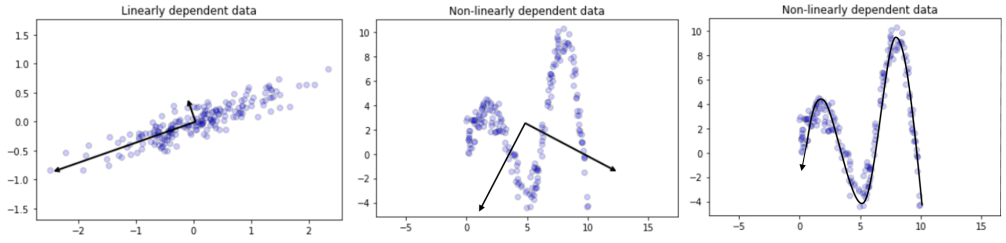

Figure 2: Demonstration of PCA's low performance for non-linearly dependent data

2. Spread maximization. PCA searches for a subspace where projected data has the maximal spread. However, this is not always the best way to represent data in lower-dimension subspace. A common example is the task of separating and counting pancakes from an image (see Figure 3). Important information for the task (number of pancakes) is located along Z axis which has the lowest variance. So Z axis will be the last principal component identified by PCA which is not what required.

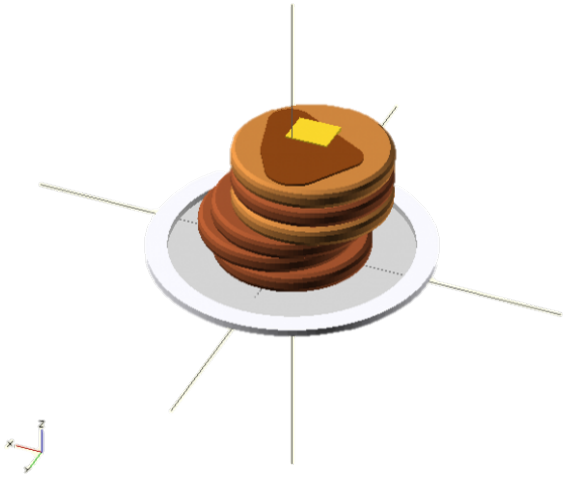

Figure 3: Image from 3D Parametric Pancakes

3. Principal components are hard to interpret. PCA reveals implicit dependencies in data which is non-trivial to interpret in case of high-dimensional space reduction. The leverage for this issue is deep understanding of domain.

4. Orthogonality assumptions. PCA comes up with orthogonal principal components, which do not overlap in the space. For some tasks such functionality will produce wrong results (see Figure 4).

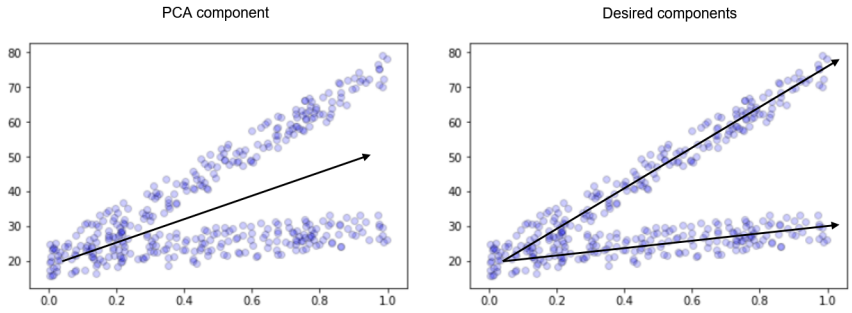

Figure 4: Demonstration of PCA's low performance for non-linearly dependent data

5. Outliers and missing data sensitivity. We introduced PCA here as eigenvalue decomposition of a data covariance matrix. The latter is highly sensitive to sample outliers and corrupted data (also referred as intra-sample outliers) (Shlens , 2014). This raises two issues:

(a) PCA is a linear combinations of all input variables whereas in case of corrupted or missing entries presence we need to exclude some incomplete variables.

(b) in some contexts, outliers can be difficult to identify. In this case we are unable to remove them in order to get better performance from PCA. Whereas there is a way to overcome the issue by introducing robustness to the algorithm. This will be considered beneath.

