Kernel Principal Component Analysis

Bernhard Schölkopf[1], Alexander Smola[2], Klaus–Robert Müller[2]

[1] Max-Planck-Institut f. biol. Kybernetik, Spemannstr. 38, 72076 Tübingen, Germany
[2] GMD FIRST, Rudower Chaussee 5, 12489 Berlin, Germany

Abstract. A new method for performing a nonlinear form of Principal Component Analysis is proposed. By the use of integral operator kernel functions, one can efficiently compute principal components in high–dimensional feature spaces, related to input space by some nonlinear map; for instance the space of all possible d–pixel products in images. We give the derivation of the method and present experimental results on polynomial feature extraction for pattern recognition.

1 Introduction

Principal Component Analysis (PCA) is a basis transformation to diagonalize an estimate of the covariance matrix of the data $\mathbf{x}_k$, $k = 1, \ldots, \ell$, $\mathbf{x}_k \in \mathbf{R}^N$, $\sum_{k=1}^{\ell} \mathbf{x}_k = 0$, defined as

$$C = \frac{1}{\ell} \sum_{j=1}^{\ell} \mathbf{x}_j \mathbf{x}_j^{\top}. \tag{1}$$

The new coordinates in the Eigenvector basis, i.e. the orthogonal projections onto the Eigenvectors, are called *principal components*.

In this paper, we generalize this setting to a nonlinear one of the following kind. Suppose we first map the data nonlinearly into a feature space F by

$$\Phi : \mathbf{R}^N \to F, \quad \mathbf{x} \mapsto \mathbf{X}. \tag{2}$$

We will show that even if F has arbitrarily large dimensionality, for certain choices of Φ, we can still perform PCA in F. This is done by the use of kernel functions known from Support Vector Machines (Boser, Guyon, & Vapnik, 1992).

2 Kernel PCA

Assume for the moment that our data mapped into feature space, $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_\ell)$, is centered, i.e. $\sum_{k=1}^{\ell} \Phi(\mathbf{x}_k) = 0$. To do PCA for the covariance matrix

$$\bar{C} = \frac{1}{\ell} \sum_{j=1}^{\ell} \Phi(\mathbf{x}_j) \Phi(\mathbf{x}_j)^{\top}, \tag{3}$$

we have to find Eigenvalues $\lambda \geq 0$ and Eigenvectors $\mathbf{V} \in F \setminus \{0\}$ satisfying $\lambda \mathbf{V} = \bar{C}\mathbf{V}$. Substituting (3), we note that all solutions $\mathbf{V}$ lie in the span of $\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_\ell)$. This implies that we may consider the equivalent system

$$\lambda (\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot \bar{C}\mathbf{V}) \text{ for all } k = 1, \ldots, \ell, \tag{4}$$

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

$$\mathbf{V} = \sum_{i=1}^{\ell} \alpha_i \Phi(\mathbf{x}_i). \tag{5}$$

Substituting (3) and (5) into (4), and defining an $\ell \times \ell$ matrix K by

$$K_{ij} := (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)), \tag{6}$$

we arrive at

$$\ell \lambda K \boldsymbol{\alpha} = K^2 \boldsymbol{\alpha}, \tag{7}$$

where $\boldsymbol{\alpha}$ denotes the column vector with entries $\alpha_1, \ldots, \alpha_\ell$. To find solutions of (7), we solve the Eigenvalue problem

$$\ell \lambda \boldsymbol{\alpha} = K \boldsymbol{\alpha} \tag{8}$$

for nonzero Eigenvalues. Clearly, all solutions of (8) do satisy (7). Moreover, it can be shown that any additional solutions of (8) do not make a difference in the expansion (5) and thus are not interesting for us.

We normalize the solutions $\boldsymbol{\alpha}^k$ belonging to nonzero Eigenvalues by requiring that the corresponding vectors in F be normalized, i.e. $(\mathbf{V}^k \cdot \mathbf{V}^k) = 1$. By virtue of (5), (6) and (8), this translates into

$$1 = \sum_{i,j=1}^{\ell} \alpha_i^k \alpha_j^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)) = (\boldsymbol{\alpha}^k \cdot K \boldsymbol{\alpha}^k) = \lambda_k (\boldsymbol{\alpha}^k \cdot \boldsymbol{\alpha}^k). \tag{9}$$

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

$$(\mathbf{V}^k \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{\ell} \alpha_i^k (\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x})). \tag{10}$$

