Extending Kernel Principal Component Analysis to General Underlying Loss Functions

Carlos Alzate, Johan A. K. Suykens
Katholieke Universiteit Leuven, ESAT-SCD-SISTA
Kasteelpark Arenberg 10
B-3001 Leuven (Heverlee), Belgium
E-mail: {carlos.alzate, johan.suykens}@esat.kuleuven.ac.be

***Abstract*— Kernel Principal Component Analysis can be considered as a natural nonlinear generalization of PCA because it performs linear PCA in a kernel induced feature space. It allows us to extract nonlinear structures in the input data. The classical kernel PCA formulation leads to an eigendecomposition of the kernel matrix: eigenvectors with large eigenvalue correspond to the principal components in the feature space. Starting from the Least Squares Support Vector Machine (LS-SVM) formulation to kernel PCA we extend it to general underlying loss functions. For classical kernel PCA, the underlying loss function is L_2. In this approach, one can easily plug in other loss functions and solve a nonlinear optimization problem to achieve desirable properties. Simulations with Huber's loss function for robustness and quadratic epsilon insensitive loss function for sparseness demonstrate the flexibility of our approach.**

I. Introduction

Principal Components Analysis (PCA) is a powerful unsupervised learning technique widely used for feature extraction, denoising and compression [1]. It is a linear orthogonal basis transformation where the new basis is found by an eigendecomposition of the centered covariance matrix C of a dataset. One is interested in finding the directions in which the variance is maximal and each eigenvalue λ of C equals the amount of variance in the direction of the associated eigenvector. In this way, one can achieve de-noising by dropping directions with small variance or feature extraction by projecting onto the most informative directions (largest eigenvalue). Although PCA is a powerful technique, it cannot detect nonlinear structures in the input data [2]. Kernel PCA as a natural nonlinear generalization of PCA first maps the input data into some feature space F via a nonlinear feature map φ induced by a kernel and then performs linear PCA on the mapped data. The projections onto the subspace spanned by the eigenvectors now lie in F but we need the corresponding patterns in the input space. In order to recover the extracted features or the denoised patterns we have to map from F back to the input space (also called the pre-image problem [3]). This is a nontrivial problem because the pre-image may not exist, so we can only approximate it [3]. Several approaches have been developed for the pre-image problem [4], [3], [5]. In [6] it was shown that kernel PCA can be derived as the dual problem to a primal optimization problem formulated in a kernel induced feature space. In this paper we consider this formulation for general nonlinear loss functions. Within this formulation other loss functions can be applied to achieve desirable properties such as robustness or sparseness.

This paper is organized as follows: In Section II we describe the links between LS-SVM and kernel PCA. In Section III we extend the kernel PCA formulation to other loss functions. In Section IV we briefly describe two pre-image algorithms. In Section V we propose an algorithm to find the principal components with general loss function. In Section VI we report the empirical results for two loss functions and in Section VII we give conclusions.

II. LS-SVMs and Kernel PCA

LS-SVM formulations to different problems were discussed by Suykens *et al.* in [7]. It emphasizes primal-dual interpretations for a class of kernel machines in the context of constrained optimization problems. LS-SVMs have natural links with kernel Fisher discriminant analysis, kernel ridge regression, kernel canonical correlation analysis, kernel partial least-squares, recurrent networks, control and kernel PCA [7], [6].

A. Classical PCA formulation

In the classical PCA formulation, one considers a given set of zero mean data $\{x_k\}_{k=1}^N$. The objective is to find projected variables $w^T x$ with maximal variance:

$$\max_w Var(w^T x) \simeq \frac{1}{N}\sum_{k=1}^{N}(w^T x_k)^2 = w^T C w \qquad (1)$$

such that $w^T w = 1$, where C is the covariance matrix. The Lagrangian for this constrained optimization problem is given by:

$$\mathcal{L}(w;\lambda) = \frac{1}{2}w^T C w - \lambda(w^T w - 1) \qquad (2)$$

where λ is a Lagrange multiplier. Using the conditions for optimality: $\partial\mathcal{L}/\partial w = 0$, $\partial\mathcal{L}/\partial\lambda = 0$ one obtains the eigenvalue problem:

$$Cw = \lambda w. \qquad (3)$$

The eigenvector w corresponding to the largest eigenvalue defines the direction in which the projected variables have maximal variance.

