Efficiently Updating and Tracking the Dominant Kernel Principal Components

Luc Hoegaerts Luc.Hoegaerts@esat.kuleuven.ac.be
Department of Electrical Engineering, ESAT-SCD-SISTA
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium

Lieven De Lathauwer Lieven.DeLathauwer@ensea.fr
ETIS, UMR 8051 (CNRS, ENSEA, UCP)
Avenue du Ponceau 6, BP 44, F 95014 Cergy-Pontoise Cedex, France

Ivan Goethals Ivan.Goethals@esat.kuleuven.ac.be
Department of Electrical Engineering, ESAT-SCD-SISTA
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium

J.A.K. Suykens Johan.Suykens@esat.kuleuven.ac.be
Department of Electrical Engineering, ESAT-SCD-SISTA
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium

Joos Vandewalle Joos.Vandewalle@esat.kuleuven.ac.be
Department of Electrical Engineering, ESAT-SCD-SISTA
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium

Bart De Moor Bart.DeMoor@esat.kuleuven.ac.be
Department of Electrical Engineering, ESAT-SCD-SISTA
Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium

Editor: Leslie Pack Kaelbling

Abstract

The dominant set of eigenvectors of the symmetric kernel Gram matrix is used in many important kernel methods (like e.g. kernel Principal Component Analysis, feature approximation, denoising, compression, prediction) in the machine learning area. Yet in the case of dynamic and/or large scale data, the batch calculation nature and computational demands of the eigenvector decomposition limit these methods in numerous applications. In this paper we present an efficient incremental approach for fast calculation of the dominant kernel eigenbasis, which allows to track the kernel eigenspace dynamically. Experiments show that our updating scheme delivers a numerically stable and accurate approximation for eigenvalues and eigenvectors at every iteration in comparison to the batch algorithm.

Keywords: Dominant eigenspace, eigenvalues, updating, tracking, kernel Gram matrix, Prinicipal Components, large scale data.

1. Introduction

Since the introduction of Support Vector Machines (SVM) (Vapnik, 1995) many learning algorithms are being transferred to a kernel representation (Schölkopf and Smola, 2002; Suykens et al., 2002). The important benefit lies in the fact that nonlinearities can be allowed, while avoiding to solve a nonlinear optimization problem. The transfer is implicitly accomplished by means of a nonlinear map into a Reproducing Kernel Hilbert Space H_k (RKHS) (Wahba, 1990).

When n data points $\mathbf{x}_i \in \mathbb{R}^p$ are given, a square symmetric $n \times n$ kernel Gram matrix K with $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ can be computed, where the kernel k provides a similarity measure between pairs of data points

$$k : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} : (\mathbf{x}_i, \mathbf{x}_j) \mapsto k(\mathbf{x}_i, \mathbf{x}_j). \tag{1}$$

A whole myriad of methods (e.g. Kernel Principal Components Analysis (KPCA) (Schölkopf et al., 1998b), Fixed Size Least Squares Support Vector Machines (FS-LSSVM) (Suykens et al., 2002), de-noising (Schölkopf et al., 1998a; Mika et al., 1999; Jade et al., 2003; Rosipal et al., 2001), data modelling (Twining and Taylor, 2003), etc.) rely, either directly or indirectly, on the dominant m eigenvectors of K, where $m \ll n$. The usual tool for computing this eigenspace is the Singular Value Decomposition (SVD) because of its high numerical accuracy (Golub and Loan, 1989). The drawback is that standard routines for the batch computation of an SVD require $O(n^3)$ operations (or $O(mn^2)$, for instance in deflation schemes (Golub and Loan, 1989)).

It is clear that for large datasets (as for example in image processing, computer vision or object recognition), the kernel method with its powerful advantage of dealing with nonlinearities, is computationally severely limited. For large datasets an eigendecomposition of K can simply become too time-consuming. But also memory is an issue as $O(n^2)$ memory units are required. In addition, computations may have to be repeated several times if the problem is nonstationary. Some applications require adaptive algorithms. A truly online algorithm for extracting kernel principal components has not been reported so far, although the problem is important and attempts to tackle it are presently undertaken (Rosipal and Girolami, 2001; Kim et al., 2003; Kuh, 2001).

The literature that spreads out over the last three decades has spent considerable effort on dealing with these issues, with a common denominator called SVD updating (Businger, 1970; Bunch and Nielsen, 1978; Moonen et al., 1992; Chandrasekaran et al., 1997; Levy and Lindenbaum, 2000; Badeau et al., 2004), the process of using the previous eigenspace when a new data instance arrives. Most of them are methods in which the eigenspace is incrementally updated when more rows (or more columns) are added to the involved matrix.

Yet, in the particular context of kernel algorithms, where the matrix dimensions correspond to the number of training data points, those updating schemes cannot be applied in a straightforward manner. The kernel matrix namely expands *both* in row and column dimension when a point is added. To our present knowledge, this case has not been addressed yet. In this paper we present a fast online updating and downdating procedure for the dominant eigenspace in case of a *growing* symmetric square matrix, with a computational complexity of $O(nm^2)$ and memory requirements of $O(nm)$ per iteration, and with very limited loss of accuracy. For the situation of subspace tracking, where one keeps the dimension of the subspace fixed, also an efficient downsizing mechanism is proposed.

In section 2 we propose an updating and downdating scheme for an existing eigenspace. In section 3 we develop an efficient way to downsize the row dimension of an updated eigenspace. In section 4 we present experiments on some typical benchmark data sets to demonstrate that accuracy and stability of the eigenvalues and eigenvectors are preserved during and after iteration. Section 5 summarizes the key results.

2. Updating and downdating

In this section we describe the updating in detail for one step of the iterative procedure and show that it compares favorably to the complexity of a batch SVD. Due to the particular structure of our updating scheme, downdating will just be a special case of updating.

2.1 Updating

For the update we assume that we have the (approximate) dominant $n \times m$ eigenspace U_{nm} of the square symmetric $n \times n$ kernel matrix K_n, based on n data points. We suppose that also an estimate for the number of dominant eigenvectors/values m is available. Obtaining m can be a subject entirely on itself, and is considered outside the scope of this paper.