Note that neither (6) nor (10) requires the $\Phi(\mathbf{x}_i)$ in explicit form — they are only needed in dot products. Therefore, we are able to use kernel functions for computing these dot products *without* actually performing the map Φ (Aizerman, Braverman, & Rozonoer, 1964; Boser, Guyon, & Vapnik, 1992): for some choices of a kernel $k(\mathbf{x}, \mathbf{y})$, it can be shown by methods of functional analysis that there exists a map Φ into some dot product space F (possibly of infinite dimension) such that k computes the dot product in F. Kernels which have successfully been used in Support Vector Machines (Schölkopf, Burges, & Vapnik, 1995) include polynomial kernels

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^d, \tag{11}$$

radial basis functions $k(\mathbf{x}, \mathbf{y}) = \exp\left(-\|\mathbf{x} - \mathbf{y}\|^2 / (2\,\sigma^2)\right)$, and sigmoid kernels $k(\mathbf{x}, \mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \Theta)$. It can be shown that polynomial kernels of degree d correspond to a map Φ into a feature space which is spanned by all products of d entries of an input pattern, e.g., for the case of $N = 2, d = 2$,

$$(\mathbf{x} \cdot \mathbf{y})^2 = (x_1^2, x_1 x_2, x_2 x_1, x_2^2)(y_1^2, y_1 y_2, y_2 y_1, y_2^2)^\top. \tag{12}$$

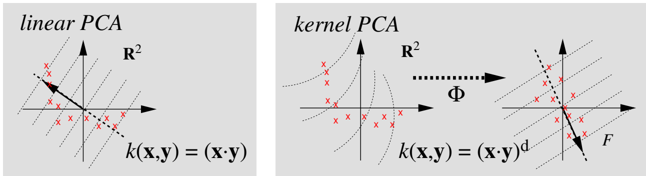

Fig. 1. Basic idea of kernel PCA: by using a nonlinear kernel function k instead of the standard dot product, we implicitly perform PCA in a possibly high–dimensional space F which is nonlinearly related to input space. The dotted lines are contour lines of constant feature value.

If the patterns are images, we can thus work in the space of all products of d pixels and thereby take into account higher–order statistics when doing PCA.

Substituting kernel functions for all occurences of $(\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y}))$, we obtain the following algorithm for kernel PCA (Fig. 1): we compute the dot product matrix (cf. Eq. (6)) $K_{ij} = (k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$, solve (8) by diagonalizing K, normalize the Eigenvector expansion coefficients α^n by requiring Eq. (9), and extract principal components (corresponding to the kernel k) of a test point $\mathbf{x}$ by computing projections onto Eigenvectors (Eq. (10), Fig. 2).

We should point out that in practice, our algorithm is not equivalent to the form of nonlinear PCA obtainable by explicitly mapping into the feature space F: even though the rank of the dot product matrix will be limited by the sample size, we may not even be able to compute this matrix, if the dimensionality is prohibitively high. For instance, 16×16 pixel input images and a polynomial degree $d = 5$ yield a dimensionality of 10^{10}. Kernel PCA deals with this problem by automatically choosing a subspace of F (with a dimensionality given by the rank of K), and by providing a means of computing dot products between vectors in this subspace. This way, we have to evaluate ℓ kernel functions in input space rather than a dot product in a 10^{10}–dimensional space.

To conclude this section, we briefly mention the case where we drop the assumption that the $\Phi(\mathbf{x}_i)$ are centered in F. Note that we cannot in general center the data, as we cannot compute the mean of a set of points that we do not have in explicit form. Instead, we have to go through the above algebra using $\tilde{\Phi}(\mathbf{x}_i) := \Phi(\mathbf{x}_i) - (1/\ell)\sum_{i=1}^{\ell} \Phi(\mathbf{x}_i)$. It turns out that the matrix that we

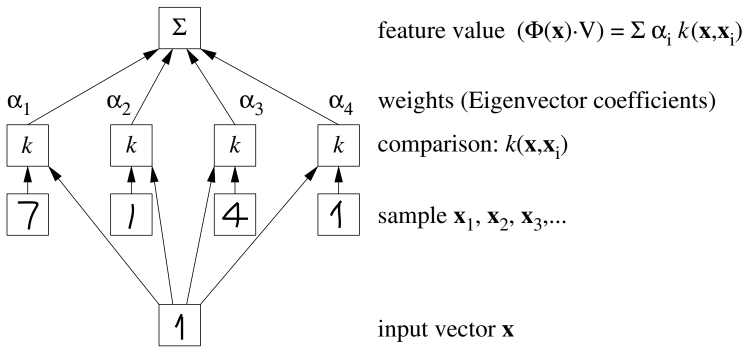

Fig. 2. Kernel PCA feature extraction for an OCR task (test point $\mathbf{x}$, Eigenvector $\mathbf{V}$).