B. LS-SVM Approach to Kernel PCA

Extending the LS-SVM interpretation of linear PCA to a high dimensional feature space, we have (See [6]):

$$\begin{aligned} \max_{w,e} J_p(w,e) &= \gamma \frac{1}{2}\sum_{k=1}^{N} e_k^2 - \frac{1}{2} w^T w \\ \text{such that} &\quad e_k = w^T(\varphi(x_k) - \hat{\mu}_\varphi),\ k = 1,...,N \end{aligned} \tag{4}$$

where $\hat{\mu}_\varphi = (1/N)\sum_{k=1}^{N}\varphi(x_k)$ and $\varphi(.) : \mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$ is the mapping to a high dimensional feature space. In this representation the role of the loss function is made explicit.

The Lagrangian becomes:

$$\begin{aligned} &\mathcal{L}(w,e;\alpha) \\ &= \gamma \frac{1}{2}\sum_{k=1}^{N} e_k^2 - \frac{1}{2} w^T w - \sum_{k=1}^{N}\alpha_k(e_k - w^T(\varphi(x_k) - \hat{\mu}_\varphi)) \end{aligned} \tag{5}$$

with conditions for optimality

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w} = 0 &\quad \rightarrow \quad w = \sum_{k=1}^{N}\alpha_k(\varphi(x_k) - \hat{\mu}_\varphi) \\ \frac{\partial \mathcal{L}}{\partial e_k} = 0 &\quad \rightarrow \quad \alpha_k = \gamma e_k,\ k = 1,...N \\ \frac{\partial \mathcal{L}}{\partial \alpha_k} = 0 &\quad \rightarrow \quad e_k - w^T(\varphi(x_k) - \hat{\mu}_\varphi) = 0,\ k = 1,...N. \end{aligned} \tag{6}$$

By elimination of e and w one obtains

$$\frac{1}{\gamma}\alpha_k - \sum_{l=1}^{N}\alpha_l(\varphi(x_l) - \hat{\mu}_\varphi)^T(\varphi(x_k) - \hat{\mu}_\varphi) = 0,\ k = 1,...,N. \tag{7}$$

Defining $\lambda = 1/\gamma$ we have the following dual problem

$$\Omega_c \alpha = \lambda \alpha \tag{8}$$

with centered kernel matrix given by Ω_c:

$$\Omega_{c,kl} = (\varphi(x_k) - \hat{\mu}_\varphi)^T(\varphi(x_l) - \hat{\mu}_\varphi),\ k,l = 1,...,N. \tag{9}$$

Applying the kernel trick one obtains:

$$\begin{aligned} &(\varphi(x_k) - \hat{\mu}_\varphi)^T(\varphi(x_l) - \hat{\mu}_\varphi) = \\ &K(x_k,x_l) - \frac{1}{N}\sum_{r=1}^{N}K(x_k,x_r) - \frac{1}{N}\sum_{r=1}^{N}K(x_l,x_r) \\ &+ \frac{1}{N^2}\sum_{r=1}^{N}\sum_{s=1}^{N}K(x_r,x_s). \end{aligned} \tag{10}$$

Hence (8) corresponds to kernel PCA as proposed by Schölkopf *et al.* in [2], where [6] showed that L_2 is the underlying loss function associated to it. It is also remarkable that Ω_c in (8) does not contain a regularization term while is present on the other hand in the primal problem in (4). The score variables become:

$$\begin{aligned} z(x) &= w^T(\varphi(x) - \hat{\mu}_\varphi) \\ &= \sum_{l=1}^{N}\alpha_l\Big(K(x_l,x) - \frac{1}{N}\sum_{r=1}^{N}K(x_r,x) \\ &- \frac{1}{N}\sum_{r=1}^{N}K(x_r,x_l) + \frac{1}{N^2}\sum_{r=1}^{N}\sum_{s=1}^{N}K(x_r,x_s)\Big). \end{aligned} \tag{11}$$