The goal of updating is to obtain the $(n+1) \times m$ eigenspace $U_{(n+1)m}$ of the $(n+1) \times (n+1)$ kernel matrix K_{n+1}, based on the previous n data points and one extra data point. In the kernel matrix an update is located in the last row and column, which expands K_n *both* in row and column dimension when a point is added:

$$K_{n+1} = \begin{bmatrix} K_n & \mathbf{a} \\ \mathbf{a}^T & b \end{bmatrix} \tag{2}$$

where $\mathbf{a}$ is a $n \times 1$ vector of kernel entries $a_i = k(\mathbf{x}_i, \mathbf{x}_{n+1})$ and scalar $b = k(\mathbf{x}_{n+1}, \mathbf{x}_{n+1})$. In the following we provide a proof of an efficient formula for the dominant eigenspace of K_{n+1} based on the previously dominant eigenspace of K_n.

The key observation is that K_{n+1} can be expanded as a stripped matrix to which a difference of 2 rank-one matrices is added. We propose the following possible, convenient expansion:

$$K_{n+1} = \begin{bmatrix} K_n & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix} + \frac{1}{b} \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} \begin{bmatrix} \mathbf{a}^T & b \end{bmatrix} - \frac{1}{b} \begin{bmatrix} \mathbf{a} \\ 0 \end{bmatrix} \begin{bmatrix} \mathbf{a}^T & 0 \end{bmatrix} \tag{3}$$

where $\mathbf{0}$ is a $n \times 1$ vector of zeros.

Let us denote the given rank-m SVD of the submatrix $K_n \approx U_{nm} \Sigma_m U_{nm}^T$. We then simply substitute this in (3) and rewrite the expression with block matrices:

$$K_{n+1} \approx \begin{bmatrix} U_{nm} \\ \mathbf{0}^T \end{bmatrix} \Sigma_m \begin{bmatrix} U_{nm}^T & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{a} \\ b \end{bmatrix} \frac{1}{b} \begin{bmatrix} \mathbf{a}^T & b \end{bmatrix} - \begin{bmatrix} \mathbf{a} \\ 0 \end{bmatrix} \frac{1}{b} \begin{bmatrix} \mathbf{a}^T & 0 \end{bmatrix}. \tag{4}$$

We can then organize the separate vectors all together in a convenient symmetric factorization:

$$K_{n+1} \approx \begin{bmatrix} U_0 & A \end{bmatrix} \begin{bmatrix} \Sigma_m & O \\ O^T & D_u \end{bmatrix} \begin{bmatrix} U_0^T \\ A^T \end{bmatrix} \tag{5}$$

in which O represents the $n \times 2$ matrix of zeros, and in which we denote

$$U_0 := \begin{bmatrix} U_{nm} \\ \mathbf{0}^T \end{bmatrix}. \tag{6}$$

The $(n+1) \times 2$ matrix A contains the extra columns,

$$A := \begin{bmatrix} \mathbf{a} & \mathbf{a} \\ b & 0 \end{bmatrix}, \tag{7}$$

and the diagonal 2×2 matrix D contains the coefficients of the rank-1 updates,

$$D_u := \begin{bmatrix} 1/b & 0 \\ 0 & -1/b \end{bmatrix}. \tag{8}$$

With the above rearrangement we have obtained now an eigenspace, with a set of orthonormal eigenvectors in U_0, extended with the two columns of A, giving the matrix $\begin{bmatrix} U_0 & A \end{bmatrix}$. The two column vectors of A disturb the orthogonality of this matrix. To restore this orthogonality, we need to compute the part of A that is orthogonal to the eigenspace. So we must decompose the vectors of A into a component $A_{U_0^\perp}$ orthogonal and a component A_{U_0} parallel to the U_0:

$$A = A_{U_0^\perp} + A_{U_0} = (I_n - U_0 U_0^T)A + U_0 U_0^T A. \tag{9}$$

This splits up the extending contribution of A into an orthogonal component that increases the rank of the eigenspace, while the parallel component will cause the eigenvectors to be rotated, see Figure 1.

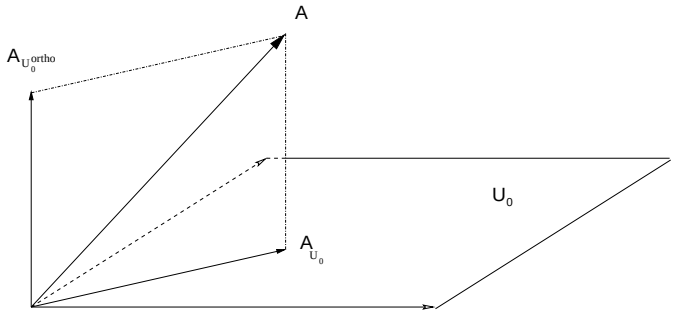


Figure 1: The previous eigenspace U_0 is being extended with information from A, which can be decomposed into a parallel and orthogonal component relative to U_0.

This leads to a factorization

$$\begin{bmatrix} U_0 & A \end{bmatrix} = \begin{bmatrix} U_0 & (I_n - U_0 U_0^T)A \end{bmatrix} \begin{bmatrix} I_m & U_0^T A \\ O^T & I_2 \end{bmatrix}. \tag{10}$$

To make the column vectors of $A_{U_0^\perp}$ mutually orthogonal, we compute the QR-decomposition

$$Q_A R_A \xleftarrow{\text{QR}} A_{U_0^\perp}, \tag{11}$$

in which $A_{U_0^\perp}$ is only of dimension $(n+1) \times 2$. This yields the QR decomposition

$$\left[\begin{array}{cc} U_0 & A \end{array}\right] = \left[\begin{array}{cc} U_0 & Q_A \end{array}\right] \left[\begin{array}{cc} I_m & U_0^T A \\ O^T & R_A \end{array}\right] =: Q_u R_u. \tag{12}$$

So far, only $O(nm)$ operations are needed for the decomposition of A into $A_{U_0^\perp}$ (if we perform multiplications in the most efficient order) and its QR-decomposition (Golub and Loan, 1989). In comparison with computing the QR factorization (12) from scratch, the number of computations has been limited by exploiting the mutual orthonormality of the columns of U_0. Substitution into (5) yields

$$K_{n+1} \approx \left[\begin{array}{cc} U_0 & Q_A \end{array}\right] \left[\begin{array}{cc} I_m & U_0^T A \\ O^T & R_A \end{array}\right] \left[\begin{array}{cc} \Sigma_m & O \\ O^T & D_u \end{array}\right] \left[\begin{array}{cc} I_m & O \\ A^T U_0 & R_A \end{array}\right] \left[\begin{array}{c} U_0^T \\ Q_A^T \end{array}\right]. \tag{13}$$

In order to obtain the optimal eigenvectors, a small SVD on the three middle matrices is necessary. The smallest singular value and corresponding eigenvector will then be discarded, which costs in total only a negligible $O(m^3)$ operations:

$$K_{n+1} \approx Q_u \left(R_u \left[\begin{array}{cc} \Sigma_m & O \\ O^T & D_u \end{array}\right] R_u^T \right) Q_u^T \overset{SVD}{=} Q_u V' \Sigma'_m V'^T Q_u^T. \tag{14}$$

It finally takes then $O(nm^2)$ operations to obtain an updated eigenspace

$$U' = Q_u V', \tag{15}$$

which is spent on the rotation of the orthonormal eigenspace Q_u according to V'. When the updating is followed by a downdating and downsizing, this last expensive operation may be postponed untill the end of the iteration step.