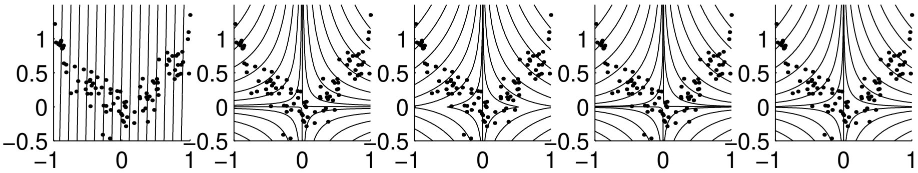

Fig. 3. PCA with kernel (11, degrees $d = 1, \ldots, 5$. 100 points $((x_i)_1, (x_i)_2)$ were generated from $(x_i)_2 = (x_i)_1^2 +$ noise (Gaussian, with standard deviation 0.2); all $(x_i)_j$ were rescaled according to $(x_i)_j \mapsto \mathrm{sgn}((x_i)_j) \cdot |(x_i)_j|^{1/d}$. Displayed are contour lines of constant value of the first principal component. Nonlinear kernels ($d > 1$) extract features which nicely increase along the direction of main variance in the data; linear PCA ($d = 1$) does its best in that respect, too, but it is limited to straight directions.

have to diagonalize in that case, call it $\tilde{K}$, can be expressed in terms of K as $\tilde{K}_{ij} = K - 1_\ell K - K 1_\ell + 1_\ell K 1_\ell$, using the shorthand $(1_\ell)_{ij} := 1/\ell$ (for details, see Schölkopf, Smola, & Müller, 1996[3]).

3 Experiments on Feature Extraction

Figure 3 shows the first principal component of a **toy data set**, extracted by polynomial kernel PCA. For an investigation of the utility of kernel PCA features for a realistic **pattern recognition problem**, we trained a separating hyperplane classifier (Vapnik & Chervonenkis, 1974; Cortes & Vapnik, 1995) on nonlinear features extracted from the US postal service (USPS) handwritten digit data base by kernel PCA. This database contains 9300 examples of dimensionality 256; 2000 of them make up the test set. For computational reasons, we used only a subset of 3000 training examples for the dot product matrix. Using polynomial kernels (11) of degrees $d = 1, \ldots, 6$, and extracting the first 2^n ($n = 6, 7, \ldots, 11$) principal components, we found the following. In the case of linear PCA ($d = 1$), the best classification performance (8.6% error) is attained for 128 components. Extracting the same number of *nonlinear* components ($d = 2, \ldots, 6$) in all cases lead to superior performance (around 6% error). Moreover, in the nonlinear case, the performance can be further improved by using a larger number of components (note that there exist more higher–order features than there are pixels in an image). Using $d > 2$ and 2048 components, we obtained around 4% error, which coincides with the best result reported for standard nonlinear Support Vector Machines (Schölkopf, Burges, & Vapnik, 1995). This result is competitive with convolutional 5–layer neural networks (5.0% were reported by LeCun et al., 1989); it is much better than linear classifiers operating directly on the image data (a linear Support Vector Machine achieves 8.9%; Schölkopf, Burges, & Vapnik, 1995). These findings have been confirmed on an object recognition task, the MPI chair data base (for details on all experiments, see Schölkopf, Smola, & Müller, 1996). We should add that our results were obtained without using any prior knowledge about symmetries of the problem at hand. This explains why the performance is inferior to Virtual Support Vector classifiers (3.2%, Schölkopf, Burges, & Vapnik, 1996), and Tangent Distance

[3] This paper, along with several Support Vector publications, can be downloaded from http://www.mpik-tueb.mpg.de/people/personal/bs/svm.html.

Nearest Neighbour classifiers (2.6%, Simard, LeCun, & Denker, 1993). We believe that adding e.g. local translation invariance, be it by generating "virtual" translated examples or by choosing a suitable kernel, could further improve the results.

4 Discussion

This paper was devoted to the exposition of a new technique for nonlinear principal component analysis. To develop this technique, we made use of a kernel method which so far only had been used in supervised learning (Vapnik, 1995). Clearly, the kernel method can be applied to *any* algorithm which can be formulated in terms of dot products exclusively, including for instance k-means and independent component analysis (cf. Schölkopf, Smola, & Müller, 1996).

In experiments comparing the utility of kernel PCA features for pattern recognition using a linear classifier, we found two advantages of nonlinear kernel PCA: first, nonlinear principal components afforded better recognition rates than corresponding numbers of linear principal components; and second, the performance for nonlinear components can be further improved by using more components than possible in the linear case.

