Robust PCA methods for complete and missing data

Juha Karhunen

Aalto University School of Science[1], Dept. of Information and Computer Science, P.O. Box 15400, FI-00076 Aalto, Espoo, Finland.
Email: juha.karhunen@aalto.fi URL: http://users.ics.tkk.fi/juha/

September 26, 2011

Abstract In this paper, we consider and introduce methods for robust principal component analysis (PCA), including also cases where there are missing values in the data. PCA is a widely applied standard statistical method for data preprocessing, compression, and analysis. It is based on the second-order statistics of the data and is optimal for Gaussian data, but is often applied to data sets having unknown or other types of probability distributions. PCA can be derived from minimization of the mean-square representation error or maximization of variances under orthonormality constraints. However, these quadratic criteria are sensitive to outliers in the data and long-tailed distributions, which may degrade the results given by PCA badly. We introduce robust methods for estimation of both the PCA eigenvectors directly or the PCA subspace spanned by them. Experimental results show that our methods provide often better results than standard PCA when outliers are present in the data. Furthermore, we extend our methods to incomplete data with missing values. The problems arising in such cases have several features typical for nonlinear models.

1 Introduction

Principal component analysis (PCA) is a well-known standard linear technique which is based on the second-order statistics of the data represented by the covariance matrix of the data. PCA is used widely especially as a preprocessing method for whitening or decorrelating the data and compressing it optimally in the mean-square error sense before application of more involved nonlinear data processing or classification methods. But sometimes applying PCA alone is sufficient for getting adequate results if the problem considered is simple enough. PCA is discussed from various viewpoints for example in [7, 9, 18, 20]. In the next section, we consider the mathematical definition of PCA, the optimization problems leading to PCA, and methods for computing it.

Robustness is a particularly important and desirable property of any method especially if the data contains exceptionally large values due to outliers, measurement errors, or impulsive noise [14, 15]. For example the standard mean-square and least-squares error criteria are not robust against such atypical values, which may cause large errors to the results or in extreme cases even render them useless.

[1]Up to the end of the year 2009 Helsinki University of Technology

This is easy to understand from a simple example. Assume for simplicity that we have at our disposal a data set for which the error e for the first sample to be considered is one measurement unit and for another sample 10 such measurement units. Using the mean-square or least-squares error criteria, the error for the latter sample gets a hundredfold (10^2) weight compared for the error for the first sample (1^2) in computing the optimal solution. Thus in an extreme case a few very large errors due to outliers can determine the optimal solution almost solely, with many small errors corresponding to appropriate measurements having a small effect to it only. This is clearly an undesirable situation.

Robustness is often achieved by using instead of standard squared criteria or loss functions robust criteria which grow less than quadratically, and we also apply this strategy later on. To this end, many different robust loss functions have been introduced, see for example Table 2.1 in [9]. Absolute error $| e |$ is in principle a simple and natural first choice for a robust criterion function, but there are two problems associated with using it. First, it is much more difficult to handle mathematically than quadratic criteria, whose optimal solutions obtained using differentiation lead typically to simple linear equations. Another problem is that the derivative of the absolute error has a discontinuity at origin. Several of the robust loss or criterion functions listed in Table 2.1 in [9] suffer from the same problem.

Many authors have studied robust PCA using various approaches. Robust PCA methods are briefly reviewed in the introductory parts of the papers [12, 31], in which one can find more references on the topic. In particular, in [4, 12, 38] Bayesian probabilistic approaches have been developed for the problem. These methods require making assumptions on the distribution of the noise and outliers. In the papers [4] and [38], the authors assume Student's t-distribution, having longer tails than the Gaussian distribution, and derive an EM (expectation maximization) algorithm for this model. Gao [12] replaces Student's t-distribution with double exponential (Laplace) distribution, and uses a variational Bayesian approach, leading to an approximative EM algorithm.