III. Extension to other loss functions

As discussed before, classical kernel PCA uses the L_2 loss function. Let us now extend the formulation to other loss functions:

$$\begin{aligned} \max_{w,e} J_p(w,e) &= \gamma \frac{1}{2}\sum_{k=1}^{N} L(e_k) - \frac{1}{2} w^T w \\ \text{such that} &\quad e_k = w^T(\varphi(x_k) - \hat{\mu}_\varphi),\ k = 1,...,N \end{aligned} \tag{12}$$

where $L(\cdot)$ is any differentiable loss function. The Lagrangian becomes:

$$\begin{aligned} &\mathcal{L}(w,e;\alpha) \\ &= \gamma \frac{1}{2}\sum_{k=1}^{N} L(e_k) - \frac{1}{2} w^T w - \sum_{k=1}^{N}\alpha_k(e_k - w^T(\varphi(x_k) - \hat{\mu}_\varphi)) \end{aligned} \tag{13}$$

with conditions for optimality:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial w} = 0 &\quad \rightarrow \quad w = \sum_{k=1}^{N}\alpha_k(\varphi(x_k) - \hat{\mu}_\varphi) \\ \frac{\partial \mathcal{L}}{\partial e_k} = 0 &\quad \rightarrow \quad \alpha_k = \gamma L'(e_k),\ k = 1,...N \\ \frac{\partial \mathcal{L}}{\partial \alpha_k} = 0 &\quad \rightarrow \quad e_k - w^T(\varphi(x_k) - \hat{\mu}_\varphi) = 0,\ k = 1,...N. \end{aligned} \tag{14}$$

Eliminating w, e leads to:

$$\alpha_k = \gamma L'\Big(\sum_{l=1}^{N}\alpha_l\Omega_c(x_l,x_k)\Big),\ k = 1,...,N \tag{15}$$

which is a set of nonlinear equations to be solved in α_k.

Rewriting (7) as:

$$f_k(\alpha) = \alpha_k - \gamma\ L'\Big(\sum_{l=1}^{N}\alpha_l\Omega_c(x_l,x_k)\Big) = 0,\ k = 1,...,N \tag{16}$$

and defining $f(\alpha) = [f_1(\alpha);\ldots;f_N(\alpha)]$ leads to:

$$f(\alpha) = \begin{bmatrix}\alpha_1 \\ \vdots \\ \alpha_N\end{bmatrix} - \gamma L'\left(\begin{bmatrix}\Omega_c(x_1,x_1) & \ldots & \Omega_c(x_1,x_N) \\ \vdots & \ddots & \vdots \\ \Omega_c(x_N,x_1) & \ldots & \Omega_c(x_N,x_N)\end{bmatrix}\begin{bmatrix}\alpha_1 \\ \vdots \\ \alpha_N\end{bmatrix}\right) = 0. \tag{17}$$

The roots of (17) are the principal components and (11) still holds.

IV. The Pre-Image Problem

The projections onto the subspace spanned by the principal components lie in some feature space F but in most applications of kernel PCA we are interested in a reconstruction in the input space rather than in F. This mapping from the feature space back to the input space is called the pre-image problem and is nontrivial because typically the exact pre-image does not exist and one can only approximate it [3]. An iterative nonlinear optimization method is proposed in [4] for Gaussian kernels, but it can suffer from numerical instabilities and local minima. Given a test point x we want to recover $\hat{x}$ for which $\varphi(\hat{x}) \approx P_{N_c}\varphi(x)$, where $P_{N_c}\varphi(x)$ is the projection of $\varphi(x)$ onto the subspace spanned by the first N_c principal components. For Gaussian kernels, the approximate pre-image $\hat{x}$ can be computed using the following iteration [4]:

$$\hat{x}_{t+1} = \frac{\sum_{k=1}^{N} \eta_k \exp(-\|\hat{x}_t - x_k\|^2/\sigma^2) x_k}{\sum_{k=1}^{N} \eta_k \exp(-\|\hat{x}_t - x_k\|^2/\sigma^2)} \tag{18}$$

where $\eta_k = \sum_{i=1}^{N_c} \beta_i u_k^i$ and β_i is the projection of $\varphi(x)$ onto the i-th component u^i. This approach will be referred to as "pre-image algorithm 1" in the sequel.