The computational complexity of kernel PCA does *not* grow with the dimensionality of the feature space that we are implicitely working in. This makes it possible to work for instance in the space of all possible d-th order products between pixels of an image. As in the variant of standard PCA which diagonalizes the dot product matrix (e.g. Kirby & Sirovich, 1990), we have to diagonalize an $\ell \times \ell$ matrix (ℓ being the number of examples, or the size of a representative subset), with a comparable computational complexity — we only need to compute kernel functions rather than dot products. If the dimensionality of input space is smaller than the number of examples, kernel principal component extraction is computationally more expensive than linear PCA; however, this additional investment can pay back afterwards: we have presented results indicating that in pattern recognition, it is sufficient to use a linear classifier, as long as the features extracted are nonlinear. The main advantage of linear PCA up to date, however, consists in the possibility to reconstruct the patterns from their principal components.

Compared to other methods for nonlinear PCA, as autoassociative MLPs with a bottleneck hidden layer (e.g. Diamantaras & Kung, 1996) or principal curves (Hastie & Stuetzle, 1989), kernel PCA has the advantage that no nonlinear optimization is involved — we only need to solve an Eigenvalue problem as in the case of standard PCA. Therefore, we are not in danger of getting trapped in local minima during during training. Compared to most neural network type generalizations of PCA (e.g. Oja, 1982), kernel PCA moreover has the advantage that it provides a better understanding of what kind of nonlinear features are extracted: they are principal components in a feature space which is fixed a priori by choosing a kernel function. In this sense, the type of nonlinearities that we are looking for are already specified in advance, however this specification is a very wide one, it merely selects the (high–dimensional) feature space, but not the relevant feature subspace: the latter is done automatically. In this respect it is worthwhile to note that by using sigmoid kernels (Sec. 2) we can in fact also extract features which are of the same type as the ones extracted by MLPs (cf. Fig. 2), and the latter is often considered a nonparametric technique. With its rather wide class of admissible nonlinearities, kernel PCA forms a framework comprising various types of feature extraction systems. A number of different kernels have already been used in Support Vector Machines, of polynomial, Gaussian, and sigmoid type. They all led to high accuracy classifiers, and constructed their decision boundaries, which are hyperplanes in different feature spaces, from almost the same Support Vectors (Schölkopf, Burges, & Vapnik, 1995). The general question of how to choose the best kernel for a given problem is yet unsolved, both for Support Vector Machines and for kernel PCA.

PCA feature extraction has found application in many areas, including noise reduction, pattern recognition, regression estimation, and image indexing. In all cases where taking into account nonlinearities might be beneficial, kernel PCA provides a new tool which can be applied with little computational cost and possibly substantial performance gains.

Acknowledgements. BS is supported by the Studienstiftung des Deutschen Volkes. AS is supported by a grant of the DFG (JA 379/51). This work profited from discussions with V. Blanz, L. Bottou, C. Burges, S. Solla, and V. Vapnik. Thanks to AT&T and Bell Labsoratories for the possibility of using the USPS database.

References

M. A. Aizerman, E. M. Braverman, & L. I. Rozonoér. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.

B. E. Boser, I. M. Guyon, & V .Vapnik. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on COLT*, Pittsburgh, 1992. ACM.

C. Cortes & V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

T. Hastie & W. Stuetzle. Principal curves. *JASA*, 84:502 – 516, 1989.

M. Kirby & L. Sirovich. Application of the Karhunen–Loève procedure for the characterization of human faces. *IEEE Transactions*, PAMI-12(1):103–108, 1990.

E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biology*, 15:267–273, 1982.

B. Schölkopf, C. Burges, & V. Vapnik. Extracting support data for a given task. In U. M. Fayyad & R. Uthurusamy, eds., *Proceedings, First International Conference on Knowledge Discovery & Data Mining*, Menlo Park, CA, 1995. AAAI Press.

B. Schölkopf, C. Burges, & V. Vapnik. Incorporating invariances in support vector learning machines. In C. v. d. Malsburg, W. v. Seelen, J. C. Vorbrüggen, & B. Sendhoff, eds., *ICANN'96*, p. 47–52, Berlin, 1996. Springer LNCS Vol. 1112.

B. Schölkopf, A. J. Smola, & K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. Technical Report 44, Max–Planck–Institut für biologische Kybernetik, 1996. Submitted to *Neural Computation*.

P. Simard, Y. LeCun, & J. Denker. Efficient pattern recognition using a new transformation distance. In S. J. Hanson, J. D. Cowan, & C. L. Giles, editors, *Advances in NIPS 5*, San Mateo, CA, 1993. Morgan Kaufmann.

V. Vapnik & A. Chervonenkis. *Theory of Pattern Recognition [in Russian]*. Nauka, Moscow, 1974. (German Translation: W. Wapnik & A. Tscherwonenkis, *Theorie der Zeichenerkennung*, Akademie–Verlag, Berlin, 1979).

This article was processed using the LaTeX macro package with LLNCS style