2.2 Downdating

For downdating, i.e. excluding the influence of a point, usually a forgetting factor is introduced for the previous eigenspace. This has on the one hand the advantage to smoothen data signal variations, but on the other hand it is only suitable for slowly varying processes. It is often more desirable to have a faster tracking response that can catch sudden signal changes, which requires that the net contribution of a point can be excluded (Basseville and Nikiforov, 1993).

The above key element in (3) of the updating scheme was that the expanded matrix can be written in terms of a stripped matrix to which a difference of 2 rank-one matrices is added. It supplies in fact immediately the right tool for performing a downdate, because in this case we only have other rank-one matrices:

$$\left[\begin{array}{cc} 0 & 0 \\ 0^T & K_n \end{array}\right] = \left[\begin{array}{cc} d & \mathbf{c}^T \\ \mathbf{c} & K_n \end{array}\right] - \frac{1}{d} \left[\begin{array}{c} d \\ \mathbf{c} \end{array}\right] \left[\begin{array}{cc} d & \mathbf{c}^T \end{array}\right] + \frac{1}{d} \left[\begin{array}{c} 0 \\ \mathbf{c} \end{array}\right] \left[\begin{array}{cc} 0 & \mathbf{c}^T \end{array}\right]. \tag{16}$$

We do not repeat the procedure for obtaining the eigenspace because it is completely analogous with the updating, only the matrices are slightly different.

However there is one difference concerning dimensions here. Whereas the row dimension of the eigenspace for an update grows from n to $n+1$, the reverse is not the case for a downdate. If the kernel matrix is exactly rank-n, then the first row of U' will be zero. However, in practice this will never be exactly the case. In the following section we propose a downsizing scheme to reduce the dimension of the kernel matrix.

3. Downsizing

As mentioned in the previous section, for each update the row dimension of the eigenspace matrix increases. This is not practical for on-line applications where one wants a subspace tracker, where the dimension of the eigenspace is to remain constant. This means we must downsize the matrix after each update, thereby preserving orthogonality. In this section we propose an efficient scheme to accomplish that and recover the orthogonal eigenspace, up to a unitary transformation.

In downsizing, the first row of U', obtained in (15) and denoted by the column vector v, must be deleted, such that we can continue with the resulting part V:

$$U' = \begin{bmatrix} v^T \\ V \end{bmatrix}. \tag{17}$$

In practice, the columns of V are not exactly mutually orthogonal. Orthogonality may be restored by means of a QR-decomposition of V, but this is expensive. In this section we will present a cheap approximate solution.

Consider

$$U'^T U' = I_m = \begin{bmatrix} v & V^T \end{bmatrix} \begin{bmatrix} v^T \\ V \end{bmatrix} = V^T V + \|v\|^2 \bar{v}\bar{v}^T \tag{18}$$

where $\bar{v} = v/\|v\|$.

Basically, we need a transformation matrix M such that V will be orthogonalized:

$$VM = Q \qquad \text{with} \qquad Q^T Q = I_m. \tag{19}$$

In this orthogonality condition we substitute (18) which yields

$$I_m = M^T V^T V M = M^T (I_m - \|v\|^2 \bar{v}\bar{v}^T) M. \tag{20}$$

Exploiting the structure, a convenient choice for M is

$$M = \begin{bmatrix} \bar{v} & \bar{v}_\perp \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{1-\|v\|^2}} & \mathbf{0}^T \\ \mathbf{0} & I_{m-1} \end{bmatrix} =: CJ, \tag{21}$$

where $\bar{v}_\perp$ is the $m \times (m-1)$ orthogonal complement of $\bar{v}^T$ such that $C^T C = I_m = CC^T$ and $\mathbf{0}$ is a $(m-1) \times 1$ vector of zeros. The computation of the matrix $\bar{v}_\perp$ costs a negligible number of $O(m^3)$ operations since it is the right nullspace of $\bar{v}^T$ (in Matlab implemented as $\text{null}(\bar{v}^T)$).

To conclude, the $(n+1) \times m$ eigenspace U' is replaced by the $n \times m$ eigenspace Q:

$$U' = \begin{bmatrix} v^T \\ V \end{bmatrix} \xrightarrow{\text{downsize}} Q := VM. \tag{22}$$

The total downsizing cost is dominated by this transformation itself, where M is multiplied with V from the left, which costs $O(nm^2)$ operations.

Of course the matrix with singular values is adapted accordingly to absorb the effect of the transformation matrix:

$$\Sigma'_m \to M^{-1}\Sigma'_m M^{-T}, \tag{23}$$

which has only a negligible cost of $O(m^3)$ operations. If the downsizing is preceded by an update and downdate, the SVD of those steps may both be postponed, to perform it only after downsizing. In this way only one SVD per iteration step is required for tracking.

The schemes of updating and downdating in combination with this downsizing method form a fast dominant eigenspace tracker algorithm that needs per step only $O(nm^2)$ operations and $O(nm)$ memory space. The overall algorithm is summarized in Table 1.