A newer approach is proposed in [5]. This method is non-iterative, involves only linear algebra and uses the distances between the projection $P_{N_c}\varphi(x)$ and its nearest neighbors. The method can be summarized as follows: given the N_n neighbors $\{x_1, \ldots, x_{N_n}\}$ and the input space distances between $P_{N_c}\varphi(x)$ and its N_n nearest neighbors $d^2 = [d_1^2, ...d_{N_n}^2]^T$ first construct the matrix $X = [x_1, x_2, \ldots, x_{N_n}]$ and compute its singular value decomposition (SVD):

$$XH = U\Lambda V^T = UZ \tag{19}$$

where H is the centering matrix, and $Z = [z_1, \ldots, z_{N_n}]$ with columns z_i being the projections of x_i onto U. The approximate pre-image can be obtained as:

$$\hat{z} = -\frac{1}{2}\Lambda^{-1}V^T(d^2 - d_0^2) \tag{20}$$

where $d_0^2 = [\|z_1\|^2, \ldots, \|z_{N_n}\|^2]^T$. Transforming back to the original coordinate system in the input space leads to:

$$\hat{x} = U\hat{z} + \bar{x} \tag{21}$$

where $\bar{x} = \frac{1}{N_n}\sum_{i=1}^{N_n} x_i$. This method will be referred to as "pre-image algorithm 2" in the sequel.

V. Algorithm

It is required that the principal components are mutually orthogonal. Therefore we can solve (17) using constrained nonlinear optimization to impose orthogonality constraints:

$$\min_{\alpha \in \mathbb{R}^N} f(\alpha)^T f(\alpha) \tag{22}$$

$$\text{such that } \begin{cases} A\alpha = 0 \\ \alpha^T\alpha = 1. \end{cases}$$

where A is the constraint matrix containing all previously found principal components in the rows. We need to provide the algorithm with initial values for α and also set γ. Kernel PCA eigenvectors are good starting points for the α vectors and γ is inversely related to the eigenvalues. Given a set of N $d-$dimensional data points $\mathcal{X} = \{x^1, \ldots, x^N\}$, the algorithm gives as output a set of N_c $N-$dimensional principal components: $\mathcal{A} = \{\alpha^1, ..., \alpha^{N_c}\}$ as follows:

1) Set the kernel parameters and the number of principal components N_c to be extracted.
2) Find the eigenvectors $\mathcal{U} = \{u^1, ..., u^{N_c}\}$ and eigenvalues $\mathcal{W} = \{\lambda_1, ..., \lambda_{N_c}\}$ of the centered kernel matrix Ω_c using singular value decomposition (SVD).
3) Calculate the first principal component α^1 by minimizing $f(\alpha)^T f(\alpha)$ with starting points $\alpha_{ini} = u^1$ and $\gamma_1 = 1/\lambda_1$ such that $(\alpha^1)^T\alpha^1 = 1$.
4) **for** $i = 2, ..., N_c$ **do**
   a) Add the previously found principal component to the constraint matrix $A \rightarrow A(i-1, :) = \alpha_{i-1}^T$.
   b) Compute the i-th principal component α^i by minimizing $f(\alpha)^T f(\alpha)$ with starting points $\alpha_{ini} = u^i$ and $\gamma_i = 1/\lambda_i$ such that α^i is orthogonal to all the previously found components$\rightarrow A\alpha^i = 0$ and $(\alpha^i)^T\alpha^i = 1$.

   end for
5) Normalize each component:

$$\alpha^i := \frac{\alpha^i}{\sqrt{\lambda_i}},\ i = 1, ..., N_c.$$

VI. Empirical Results

In this section, some experimental results using two different loss functions are presented. We used the MATLAB function *fmincon* to solve the constrained nonlinear optimization problem. First, Huber's loss function (Fig. 1) was chosen to include robustness into kernel PCA. In the context of robust statistics [8], [9], it is known that the L_2 loss function is not robust against outliers (data points that do not follow the distribution of the bulk of the data) because an arbitrary outlier can have a large influence on the solution. This is due to the fact that the influence function of L_2 is not bounded and increases linearly with the error. Huber's loss function has a bounded influence function so a single outlying data point cannot cause a large influence.

The second experiment consisted in using a quadratic epsilon-insensitive loss function [10] (Fig. 2) to achieve sparseness. With this loss function only a fraction of the principal component vectors will be non-zero which can be controlled with the epsilon parameter which controls the width of the zero region around the origin. In that case in (11) one only has to take the sum over the support vectors, which is also a computational advantage.