Torre and Black [31] use robust M-estimation developed in statistics for developing the theory of robust subspace learning for linear models. In the early paper by Xu and Yuille [35], entire data vectors are regarded as outlier vectors. In many applications it is however more realistic to assume that only some components in a data vector are contaminated (that is, outliers), while most of their components are not outliers. Ding et al. [11] develop for robust PCA so-called R_1-norm, which is a kind of intermediate form between the standard Euclidean norm and L_1-norm.

Another problem appearing often in practical data sets is that there are missing values in the data vectors. General methods for handling missing values are presented in [2, 21]. Methods for treating missing values in PCA are discussed in Section 13.6 in [18]. We shall discuss them in more detail in Section 4. We have developed probabilistic methods for missing values in PCA, and applied them successfully to the huge Netflix data set in [26, 27]. This work is summarized in the long journal paper [17]. There one can find also a general discussion of the problems arising with missing values in PCA, demonstrating several features which are usually associated with nonlinear models, such as ovefitting and poor local minima.

There exist many papers on robust PCA methods and on the other hand on treating missing values in PCA, but very few authors have considered both these problems simultaneously. The only paper that we know is our own one [22]. There a variational Bayesian methods is developed for this problem, and applied to a difficult real-world data set with very good results. This Bayesian method is, however, mathematically involved and quite demanding. In this paper, we develop simpler neural approaches for this combined problem.

The remainder of this paper is organized as follows. In the next section, we discuss PCA from various viewpoints, which are then used in Section 3 for deriving several robust methods. In Section 4, we discuss how missing values can be handled in context with PCA and robust PCA. In the following section, we present experimental results for both artificially generated data for which the theoretically correct results are known, and for real-world forest fires data. The paper ends with conclusions in Section 6.

2 Principal Component Analysis

2.1 Definition and batch estimation

Denote the n-dimensional i:th data vector by $\mathbf{x}_i$. We assume that there are available a total of N such data vectors. They can be represented compactly using an $n \times N$ data matrix

$$\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N] \tag{1}$$

For simplicity, the data vectors are assumed to have zero mean vector $\mathbf{m}_x = \mathrm{E}[\mathbf{x}]$, where $\mathrm{E}[\cdot]$ denotes the expectation operator. If this is not the case, the mean vector $\mathbf{m}_x$ can be easily estimated from the data as the average of the data vectors and subtracted from them as a preprocessing step.

For zero mean data vectors $\mathbf{x}$, their covariance matrix is theoretically

$$\mathbf{C}_{xx} = \mathrm{E}[\mathbf{x}\mathbf{x}^T] \tag{2}$$

Its standard sample estimate is

$$\widehat{\mathbf{C}}_{xx} = \frac{1}{N}\mathbf{X}\mathbf{X}^T = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T \tag{3}$$

In principal component analysis (PCA) [7, 9, 16, 18], the data vectors are expressed in the basis of the eigenvectors $\mathbf{u}_j$, $j = 1, \ldots, n$, of the covariance matrix $\mathbf{C}_{xx}$:

$$\mathbf{x}_i = \sum_{j=1}^{n}(\mathbf{x}_i^T\mathbf{u}_j)\mathbf{u}_j = \mathbf{U}\mathbf{U}^T\mathbf{x}_i \tag{4}$$

where the matrix

$$\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_n] \tag{5}$$

has as its columns the eigenvectors of the covariance matrix $\mathbf{C}_{xx}$.

In practice, the theoretical PCA eigenvectors are usually not known, but they must be estimated from the data (1) in a way or another. Denote these estimates, computed for example from the sample covariance matrix (3), by $\widehat{\mathbf{u}}_j$. Furthermore, dimensionality is often compressed in context with the PCA expansion (4) from n to m, where m is often a small fraction of n only, by using only the principal eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_m$ corresponding respectively to the m largest eigenvalues $\lambda_1, \ldots, \lambda_m$, where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_m \geq \lambda_{m+1}$. Thus

$$\mathbf{C}_{xx}\mathbf{U} = \mathbf{U}\boldsymbol{\Lambda} \tag{6}$$

where the diagonal matrix

$$\boldsymbol{\Lambda} = \mathrm{diag}[\lambda_1, \ldots, \lambda_m] \tag{7}$$

contains the PCA eigenvalues in descending order.