$K_n^{(k)}$ is a kernel matrix associated with data points $\mathbf{x}_k, \mathbf{x}_{k+1}, \ldots, \mathbf{x}_{k+n-1}$.
Given $U_{nm}^{(k)}\ \Sigma_m^{(k)}\ (U_{nm}^{(k)})^T \xleftarrow[m\text{-rank}]{\text{SVD}} K_n$.
$K_n^{(k+1)}$ is a kernel matrix associated with data points $\mathbf{x}_{k+1}, \mathbf{x}_{k+2}, \ldots, \mathbf{x}_{k+n}$.
Calculate $K_n^{(k+1)} \approx U_{nm}^{(k+1)}\ \Sigma_m^{(k+1)}\ (U_{nm}^{(k+1)})^T$:

- Update:
  - $U_0 = \begin{bmatrix} U_{nm}^{(k)} \\ \mathbf{0}^T \end{bmatrix}$
  - $Q_A R_A \xleftarrow{\text{QR}} (I - U_0 U_0^T) \begin{bmatrix} \mathbf{a} & \mathbf{a} \\ b & 0 \end{bmatrix}$
  - $Q_u = [U_0\ Q_A]$
  - $\Sigma_u = R_u \begin{bmatrix} \Sigma_m^{(k)} & O \\ O^T & D_u \end{bmatrix} R_u^T$
- Downdate:
  - $Q_B R_B \xleftarrow{\text{QR}} (I - Q_u Q_u^T) \begin{bmatrix} d & 0 \\ \mathbf{c} & \mathbf{c} \end{bmatrix}$
  - $Q_d = [Q_u\ Q_B]$
  - $\Sigma_d = R_d \begin{bmatrix} \Sigma_u & O \\ O^T & D_d \end{bmatrix} R_d^T$
- Downsize: $Q_d \rightarrow$ orthogonal eigenspace T and transformation M
- Perform SVD:
  - $W\ \Sigma_m^{(k+1)}\ W^T \xleftarrow{\text{SVD}} M^{-1} \Sigma_d M^{-T}$
  - discard last 4 singular values and vectors
- Perform rotation: $U_{nm}^{(k+1)} = TW$.

Table 1: Schematic overview of the proposed algorithm for tracking the dominant kernel kernel eigenspace.

4. Experiments

We implemented the proposed up/downdating and downsizing schemes in Matlab. A straightforward comparison of our (non-optimized) code with the built-in non-recursive Matlab SVDS routine shows that a computational gain is indeed achieved, especially for large dimensions (1000 and more) and few eigenvectors (order 10-100). Yet, in the experiments we primarily aim to characterize how well the true eigenvectors are approximated whilst tracking (i.e. up/downdating and downsizing) for the kernel matrix and only consider stationary data.

We considered three representative data sets that are typically assessed in the kernel based machine learning domain, on which kernel methods were applied which required the dominant kernel eigenspace. We employed the radial basis function (RBF) kernel, with h the kernel width parameter:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{h^2}\right). \tag{24}$$

In this experimental section we will illustrate that, in addition to reliable *tracking*, our algorithm can also yield the dominant eigenspace of a *very large* kernel matrix (say of order 10,000 $\times$ 10,000 or more), by consecutively applying updates on a given rank-m SVD of a kernel matrix.

4.1 Eigenspace Tracking

The Abalone benchmark data set (Blake and Merz, 1998) has $n = 3000$ training instances, with dimension $p = 7$. The RBF kernel width was fixed at 18.57. In Figure 2 the eigenspectrum of the symmetric kernel matrix K is shown, which has a quite steep exponentially decreasing behaviour, typical in most kernel based applications. The first 10 eigenvalues have a distinct interspacing and they cover 90% of the total mass of the energy. Based on this visual inspection we expect good tracking results for the eigenspace of dimension 500×8. We take some extra safety and track the m=20 first eigenvectors to compare it with the batch SVD.

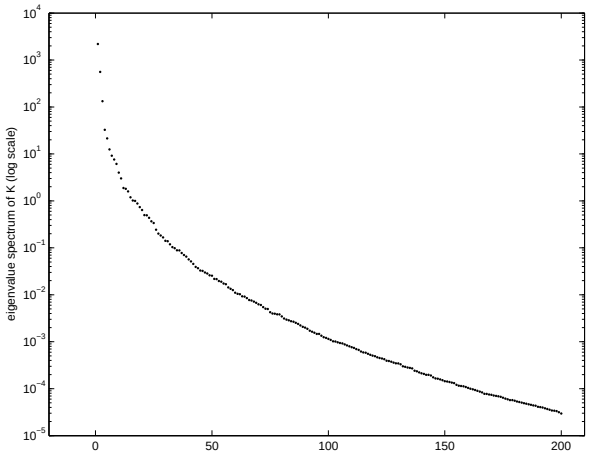


Figure 2: Eigenspectrum of the symmetric kernel matrix K for the Abalone benchmark data set, shown for the first 300 (of 3000) values.

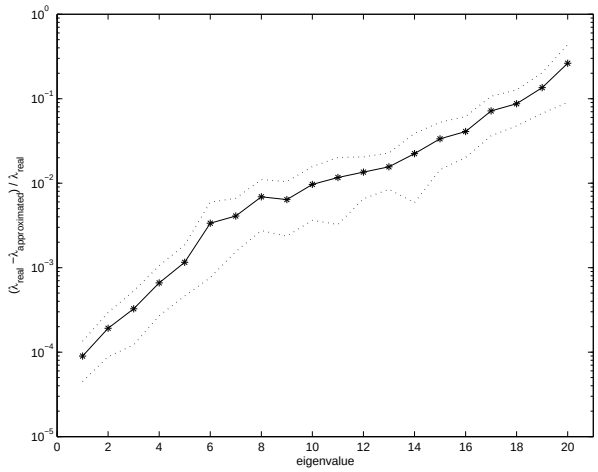


Figure 3: Relative error between the eigenvalues computed by SVD tracking and by batch SVD, averaged over 2500 iteration steps, with a standard deviation band equal to one (dotted), for the tracked 500×20 kernel matrix of the Abalone data set. The dominant (first 10) eigenvalues remain approximated well.