A. Experiment 1 - Robustness

In this experiment we report denoising results on 15 x 16 handwritten digits. The training set consisted of 200 images

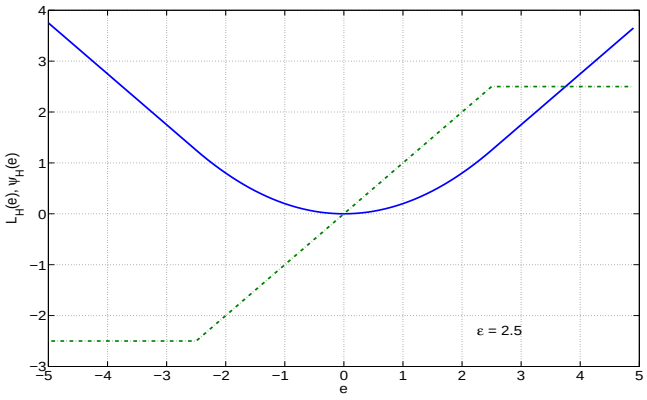


Fig. 1. Huber's loss function (solid line) with epsilon = 2.5 and its influence function (dashed line).

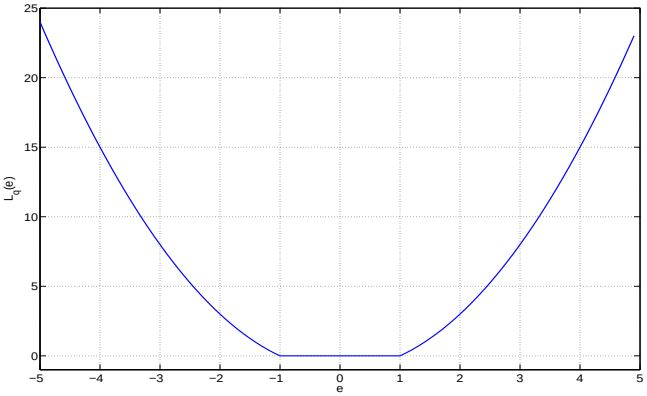


Fig. 2. Quadratic epsilon insensitive loss function with epsilon = 1.0.

(20 for each digit) with 22 outliers (from the Columbia image dataset [11]) (Fig. 3(a)). The denoised test digits using kernel PCA with robust Huber's loss function and two pre-image algorithms can be seen in (Fig. 3). From (Fig. 3(c)) one can see that the results from kernel PCA are not very accurate, nevertheless using the robust approach the denoised digits are visually more appealing. In Fig. (3(d)) kernel PCA with Huber's loss function outperforms the classical one. Table I shows the mean squared error (MSE) between the denoised test digits and the original test set. Table II shows the computation time in seconds.

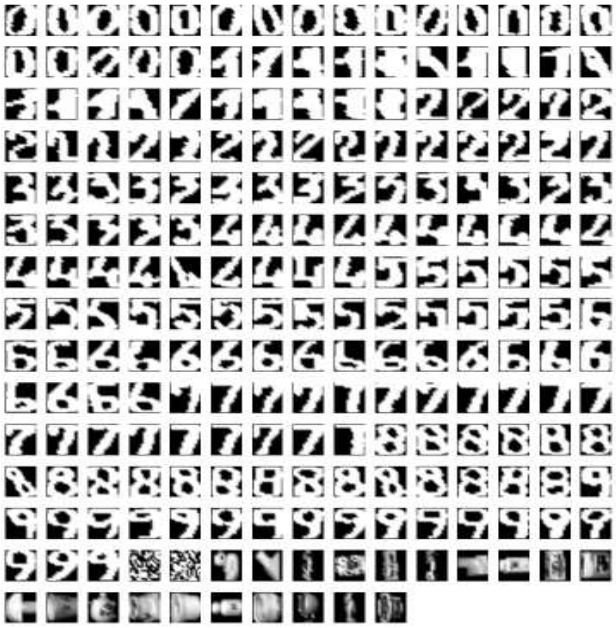
TABLE I

EXPERIMENT 1 MSE OF THE DENOISED TEST DIGITS

	Pre-image algorithm	
	Algorithm 1	Algorithm 2
Kernel PCA	0.29	0.17
Kernel PCA with Huber's loss function	0.21	0.07

B. Experiment 2 - Sparseness

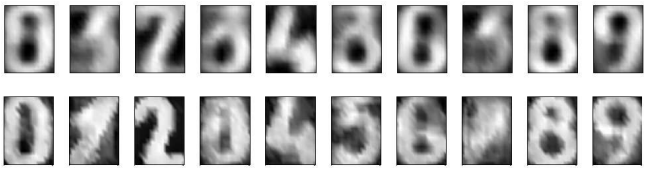
Here we considered two datasets: a simple curve of 500 data points and a square-like dataset of 600 data points. For the curve dataset we trained with 200 data points and kernel



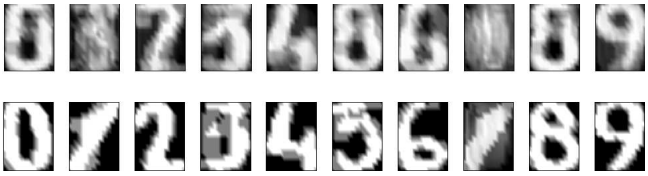
(a) Handwritten digit training set with outliers



(b) Test set corrupted with Gaussian noise and outliers imposed on each digit. *Top:* Original test set. *Bottom:* Corrupted test set.