The equations (4), (5), (6), and (7) hold for the estimated quantities $\widehat{\mathbf{u}}_j$, $\widehat{\lambda}_j$, $\widehat{\mathbf{U}}$, $\widehat{\mathbf{\Lambda}}$, and $\widehat{\mathbf{C}}_{xx}$ as well. Thus the truncated estimated PCA expansion of the data vector $\mathbf{x}_i$ is

$$\widehat{\mathbf{x}}_i = \sum_{j=1}^{m} (\mathbf{x}_i^T \widehat{\mathbf{u}}_j)\widehat{\mathbf{u}}_j = \widehat{\mathbf{U}}_m \widehat{\mathbf{U}}_m^T \mathbf{x}_i \tag{8}$$

where the matrix $\widehat{\mathbf{U}}_m = [\widehat{\mathbf{u}}_1, \ldots, \widehat{\mathbf{u}}_m]$ contains as its m columns the estimated m first principal (PCA) eigenvectors of the covariance matrix of the data. These estimates can be obtained from the eigendecomposition of the sample covariance matrix (3), or from various iterative and/or adaptive algorithms to be discussed in more detail later on.

There are two major classes of PCA methods: those which estimate the PCA eigenvectors themselves and those which estimate a PCA subspace only. If the column vectors of the matrix $\widehat{\mathbf{U}}_m$ are the first m estimated principal eigenvectors of the covariance matrix of the data, the approximation $\widehat{\mathbf{x}}_i$ in (8) is optimal for every $m = 1, 2, \ldots$ as discussed in the next subsections. In the PCA subspace approaches, the vectors $\widehat{\mathbf{u}}_j$ are some mutually orthonormal basis vectors of the m-dimensional PCA subspace spanned by the estimated first m PCA eigenvectors only [5, 25]. In this case, the expansion (8) is optimal for that dimensionality m of the PCA subspace only. There exist infinitely many bases of the PCA subspace, obtained by applying any $m \times m$ rotation matrix $\mathbf{Q}_m$ satisfying the property $\mathbf{Q}_m \mathbf{Q}_m^T = \mathbf{Q}_m^T \mathbf{Q}_m = \mathbf{I}$ to the matrix $\widehat{\mathbf{U}}_m$. Because $\widehat{\mathbf{U}}_m \mathbf{Q}_m (\widehat{\mathbf{U}}_m \mathbf{Q}_m)^T = \widehat{\mathbf{U}}_m \widehat{\mathbf{U}}_m^T$, the expansion (8) remains optimal in spite of the orthogonal transformation $\mathbf{Q}_m$. The true PCA eigenvectors can be computed from a basis of the corresponding PCA subspace as explained in Section 2.4 in [17].

2.2 PCA from variance maximization

PCA eigenvectors directly or the PCA subspace spanned by them can be derived as optimal solutions from several quadratic criteria. The first one is maximization of the variances of the basis vectors of the PCA subspace (assuming zero mean data $\mathbf{X}$) [16, 20]:

$$\mathcal{J}_{var} = \mathrm{E}[\| \mathbf{y} \|^2] = \mathrm{E}[\| \mathbf{W}^T \mathbf{x} \|^2] = \mathrm{tr}(\mathbf{W}^T \mathbf{C}_{xx} \mathbf{W}) \tag{9}$$

under the constraint that the column vectors $\mathbf{w}_j$ of the $n \times m$ weight matrix

$$\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m] \tag{10}$$

which constitute the basis vectors of the m-dimensional PCA subspace, are mutually orthonormal:

$$\mathbf{W}^T \mathbf{W} = \mathbf{I}. \tag{11}$$

There

$$\mathbf{y} = \mathbf{W}^T \mathbf{x} \tag{12}$$

is the m-dimensional feature vector which is in this case also the projection $[\mathbf{W}^T \mathbf{W}]^{-1} \mathbf{W}^T \mathbf{x}$ of $\mathbf{x}$ onto the m-dimensional PCA subspace due to the orthonormality constraints (11), and $\mathrm{tr}(\mathbf{A})$ denotes the trace of the matrix $\mathbf{A}$.