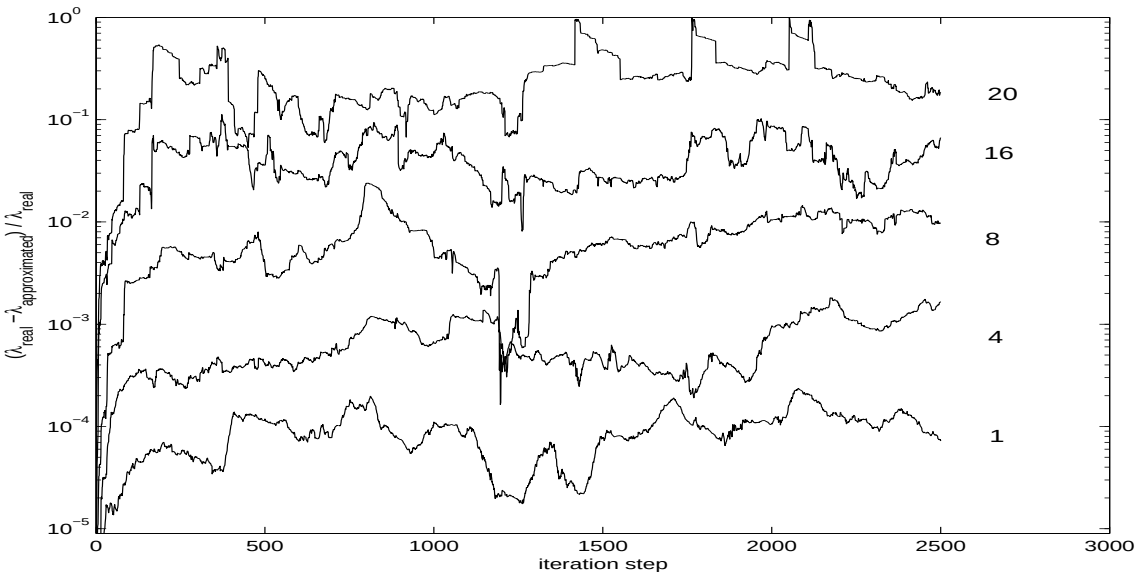


Figure 4: Relative error of the eigenvalues versus iteration step, obtained by SVD tracking, for some eigenvalues (as indicated by the number), for the tracked 500×20 kernel matrix of the Abalone data set. The dominant eigenvalues remain accurately approximated during the iteration process.

In order to assess the accuracy and stability of the tracking algorithm, we employed the following measures:

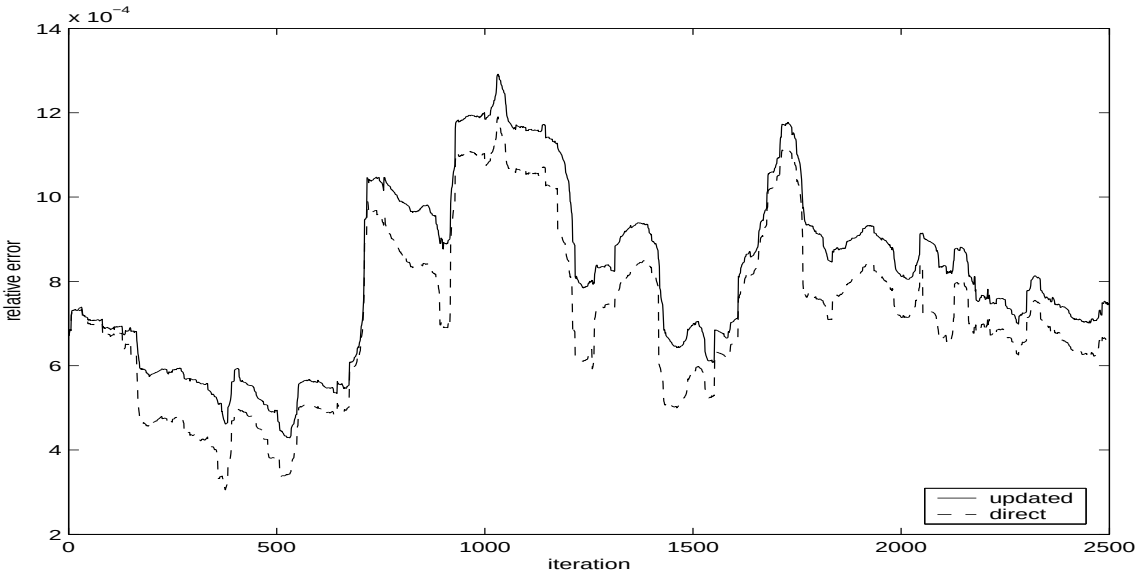


Figure 5: Relative error versus iteration step, shown for the tracked m-dimensional eigenspace and the m-rank eigenspace obtained by a batch SVD, for the tracked 500×20 kernel matrix of the Abalone data set. The two plotted lines follow each other closely, indicating good approximation of the eigenspace.

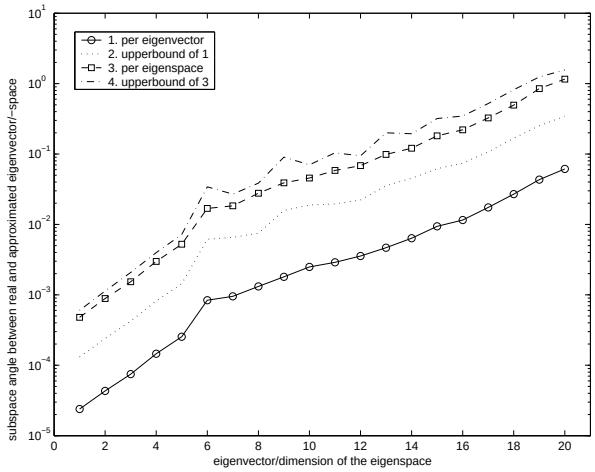


Figure 6: Subspace angle between the real and tracked eigen*vector*, shown per eigenvector (circles), with a standard deviation upper bound equal to one (dotted), for the tracked 500×20 kernel matrix of the Abalone data set. Subspace angle of the real and tracked eigen*spaces* (squares) versus the dimension of the eigenspace, with a standard deviation upper band equal to one (dash-dot).

- The relative error between the eigenvalue computed by SVD tracking $\tilde{\lambda}_i$ and the one by batch SVD λ_i is given by

$$E_\lambda^{rel} = \frac{\lambda - \tilde{\lambda}}{\lambda}. \tag{25}$$

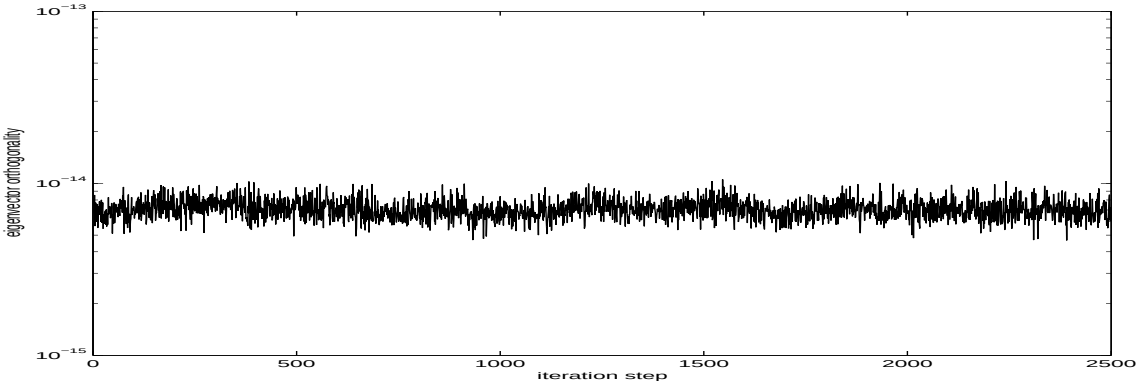


Figure 7: Orthogonality of the eigenvectors versus iteration step, for the tracked 500×20 kernel matrix of the Abalone data set. Smaller values indicate better preservation of mutual orthogonality between the eigenvectors.