(c) Denoised digits using pre-image algorithm 1.



(d) Denoised digits using pre-image algorithm 2. The number of nearest neighbors (N_n) was set to 5.

Fig. 3. Handwritten digits denoising. All simulations were done with RBF kernel $\sigma^2 = 50, N_c = 190, \epsilon = 15$. The top part of (c) and (d) corresponds to classical kernel PCA and the bottom part to kernel PCA with Huber's loss function.

TABLE II

EXPERIMENT 1 COMPUTATION TIME IN SECONDS FOR PRINCIPAL COMPONENTS EXTRACTION + DENOISING

	Pre-image algorithm	
	Algorithm 1	Algorithm 2
Kernel PCA	1.3	6.1
Kernel PCA with Huber's loss function	46.2	50.0

PCA with quadratic epsilon insensitive loss function with epsilon value of 0.8 and RBF kernel with $\sigma^2 = 0.2$. The denoising results using all 500 data points and the first 5 principal components can be seen in (Fig. 4). Table III shows the number of non-zero α_k for each principal component. The support vectors in the first three principal components are shown in (Figs. 5, 6, 7).

For the square dataset we trained with 200 data points and used an epsilon value of 1.8 and RBF kernel with $\sigma^2 = 0.1$. The denoising results using all 600 data points and 8 components are shown in (Fig. 8). The support vectors in the first principal component are shown in (Fig. 9). Table IV shows the number of non-zero α_k for each principal component.

TABLE III

EXPERIMENT 2 - CURVE - NUMBER OF NON-ZERO α_k FOR EACH PRINCIPAL COMPONENT

Principal Component	Number of non-zero α_k
First	92 out of 200
Second	86 out of 200
Third	34 out of 200
Fourth	49 out of 200
Fifth	87 out of 200

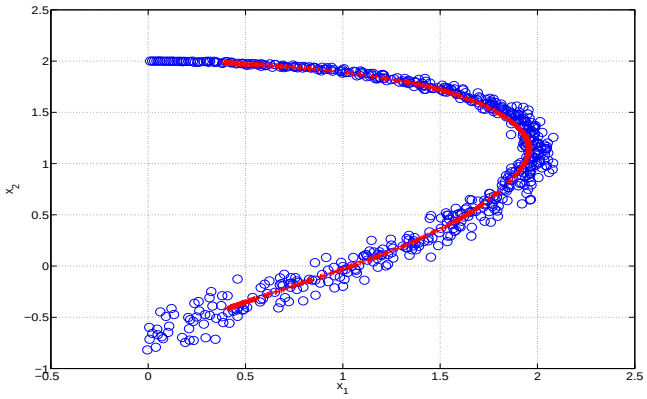


Fig. 4. Denoising results using kernel PCA with quadratic epsilon insensitive loss function and pre-image algorithm 1. Original variables are depicted as 'o' and reconstructed data points as '+'.

VII. CONCLUSION AND DISCUSSION

Starting from the LS-SVM context, a generalized form of kernel PCA is derived in order to include other loss

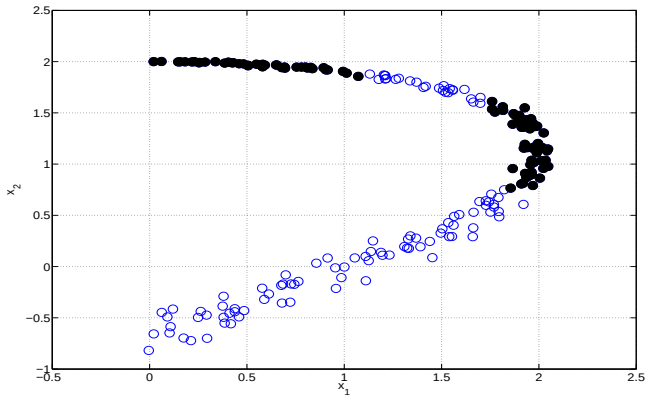


Fig. 5. Support vectors (filled circles) corresponding to the non-zero α_k in the first principal component.