Taking into account the orthonormality constraints (11) via Lagrange multipliers, it is easy to derive the following Oja's PCA subspace rule for estimating the PCA subspace [16, 19, 20]:

$$\mathbf{\Delta W} = \mu[\mathbf{I} - \mathbf{W}\mathbf{W}^T]\mathbf{x}\mathbf{x}^T \mathbf{W} = \mu[\mathbf{x} - \mathbf{W}\mathbf{y}]\mathbf{y}^T \tag{13}$$

where μ denotes the learning parameter. The corresponding stochastic gradient algorithm utilizing one data vector $\mathbf{x}_k$ only at each iteration is then

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k \mathbf{e}_k \mathbf{x}_k^T \mathbf{W}_k = \mathbf{W}_k + \mu_k \mathbf{e}_k \mathbf{y}_k^T \tag{14}$$

where $\mathbf{y}_k = \mathbf{W}_k^T \mathbf{x}_k$, and the error vector

$$\mathbf{e}_k = \mathbf{x}_k - \mathbf{W}_k \mathbf{W}_k^T \mathbf{x}_k = \mathbf{x}_k - \mathbf{W}_k \mathbf{y}_k \tag{15}$$

Because Oja's PCA subspace rule (14)–(15) is a stochastic gradient algorithm, it converges slowly but anyway to some orthonormal basis of the PCA subspace spanned by the columns of the matrix $\mathbf{U}_m$ in (8). The corresponding algorithm for estimating the PCA eigenvectors $\mathbf{u}_i$ directly is the generalized Hebbian algorithm (GHA) made popular by Sanger, see [29, 19, 20].

A problem with the GHA algorithm and its robust and nonlinear generalizations [19, 20] is that they are deflation type algorithms. That is, the first PCA eigenvector is estimated first, with the estimate of the second PCA eigenvector depending on it, and so on for the subsequent estimates of PCA eigenvectors. Thus for example the estimate of the second PCA eigenvector cannot converge until the estimate of the first eigenvector has converged. This is the basic reason for the undesirable property which the deflation type PCA algorithms have, namely that the errors accumulate so that they are larger for the estimate of the next principal eigenvector than for the current one. This makes such deflation type algorithm inaccurate and slowly converging in estimating more than just a few principal eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_m$.

On the other hand, in the PCA subspace type algorithms like Oja's subspace rule (13) all the estimates of the basis vectors of the PCA subspace (column vectors of the weight matrix $\mathbf{W}$) are treated in a symmetric and equal way, which usually leads to a faster convergence and better accuracy than for the corresponding deflationary algorithms.

2.3 PCA from minimization of mean-square approximation error

Oja's PCA subspace rule can be derived also by minimizing the mean-square approximation error (MSE) [16, 19, 20, 34]

$$\mathcal{J}_{MSE} = \mathrm{E}[\| \mathbf{x} - \widehat{\mathbf{x}} \|^2] = \mathrm{E}[\| \mathbf{x} - \mathbf{W}\mathbf{W}^T \mathbf{x} \|^2] = \mathrm{E}[\| \mathbf{x} - \mathbf{W}\mathbf{y} \|^2] \tag{16}$$

Here

$$\widehat{\mathbf{x}} = \sum_{j=1}^{m} (\mathbf{x}^T \mathbf{w}_j) \mathbf{w}_j = \mathbf{W}\mathbf{W}^T \mathbf{x} = \mathbf{W}\mathbf{y} \tag{17}$$

is a similar approximation of the data vector $\mathbf{x}$ as (8) but now in terms of the m weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$.