In Figure 3 we depict the average of this measure over 2500 iteration steps, per eigenvalue and shown with a standard one-deviation band. One may observe that the first 10 dominant eigenvalues are very well approximated, with relative errors ranging from 0.01% up to 1%. The average accuracy decreases exponentially towards the last computed component, while the standard deviation increases accordingly. Only eigenvalues separated with a relatively large gap are being tracked well, which is inherent in an approximation scheme (Golub and Loan, 1989). Simple tests verified the fact that enlarging the column dimension of the eigenspace, decreases the relative error. Enlarging the row dimension has the same influence, but to a less extent.

In Figure 4 we show the relative error of the eigenvalues versus iteration step, for some eigenvalues (as indicated by the number) of the kernel matrix based on the Abalone data set. The dominant eigenvalues remain stable and are consistently approximated during the iterations from 0.001 up to 1 percent error.

- A good low-rank approximation delivers a faithful representation of the original matrix. Therefore, an overall measure of accuracy often used is the relative error based on a matrix norm:

$$E_\mu^{rel} = \frac{\|K - \tilde{K}\|_F}{\|K\|_F}, \tag{26}$$

where $\tilde{K}$ is a rank-m approximation of K and $\|\cdot\|_F$ denotes the Frobenius norm. In this case we have a low-rank approximation $\tilde{K} = \tilde{U}\tilde{S}\tilde{U}^T$ where $\tilde{U}$ is the eigenspace of K and $\tilde{S}$ is the matrix with singular values on the diagonal, both resulting from our algorithm. In our experiments we compare the relative error of the rank-m approximation obtained by our algorithm and the best m-rank approximation obtained by the batch SVD. Figure 5 shows for the Abalone data set example that the tracked eigenspace approximates the batch eigenspace very closely and consistently.

- The subspace angles (Golub and Loan, 1989) between the dominant eigenspaces of $\tilde{K}$ and K are direct indicators of how close these eigenspaces are. Figure 6 shows subspace

angles between eigen*vectors* versus the eigenvector number (circles), for the kernel matrix, based on the Abalone data set. Also shown is the subspace angle between the eigen*spaces* versus the dimension of the eigenspaces (squares). The subspace angle can range from zero (totally linearly dependent and coincident) to $\pi/2$ (orthogonal). The subspace angle value of 10^{-2} corresponds to deviations of half a degree, so the dominant (first 10) eigenvectors are quite well approximated for the largest part, taking into account the upper band of a standard deviation equal to one (dotted). The approximation is found to be closely connected to the gap between the successive eigenvalues. From the moment the eigenspectrum becomes flatter (between eigenvalue 5 and 10), the corresponding errors increase.

- Another important criterion concerns the orthogonality of the eigenspace during successive iterations. This can be monitored by the scalar orthogonality measure

$$E_\perp = \|I - \tilde{U}^T \tilde{U}\|_2. \tag{27}$$

A zero value indicates perfect orthogonality. In Figure 7 we show the typical evolution of this measure per iteration step, for the Abalone data set. In all experiments the orthogonality is very well-preserved and shows values in the range $\left[10^{-13}, 10^{-15}\right]$. When the row dimension is relatively small, reorthogonalization (by e.g. QR) may be necessary at certain intervals. In practice one can compare the orthogonality between the first and last eigenvector, for instance by taking the inner product and comparing against some preset tolerance. This renders the algorithm numerically more robust.

A second large data set considered for tracking is the Adult UCI data set (Blake and Merz, 1998), which consists of $45,222$ cases having 14 input variables. The aim is to classify if the income of a person is greater than 50K based on several census parameters, such as age, education, marital status, etc. We standardized the data to zero mean and unit variance. We picked at random a training set of size $n = 33000$. We used a Gaussian kernel with $h^2 = 29.97$ derived from preliminary experiments.

The spectrum for the whole set can not be calculated with present standard PC equipment, but estimates for different subsets indicate that 50 eigenvalues will suffice for capturing most of the energy of the spectrum. We tracked therefore a 500×50 dimensional eigenspace. The tracked spectra show on average a flat behaviour for the last 30 eigenvalues. The first 14 eigenvalues are most distinct and are covering 90% of the eigenvalue spectrum, having an average relative eigenvalue error level of 0.05%.

The relative error, defined in (26), for the rank-14 approximation obtained by our algorithm and the batch computation over 32500 iterations differs by $(1.2 \pm 0.2)\%$. Tracking a 1000×50 subspace improves this down to a $0.8 \pm 0.3)\%$ error difference, which indicates reliable close tracking. Also orthogonality remains well-preserved in the range $\left[10^{-12}, 10^{-14}\right]$, and in total 6 additional re-orthogonalizations (by a QR routine) were performed. Also on this data set we obtain a very good tracking accuracy for the dominant eigenspace and eigenvalues. We conclude that, given the fact that the spectrum is continuous, the tracking results of our algorithm are accurate.

4.2 Updating

In case when the dominant eigenspace of a very large matrix is needed (not fitting into memory or taking too many hours to compute), our updating procedure offers a considerably faster

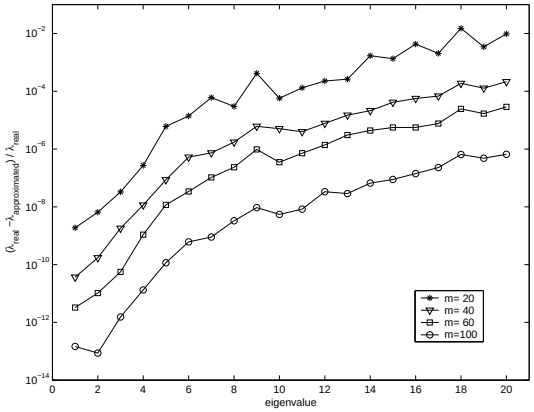


Figure 8: Relative errors for each of the 20 first eigenvalues, for different initial eigenbasis dimensions m=20,40,60,100 (while n was fixed at 1000), for the full kernel matrix of the Abalone data set. The updating procedure delivers eigenvalues with high accuracy.