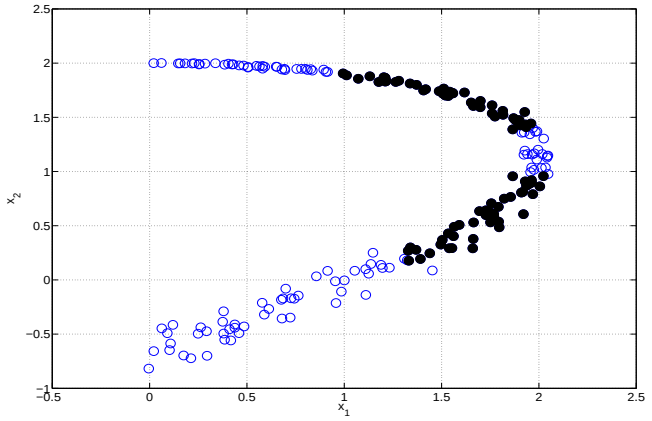


Fig. 6. Support vectors (filled circles) corresponding to the non-zero α_k in the second principal component.

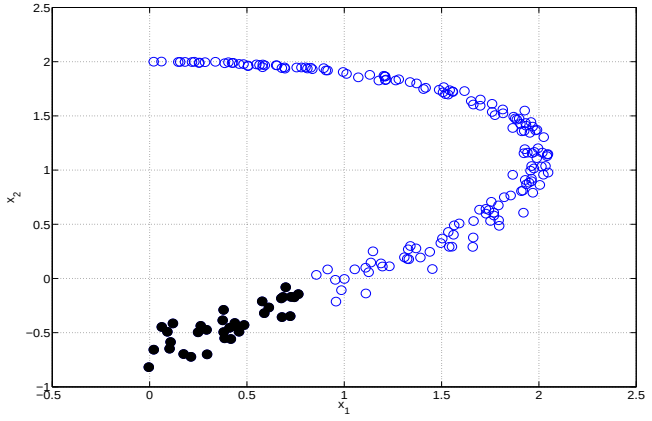


Fig. 7. Support vectors (filled circles) corresponding to the non-zero α_k in the third principal component.

TABLE IV

EXPERIMENT 2 - SQUARE - NUMBER OF NON-ZERO α_k FOR EACH PRINCIPAL COMPONENT

Principal Component	Number of non-zero α_k
First	110 out of 200
Second	84 out of 200
Third	58 out of 200
4th	65 out of 200
5th	78 out of 200
6th	108 out of 200
7th	62 out of 200
8th	131 out of 200

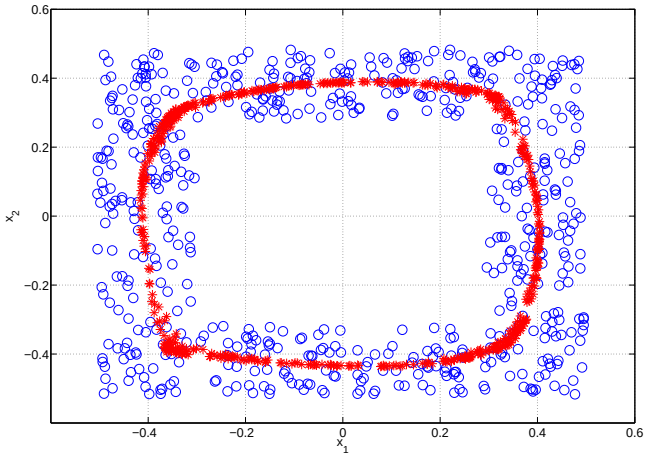


Fig. 8. Denoising results using kernel PCA with quadratic epsilon insensitive loss function and pre-image algorithm 1.

functions. In this representation, the kernel PCA loss function is made explicit so we can easily plug in other loss functions and solve a nonlinear optimization problem. From simulation results, our approach with Huber's loss function performed better than classical kernel PCA in the sense of robustness demonstrating some insensitivity to the outliers. In the case of quadratic epsilon insensitive loss function, the epsilon value controls the number of non-zero α_k in the principal components introducing sparseness into the formulation. With this sparse representation, the score variables can be calculated using only the support vectors, which is a computational advantage. Even though our approach produced good results and converged for most of the parameter values, its convergence is not guaranteed. However, we noticed that using kernel PCA eigenvectors/eigenvalues as initial values for the nonlinear optimization problem leads to quick convergence. For very small epsilon values in the first experiment, the algorithm tends to get stuck, because we are using a gradient-based optimization algorithm and for very small epsilon values Huber's loss function behaves as L_1 and is non-differentiable at the origin. When extracting few components, both kernel PCA and our approach take approximately the same time but when the number of components is large the difference in time will be noticeable.