2 Modifications of PCA

Despite the limitations mentioned, PCA is still a powerful method for data analysis, visualization and dimensionality reduction. That is why numerous modifications were developed to overcome its drawbacks and broaden scope of applications. We will provide succinct description of the most common extensions of PCA.

2.1 Robust PCA

Gross errors in observations are now ubiquitous in modern applications such as image processing, web data analysis, and bioinformatics, where some measurements may be arbitrarily corrupted (due to occlusions, malicious tampering, or sensor failures) or simply irrelevant to the low-dimensional structure we seek to identify with PCA.The best algorithm which deals with data corruption is Robust PCA (RPCA).

Suppose the data under study can naturally be decomposed into low-rank (L_0) and sparse (S_0) components[1]:

$$M = L_0 + S_0 \tag{6}$$

RPCA allows to recover a low-rank matrix L_0 from highly corrupted measurements in M. Classical PCA works good when the noise term N_0 in M is small. In contrast to this S_0 can have arbitrarily large magnitude, and their support is assumed to be sparse but unknown (Cardes , 2009).

2.2 Multilinear PCA

Suppose we need to reduce dimensionality not of a simple matrix $M \in R^2$ but of a n-way array, i.e. a cube or hyper-cube of numbers, also informally referred to as a "data tensor". Common examples of tensor are 2-D/3-D images and video sequences. To deal with tensors PCA is generalized to multilinear PCA (MPCA). MPCA performs feature extraction by determining a multilinear projection that captures most of the original tensorial input variation. The solution is iterative in nature and it proceeds by decomposing the original problem to a series of multiple projection subproblems (Lu , 2008).

MPCA is applied to 3-D object recognition tasks in machine vision, medical image analysis, space-time analysis of video sequences for activity recognition in human-computer interaction, etc. MPCA is further extended to uncorrelated MPCA, non-negative MPCA and robust MPCA.

[1]This is not a synthetic requirement and can be done in a number of applications, refer to (Cardes , 2009) for examples.

2.3 Nonlinear generalizations

In real-world problems we often need to work with non-linear dependencies between features in data. For such tasks Hastie and Stuetzle (Hastie , 1989) proposed bending the loading vectors to produce curves that approximate the nonlinear relationship between a set of two variables. Such curves are called principal curves, their multidimensional extensions produce principal surfaces or principal manifolds (Gorban , 2007).

Another popular way to use PCA in case of nonlinearity is kernel PCA and rest of the report will be devoted to it.

3 Kernel Principal Component Analysis

Kernel principal component analysis, or kPCA, is a nonlinear generalization of PCA using technique of kernel methods, also known as "kernel trick". The main idea is to map the original data nonlinearly into a feature space F by

$$\phi : R^N \to F \tag{7}$$

and then perform PCA, which implicitly defines nonlinear principal components in the original data space (see Figure 5). Even if F has arbitrarily large dimensionality, for certain choices of ϕ it is still possible to perform PCA in F. This is done by use of kernel functions (Scholkopf , 1998).

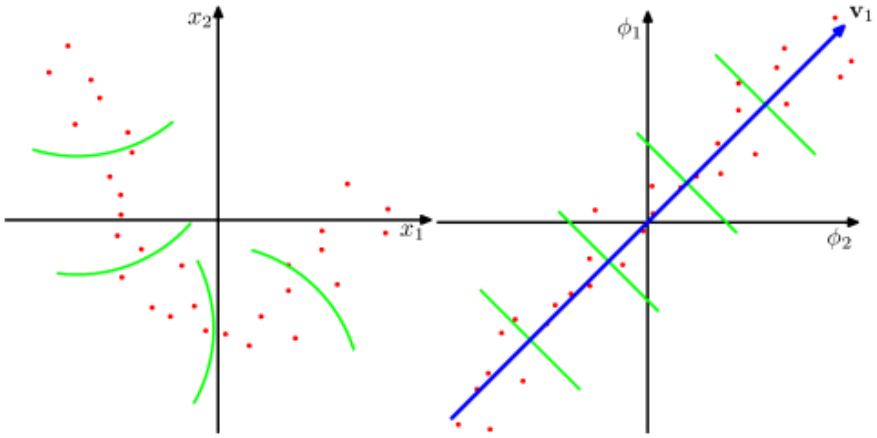

Figure 5: Kernel PCA. Data set in original data space (left-hand plot) and in the feature space F (right-hand plot). $\boldsymbol{v_1}$ is a first principal component obtained in the feature space. Green lines indicate linear projections of data onto $\boldsymbol{v_1}$ (Bishop , 2009)