It can be shown that the MSE error in (16) is minimized when the weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$ constitute some orthonormal basis of the m-dimensional PCA subspace [5, 25]. The mean-square error (16) is then the sum of discarded smallest eigenvalues of the data covariance matrix $\mathbf{C}_{xx}$:

$$\mathcal{J}_{MSE} = \mathrm{E}[\| \mathbf{x} - \widehat{\mathbf{x}} \|^2] = \sum_{j=m+1}^{n} \lambda_j \tag{18}$$

which can be compared to the total power of the data:

$$\mathrm{E}[\| \mathbf{x} \|^2] = \sum_{j=1}^{n} \lambda_j \tag{19}$$

The mean-square error (18) is often rather small compared with the total power (19) even though the dimensionality m of the PCA subspace is much smaller than the dimensionality n of the data, allowing one to drop the dimensionality of the data vectors quite significantly in data compression with a relatively small MSE error. This is turn brings out large computational savings in subsequent processing especially for methods whose computational load grows fast with the dimensionality of the data.

The mean-square error criterion (16) is a more appropriate starting point for deriving PCA solutions than the variance maximization criterion (9), because the orthonormality constraints (11) are not needed in deriving the optimum solution [5, 25].

Several authors (see [19, 20, 34]) have derived the following algorithm for minimizing the MSE error (16):

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k[\mathbf{x}_k\mathbf{e}_k^T\mathbf{W}_k + \mathbf{e}_k\mathbf{x}_k^T\mathbf{W}_k] \tag{20}$$

The first update term $\mathbf{x}_k\mathbf{e}_k^T\mathbf{W}_k$ is proportional to the same vector, the data vector $\mathbf{x}_k$, for all the weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_m$, and it is usually small. Therefore it can be in practice usually neglected [19, 20]. This approximation leads to standard Oja's subspace rule (14)–(15). It converges as quickly as the algorithm (20) to the same solution. Again, one can derive the respective hierarchic (deflationary type) algorithm for estimating the PCA eigenvectors directly, and approximating it in a similar manner as above leads to Sanger's GHA algorithm [20].

2.4 Steepest descent and Newton type algorithms

In practice, the mean-square approximation error in (16) is not known, but it must be replaced by its finite sample approximation based on the known data $\mathbf{X}$ in (1). Thus the data vectors are modeled as

$$\mathbf{x}_i = \mathbf{W}\mathbf{y}_i + \mathbf{e}_i = \widehat{\mathbf{x}}_i + \mathbf{e}_i \tag{21}$$

There $\mathbf{e}_i$ is the n-dimensional error vector incurred by approximating the n-dimensional data vector $\mathbf{x}_i$ by $\widehat{\mathbf{x}}_i = \mathbf{W}\mathbf{y}_i = \mathbf{W}\mathbf{W}^T\mathbf{x}_i$ which lies in the m-dimensional subspace spanned by the weight vectors $\mathbf{w}_j$, $j = 1, 2, \ldots, m$.

The data model (21) can be written compactly for the entire data $\mathbf{X}$ as

$$\mathbf{X} = \mathbf{W}\mathbf{Y} + \mathbf{E} \tag{22}$$

where the feature matrix $\mathbf{Y} = \mathbf{W}^T\mathbf{X}$ and error matrix $\mathbf{E}$ are defined by

$$\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N] \tag{23}$$

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_N] \tag{24}$$

Now the unknown mean-square error (16) is replaced by the squared Frobenius norm of the error matrix $\mathbf{E}$:

$$\begin{aligned} \mathcal{C} &= \| \mathbf{E} \|_F^2 = \| \mathbf{X} - \mathbf{W}\mathbf{Y} \|_F^2 = \sum_{j=1}^{N} \| \mathbf{x}_j - \mathbf{W}\mathbf{W}^T\mathbf{x}_j \|^2 \qquad (25) \\ &= \sum_{j=1}^{N} \| \mathbf{x}_j - \mathbf{W}\mathbf{y}_j \|^2 = \sum_{i=1}^{n}\sum_{j=1}^{N}(x_{ij} - \mathbf{b}_i^T\mathbf{y}_j)^2 = \sum_{i=1}^{n}\sum_{j=1}^{N}(x_{ij} - \hat{x}_{ij})^2 \end{aligned}$$

where $\mathbf{b}_i^T$ is the i:th m-dimensional row vector of the $n \times m$ weight matrix $\mathbf{W}$, and $\hat{x}_{ij} = \mathbf{b}_i^T \mathbf{y}_j$ is the approximation of the element x_{ij} of the data matrix $\mathbf{X}$.