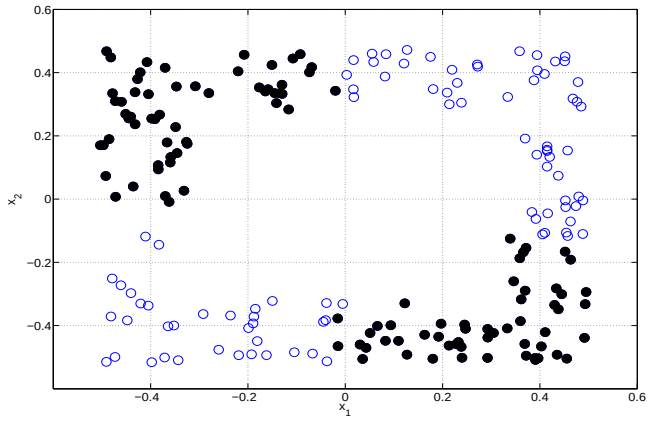


Fig. 9. Support vectors (filled circles) corresponding to the non-zero α_k in the first principal component.

ACKNOWLEDGMENTS

This work was supported by grants and projects for the Research Council K.U.Leuven (GOA-Mefisto 666, GOA-Ambiorics, several PhD/Postdocs & fellow grants), the Flemish Government FWO: PhD/Postdocs grants, projects G.0240.99, G.0407.02, G.0197.02, G.0080.01, G.0141.03, G.0211.05, G.0491.03, G.0120.03, G.0452.04, G.0499.04, ICCoS, ANMMM; AWI; IWT: PhD grants, GBOU (McKnow) Soft4s, the Belgian Federal Government (Belgian Federal Science Policy Office: IUAP V-22; PODO-II (CP/01/40), the EU(FP5-Quprodis;ERNSI, Eureka 2063-Impact; Eureka 2419-FLiTE) and Contracts Research/Agreements (ISMC/IPCOS, Data4s, TML, Elia, LMS, IPCOS, Mastercard). Johan Suykens is an associate professor at the K.U.Leuven, Belgium. The scientific responsibility is assumed by its authors.

REFERENCES

[1] I. T. Jolliffe, *Principal Component Analysis*. Springer Verlag, 1986.
[2] B. Schölkopf, A. Smola, and K. R. Muller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computation*, vol. 10, pp. 1299–1319.
[3] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K.-R. Müller, M. Scholz, G. Rätsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Networks*, vol. 10, no. 5, pp. 1000–1017, Sept. 1999.
[4] S. Mika, B. Schölkopf, A. J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch, "Kernel PCA and de–noising in feature spaces," in *Advances in Neural Information Processing Systems 11*, M. S. Kearns, S. A. Solla, and D. A. Cohn, Eds. MIT Press, 1999.
[5] J. Kwok and I. Tsang, "The pre-image problem in kernel methods," *IEEE Trans. Neural Networks*, vol. 15, no. 6, pp. 1517–1525, Nov. 2004.
[6] J. A. K. Suykens, T. Van Gestel, J. Vandewalle, and B. De Moor, "A support vector machine formulation to PCA analysis and its kernel version," *IEEE Trans. Neural Networks*, vol. 14, no. 2, pp. 447–450, Mar. 2003.
[7] J. A. K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. World Scientific, Singapore, 2002.
[8] P. Huber, *Robust Statistics*. John Wiley & Sons, 1981.
[9] P. J. Rousseeuw and A. M. Leroy, *Robust Regression and Outlier Detection*. Wiley New York (N. Y.), 1987.
[10] V. Vapnik, *Statistical Learning Theory*. Wiley, New York, 1998.
[11] S. Nene, S. Nayar, and H. Murase, "Columbia object image library: Coil," Columbia University," Technical Report CUCS-006-96, 1996.