3.1 Algorithm explanation

Consider a data set X containing N observations of D features ($D < N$) and a nonlinear transformation $\phi(x)$ into an M-dimensional feature space F. For now let us assume that

projected data set is centered, so $\frac{1}{N}\sum_{n=1}^{N}\phi(x_n) = 0$. The $M \times M$ covariance matrix in feature space is given by

$$C = \frac{1}{N}\sum_{n=1}^{N}\phi(x_n)\phi(x_n)^T \tag{8}$$

3.1.1 Eigenvalue equation derivation

We need to solve the following eigenvalue problem

$$C\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \tag{9}$$

$i = 1, ..., M$. Our goal is to solve this equation without working directly in the feature space as due to its size this will be computationally inefficient. Substituting C from (8) we get

$$\frac{1}{N}\sum_{n=1}^{N}\phi(x_n)\{\phi(x_n)^T\boldsymbol{v}_i\} = \lambda_i \boldsymbol{v}_i \tag{10}$$

Provided $\lambda_i > 0$, the vector v_i is given by a linear combination of the $\phi(x_n)$ and so $\forall\ i\ \exists$ column vector $\boldsymbol{\alpha}_i \in R^N$ such that:

$$v_i = \sum_{n=1}^{N}\alpha_{in}\phi(x_n) \tag{11}$$

Substituting (8) and (11) into (9), we obtain

$$\frac{1}{N}\sum_{n=1}^{N}\phi(x_n)\phi(x_n)^T\sum_{m=1}^{N}\alpha_{im}\phi(x_m)^T = \lambda_i\sum_{n=1}^{N}\alpha_{in}\phi(x_n)^T \tag{12}$$

The key thing here is to express last equation in terms of kernel function defined as $k(x_n, x_m) = \phi(x_n)^T\phi(x_m)$. It is done by multiplying both sides by $\phi(x_l)^T$ which results to the next:

$$\frac{1}{N}\sum_{n=1}^{N}k(x_l, x_m)\sum_{m=1}^{N}\alpha_{im}k(x_n, x_m) = \lambda_i\sum_{n=1}^{N}\alpha_{in}k(x_l, x_m) \tag{13}$$

or in matrix notation

$$K^2\boldsymbol{\alpha}_i = \lambda_i N K\boldsymbol{\alpha}_i \tag{14}$$

If we remove a factor of K from both sides we obtain following eigenvalue problem

$$K\boldsymbol{\alpha}_i = \lambda_i N\boldsymbol{\alpha}_i \tag{15}$$

By solving the problem we find eigenvectors $\boldsymbol{\alpha}_i$. Note that solutions of (14) and (15) differ only by eigenvectors that correspond to zero eigenvalues of K, hence removing K from both sides of (14) does not affect principal components.

3.1.2 Kernel centering

So far, we assumed that projected data set has zero mean. But in general it will not be the case. The standard way to centralize data set is to compute mean and then subtract it from every data point. Here we wish to avoid working in feature space and express everything in terms of kernel function. Let's denote projected data set after centering as $\widetilde{\phi}(x_n)$.

$$\widetilde{\phi}(x_n) = \phi(x_n) - \frac{1}{N}\sum_{l=1}^{N} x_l \tag{16}$$

and the corresponding elements of the Gram matrix

$$\begin{aligned} K_{nm} = \widetilde{\phi}(x_n)^T\widetilde{\phi}(x_m) = \phi(x_n)^T\phi(x_m) - \sum_{l=1}^{N}\phi(x_n)^T\phi(x_l) - \\ \sum_{l=1}^{N}\phi(x_l)^T\phi(x_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}\phi(x_j)^T\phi(x_l) = \\ k(x_n, x_m) - \sum_{l=1}^{N}k(x_n, x_l) - \sum_{l=1}^{N}k(x_l, x_m) + \frac{1}{N^2}\sum_{j=1}^{N}\sum_{l=1}^{N}k(x_j, x_l) \end{aligned} \tag{17}$$

or in a matrix notation

$$\widetilde{K} = K - 1_N K - K 1_N + 1_N K 1_N \tag{18}$$

where $\widetilde{K}$ is centered kernel matrix, 1_N is $N \times N$ matrix in which every element equals to $\frac{1}{N}$ (Bishop , 2009). So, we are able to evaluate $\widetilde{K}$ using only kernel function.