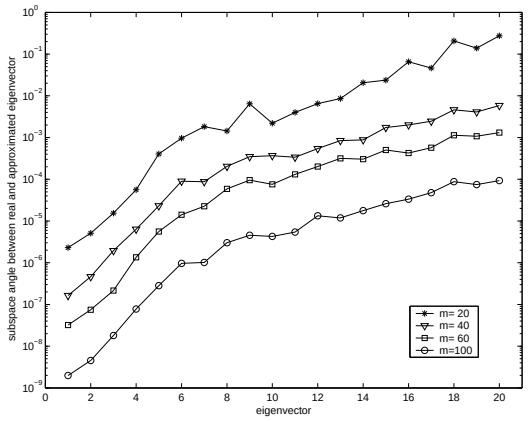


Figure 9: Subspace angles of the 20 first eigenvectors, for different initial eigenbasis dimensions m=20,40,60,100 (while n was fixed at 1000), for the full kernel matrix of the Abalone data set. The updating procedure delivers eigenvectors with high accuracy.

solution. The data is consecutively added to some initial m-rank SVD decomposition and one obtains the $n \times m$-sized dominant eigenspace and eigenvalues with a minimum of calculations. Again high accuracy rates are obtained, as experiments demonstrate.

In Figure 8 we plotted the relative errors for each of the 20 first eigenvalues, for different initial eigenbasis dimensions, with n fixed at 1000 and m varied over 20, 40, 60, 100. One may observe very low error values never exceeding one percent. In comparison these values are better than the tracking results because no downsizing is involved in updating, which incurs some loss of numerical precision.

In Figure 9 we assess the subspace angles of the eigenvectors in comparison with those obtained by batch SVD. Again one may observe that the eigenvectors are very well approximated: the deviations from the dominant real ones are significantly smaller than one degree.

Table 2 shows relative errors, defined in (26), and their ratios for different initial eigenbasis dimensions and approximation ranks. From the ratio for the rank-m approximation we see that the difference can vary up to a few percent, while the dominant rank-8 approximation are upper bounded by $10^{-5}\%$ ratios. Orthogonality was on average preserved up to values of 10^{-14}. We conclude that all measures demonstrate a good to excellent level of numerical precision for the obtained eigenspace and eigenvalues, which is related to the gap between the eigenvalues.

A third, large data set was used to demonstrate the updating results on the commonly known and widely applied USPS digits data (Guyon et al., 1989). This set consists of normalized handwritten digits, automatically scanned from envelopes by the U.S. Postal Service. There are 7291 instances of dimension 256.

Given that the RAM memory of the available computer is 1.5 Gigabytes, the SVD routine can be just fit into memory without any memory problems from the Matlab compiler (contrary to the SVDS routine). The full batch SVD computation takes about 20 hours on a 1GHz

		relative error				
		rank-m approximation			rank-8 approximation	
initial n	m	updating	batch	ratio	updating	ratio
500	20	0.0133082	0.0132063	1.0077	0.050801115068	1.000002791643
500	40	0.0050548	0.0047341	1.0677	0.050800975593	1.000000033648
1000	20	0.0132701	0.0132063	1.0048	0.050801161542	1.000003693981
1000	40	0.0050505	0.0047341	1.0668	0.050800975200	1.000000025900
1000	60	0.0025403	0.0024522	1.0359	0.050800973968	1.000000001648
1000	100	0.0008069	0.0007887	1.0230	0.050800973884	1.000000000004

Table 2: Relative errors for the rank-m and rank-8 approximations of the full kernel matrix, obtained by the SVD updating algorithm and the batch SVD, for the Abalone data set. The close to one ratios indicate good preservation of accuracy. The result of the batch rank-8 approximation is 0.050800973884.

PC, while a computation of the 100-dimensional subspace with eigenvalues takes 30 minutes with our algorithm. We note that the algorithm was a straightforward non-optimized Matlab script. More efficient coding (for instance in C) can reduce the computation time to a couple of minutes.

We used the RBF kernel with $h = 16$ (Schölkopf et al., 1999). The eigenspectrum may be quite steep, but no distinct gap is identifiable. The first 100 eigenvalues were taken on a visual inspection (covering 65% of the total mass of the eigenvalue spectrum, one would need an aditional 1550 eigenvalues to reach 90%).

In Table 3 we report on the relative error of the rank-100 approximation (in Frobenius norm, with $\|K\|_F = 1.9097e + 03$) and the mean of the relative error of the corresponding eigenvalues. The very close to one ratios, with deviations around one percent or less, indicate that we have again very good coincidence with the batch eigensubspace. Also the eigenvalues remain well-approximated within 2 percent, due to a slight bias of underestimation. We may conclude that even starting with a 100×100 SVD leads to good results for the dominant eigenspace and eigenvalue estimation of a 7291×7291 square kernel matrix. Orthogonality remained well-preserved in the range $\left[10^{-14}, 10^{-15}\right]$, and no re-orthogonalizations were required.

We conclude that the measures for the accuracy of the approximated dominant eigenspace and eigenvalues of a large kernel matrix indicate very good and stable agreement with those computed by batch solutions.

initial n	rel. error of rank-100 approximation		avg. rel. error
	updating	ratio	of eigenvalues
100	0.03273108890634	1.01538	2.22
500	0.03270590442933	1.01469	2.09
1000	0.03269606721720	1.01431	1.80
1500	0.03261049908786	1.01160	1.44
2000	0.03253083860862	1.00917	1.08
2500	0.03245233729252	1.00673	0.79
3000	0.03241386375776	1.00554	0.64

Table 3: Relative errors for the rank-100 approximations of the full kernel matrix, obtained by the SVD updating algorithm and the batch SVD, for the USPS data set with $h = 16$. The close to one ratios indicate good preservation of accuracy. The result of the batch rank-100 approximation is 0.0322351367302. The average relative error of the eigenvalues amounts to 2 percent or less. Both eigenvectors and eigenvalues are thus well-approximated by the proposed algorithm.

5. Conclusions

This paper introduced a novel up- and downdating algorithm for the dominant eigenspace of a square large scale symmetric matrix, in which the adaptation simultaneously occurs both in the row and columns. Additionally, a downsizing mechanism was proposed such that the algorithm is also capable of tracking the dominant eigenspace (while the dimension remains constant).

The dominant eigenspace of such matrix is relevant for several kernel based methods in machine learning (Suykens et al., 2002) for tasks in classification, pattern recognition, function estimation, system identification and signal processing.

Although these kernel methods are powerful models for dealing with nonlinearities, they are severely limited when it comes to large scale applications. In that case the computation times often become unacceptable or the kernel matrix cannot even be completely stored in memory. Furthermore there is no true on-line kernel principal component analysis algorithm known that can handle time-varying data. Our algorithm fills this gap, in an efficient manner with a time complexity of $O(nm^2)$ and memory requirements of $O(nm)$ per iteration.

In the experiments we focused on the accuracy and stability of the algorithm and found very good to excellent approximation of the eigenvalues and eigenvectors, with preservation of orthogonality and small relative error for the resulting low-rank approximations.

Acknowledgments

This research work was carried out at the ESAT laboratory of the Katholieke Universiteit Leuven. It is supported by grants from several funding agencies and sources: Research Council KU Leuven: Concerted Research Action GOA-Mefisto-666 (Mathematical Engineering), GOA-Ambiorics, IDO (IOTA Oncology, Genetic networks), PhD/postdoc & fellow grants; Flemish Government: Fund for Scientific Research Flanders (PhD/postdoc grants, projects G.0407.02 (support vector machines), G.0256.97 (subspace), G.0499.04 (robust statistics), G.0211.05 (nonlinear identification), G.0080.01 (collective behaviour), G.0115.01 (bio-i & microarrays), G.0240.99 (multilinear algebra), G.0197.02 (power islands), research communities ICCoS, ANMMM), AWI (Bil. Int. Collaboration Hungary/ Poland), IWT (Soft4s (softsensors), STWW-Genprom (gene promotor prediction), GBOU-McKnow (Knowledge management algorithms), Eureka-Impact (MPC-control), Eureka-FLiTE (flutter modeling), PhD grants); Belgian Federal Government: DWTC (IUAP IV-02 (1996-2001) and IUAP V-10-29 (2002-2006) and IUAP V-22 (2002-2006): Dynamical Systems & Control: Computation, Identification & Modelling), Program Sustainable Development PODO-II (CP/40: Sustainibility effects of Traffic Management Systems); Direct contract research: Verhaert, Electrabel, Elia, Data4s, IPCOS. Luc Hoegaerts is a PhD student supported by the Flemish Institute for the Promotion of Scientific and Technological Research in the Industry (IWT). Lieven De Lathauwer holds a permanent research position with the French Centre National de la Recherche Scientifique (C.N.R.S.); he also holds a honorary research position with the K.U.Leuven, Leuven, Belgium. Ivan Goethals is a PhD student supported by the Fund for Scientific Research Flanders (FWO-Vlaanderen). Johan Suykens is an associate professor at KU Leuven, Belgium. Joos Vandewalle and Bart De Moor are full professors at KU Leuven, Belgium.

References

R. Badeau, G. Richard, and B. David. Sliding window adaptive SVD algorithms. *IEEE Transactions on Signal Processing*, 52(1):1–10, january 2004.

M. Basseville and I. Nikiforov. *Detection of Abrupt Changes - Theory and Application*. Prentice-Hall, Inc., April 1993.

C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. URL `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

J. R. Bunch and J. P. Nielsen. Updating the singular value decomposition. *Numer. Math.*, 31:111–129, 1978.

P. Businger. Updating a singular value decomposition. *BIT*, 10:376–385, 1970.

S. Chandrasekaran, B. S. Manjunath, Y. F. Wang, J. Winkeler, and H. Zhang. An eigenspace update algorithm for image analysis. *Graphical models and image processing: GMIP*, 59 (5):321–332, 1997.

G. H. Golub and C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, Maryland 21218, 2nd edition, 1989.

I. Guyon, I. Poujaud, L. Personnaz, and G. Dreyfus. Comparing different neural network architectures for classifying handwritten digits. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN-89)*, volume II, pages 127–132, Washington, D.C, 1989. IEEE Press, New York.

A. M. Jade, B. Srikanth, V.K. Jayaraman, B.D. Kulkarni, J.P. Jog, and L. Priya. Feature extraction and denoising using kernel PCA. *Chemical Engineering Science*, 58(19):4441–4448, October 2003.

K. In Kim, M.O. Franz, and B. Schölkopf. Kernel Hebbian algorithm for iterative kernel principal component analysis. Technical Report 109, Max-Planck-Institut für biologische Kybernetik, Tübingen, June 2003.

A. Kuh. Adaptive kernel methods for CDMA systems. In *Proc. of the International Joint Conference on Neural Networks (IJCNN 2001)*, pages 1404–1409, Washington DC, 2001.

A. Levy and M. Lindenbaum. Sequential Karhunen-Loeve basis extraction. *IEEE Transactions on Image processing*, 9(8):1371–1374, 2000.

S. Mika, B. Schölkopf, A.J. Smola, K.-R. Müller, M. Scholz, and G. Rätsch. Kernel PCA and de-noising in feature spaces. *Advances in Neural Information Processing Systems*, 11:536 – 542, 1999.

M. Moonen, P. Van Dooren, and J. Vandewalle. An SVD updating algorithm for subspace tracking. *SIAM J. Matrix Anal. Appl.*, 13(4):1015–1038, 1992.

R. Rosipal and M. Girolami. An expectation-maximization approach to nonlinear component analysis. *Neural Computation*, 13(3):505–510, 2001.

R. Rosipal, M. Girolami, L.J. Trejo, and A. Cichocki. Kernel PCA for feature extraction and de-noising in nonlinear regression. *Neural Computing & Applications*, 10(3):231–243, 2001.

B. Schölkopf, S. Mika, C.J.C. Burges, P. Knirsch, K.-R. Mller, G. Rätsch, and A.J. Smola. Input space versus feature space in kernel-based methods. *IEEE Transactions On Neural Networks*, 10(5):1000–1017, September 1999.

B. Schölkopf, S. Mika, A.J. Smola, G. Rätsch, and K.-R. Mller. Kernel PCA pattern reconstruction via approximate pre-images. Perspectives in Neural Computing, pages 147–152, Berlin, Germany, 1998a. Springer.

B. Schölkopf and A. Smola. *Learning with kernels*. MIT Press, 2002.

B. Schölkopf, A.J. Smola, and K.R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998b.

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.

C. J. Twining and C. J. Taylor. The use of kernel principal component analysis to model data distributions. *Pattern Recognition*, 36(1):217–227, January 2003.

V. N. Vapnik. *The Nature of Statistical Learning Theory*. Berlin: Springer-Verlag, 1995, 1995.

G. Wahba. *Spline Models for Observational Data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. SIAM, Philadelphia, PA, 1990.