3.1.3 Finding principal component projection

After solving the eigenvalue problem $\widetilde{K}\boldsymbol{\alpha}_i = \lambda_i N\boldsymbol{\alpha}_i$, we can find projection onto principal components in terms of the kernel function. Using (11), projection of a point x onto eigenvector i is given by

$$\phi(x)^T\boldsymbol{v}_i = \sum_{n=1}^{N}\alpha_{in}\phi(x)^T\phi(x_n) = \sum_{n=1}^{N}\alpha_{in}k(x, x_n) \tag{19}$$

Note that neither (9) nor (19) requires the $\phi(x)$ in explicit form, we only need their dot product to use kernel function without actually performing the map ϕ (Scholkopf , 1998).

Note that the number of nonzero eigenvalues cannot exceed the number of observations N, because the $M \times M$ covariance matrix in feature space has rank at most N ($N < M$). So, kPCA leads to eigenvector expansion of the $N \times N$ kernel matrix K (Bishop , 2009).

3.2 Implementation of kPCA

Main steps of kPCA algorithm:

1. Pick up a kernel function;
2. Construct $M \times M$ dot product kernel matrix K of original data set;
3. Compute centered kernel matrix $\widetilde{K} = K - 1_N K - K 1_N + 1_N K 1_N$;
4. Solve an eigenvalue problem $\widetilde{K}\boldsymbol{\alpha}_i = \lambda_i N \boldsymbol{\alpha}_i$;
5. Compute projections of any point x onto eigenvectors (principal components);

We managed to implement kPCA by ourselves. The source code can be found here.

3.3 Examples of kPCA performance

Here we want to demonstrate kPCA performance on specific examples.

Consider three centric clouds of points on plane (left-hand plot on Figure 6). Colors of the points are not known to algorithms and used merely for visualization purposes. As we can see the data is not linearly separable, so standard PCA won't work. Let's now map the data on higher dimension using the (Gaussian) radial basis function kernel, or RBF kernel, defined as

$$k(x, y) = e^{\frac{-||x-y||^2}{2\sigma^2}} \tag{20}$$

As a result in 3-D we obtain a representation as on right-hand plot of Figure 6.

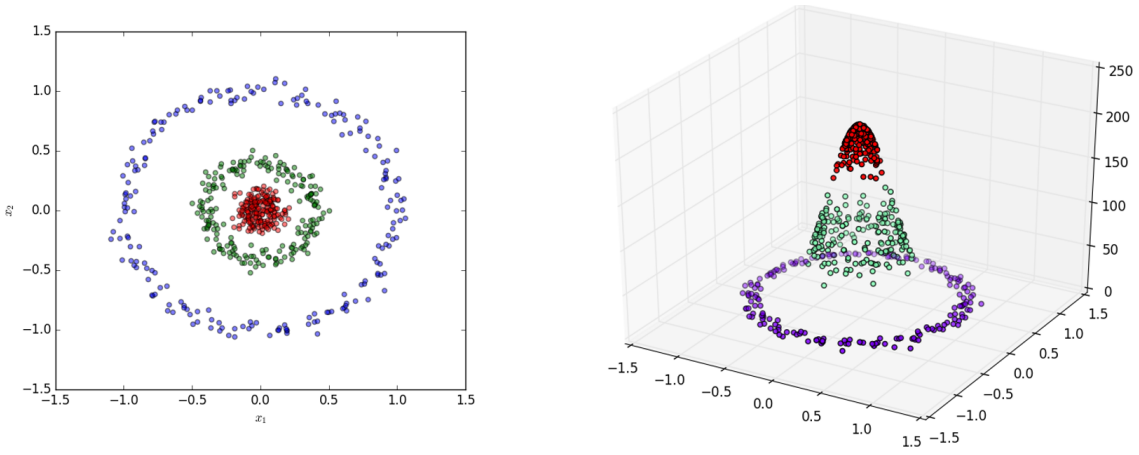

Figure 6: Three concentric clouds of points on 2-D and 3-D

We can see that in 3-D the points are linearly separable by a hyperplane. This is exactly what kPCA does.

The result of applying kPCA to the initial data set can be seen on left-hand plot of Figure 7. What noticeable is that we can pick out three groups of data merely along the first

principal component produced by kPCA. Whereas linear PCA (right-hand plot on Figure 7) operates only in the given (in our case 2-D) space, in which the concentric point clouds are not linearly separable.

The described capabilities of kernel PCA make it widely applicable in novelty detection (Hoffman , 2007) and image de-noising (Mika , 1999).

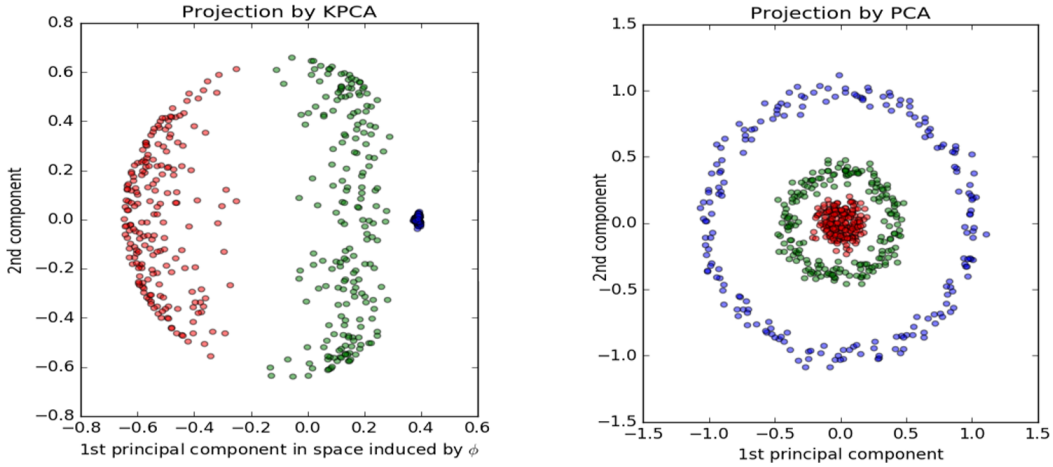

Figure 7: Comparison of applying PCA and kPCA to three concentric clouds of points

3.4 Pros and cons of kPCA

Kernel PCA has the next advantages:

- able to separate data in nonlinear case;
- as PCA, kPCA is an unsupervised method, so the number of principal components does not need to be specified in advance.

On the other hand, kPCA has the next drawbacks:

- the kernel matrix K is $N \times N$, so in case of big N storing K will be a problem and finding eigenvalues/eigenvectors will be a computationally complex operation.
- as PCA, kPCA is still vulnerable to such issues as outliers, noise and missing data.

4 Summary

In this project we have studied principal component analysis in details, discussed the areas of its application. We have identified the limitations of linear PCA and its possible modifications to overcome them. Then represented a comprehensive explanation of kernel PCA which extends PCA to non-linearly dependent input data.

References

Bishop, Christopher M.*Pattern Recognition and Machine Learning*. Springer, Cambridge, U.K., 2006.

Shlens, Jonathon. *A Tutorial on Principal Component Analysis.*Google Research, Mountain View, USA, 2014.

Candes, Emmanuel J.; Li, Xiaodong ; Ma, Yi; Wright, John. *Robust Principal Component Analysis?* J.ACM, 58(11):1-37, 2009

Lu, Haiping; Plataniotis, Konstantinos N.; Venetsanopoulos, Anastasios N. *MPCA: Multilinear Principal Component Analysis of Tensor Objects*. IEEE Transactions on Neural Networks, 19(1):18-39, 2008

Hastie, Trevor; Stuetzle, Werner. *Principal Curves*. Journal of the American Statistical Association, 84(406): 502-516, 1989.

Gorban, Alexander N.; Kegl, Balazs; Wunsch, Donald C.; Zinovyev, Andrei. *Principal Manifolds for Data Visualisation and Dimension Reduction*. Springer, Berlin – Heidelberg – New York, 2007.

Scholkopf, Bernhard; Smola, Alex; Muller, Klaus-Robert. *Nonlinear component analysis as a kernel eigenvalue problem.* Berlin, Germany., 1998.

Hoffmann, Heiko. *Kernel PCA for Novelty Detection*. Pattern Recognition, 40: 863-874, 2007.

Mika, Sebastian; Scholkopf, Bernhard. *Kernel PCA and De-Noising in Feature Spaces*. NIPS, 1999.