In [17], Ilin and Raiko presented a variant of Oja's PCA subspace rule which can be derived by minimizing the error (25) with respect to both $\mathbf{W}$ and $\mathbf{Y}$, without using the relationship $\mathbf{Y} = \mathbf{W}^T\mathbf{X}$. Differentiating $\mathcal{C}$ with respect to $\mathbf{W}$ and $\mathbf{Y}$ leads to the steepest descent PCA subspace learning algorithm

$$\begin{aligned} \mathbf{W}_{k+1} &= \mathbf{W}_k - \mu_k \frac{\partial \mathcal{C}}{\partial \mathbf{W}} && (26) \\ \mathbf{Y}_{k+1} &= \mathbf{Y}_k - \mu_k \frac{\partial \mathcal{C}}{\partial \mathbf{Y}} && (27) \end{aligned}$$

where the elements of the partial derivative matrices are

$$\begin{aligned} \frac{\partial \mathcal{C}}{\partial w_{il}} &= -2\sum_{j=1}^{N}(x_{ij} - \hat{x}_{ij})y_{lj} && (28) \\ \frac{\partial \mathcal{C}}{\partial y_{lj}} &= -2\sum_{j=1}^{N}(x_{ij} - \hat{x}_{ij})w_{il} && (29) \end{aligned}$$

and $\hat{x}_{ij}$ is the element ij of the matrix $\widehat{\mathbf{X}} = \mathbf{W}\mathbf{Y}$.

This algorithm can be represented compactly in matrix form as

$$\begin{aligned} \mathbf{W}_{k+1} &= \mathbf{W}_k + \mu_k \mathbf{E}_k \mathbf{Y}_k^T &&= \mathbf{W}_k + \mu_k[\mathbf{X} - \mathbf{W}_k\mathbf{Y}_k]\mathbf{Y}_k^T && (30) \\ \mathbf{Y}_{k+1} &= \mathbf{Y}_k + \mu_k \mathbf{W}_k^T \mathbf{E}_k &&= \mathbf{Y}_k + \mu_k \mathbf{W}_k^T[\mathbf{X} - \mathbf{W}_k\mathbf{Y}_k] && (31) \end{aligned}$$

where the error matrix on iteration k is

$$\mathbf{E}_k = \mathbf{X} - \mathbf{W}_k\mathbf{Y}_k \qquad (32)$$

and it should be recalled that $\mathbf{Y}_k$ is treated in Eqs. (30)-(31) as an independent variable, without resorting to the relationship $\mathbf{Y} = \mathbf{W}^T\mathbf{X}$.

The convergence speed and stability of this algorithm can be improved by developing the corresponding Newton type algorithm [17]. In this Newton version, one first computes the second partial derivatives

$$\frac{\partial^2 \mathcal{C}}{\partial w_{il}^2} = \sum_{j=1}^{N} y_{lj}^2, \qquad \frac{\partial^2 \mathcal{C}}{\partial y_{lj}^2} = \sum_{i=1}^{n} w_{il}^2 \qquad (33)$$

The partial derivatives on the left hand side of equations (28) and (29) are then replaced by

$$\left(\frac{\partial^2 \mathcal{C}}{\partial w_{il}^2}\right)^{-1} \frac{\partial \mathcal{C}}{\partial w_{il}}, \qquad \left(\frac{\partial^2 \mathcal{C}}{\partial y_{lj}^2}\right)^{-1} \frac{\partial \mathcal{C}}{\partial y_{lj}} \qquad (34)$$

In order to keep the computational load smaller, only the diagonal elements of the Hessian matrices (33) are used especially in high-dimensional problems in practice.

A Newton version of the standard Oja's PCA subspace rule is obtained by using only the first equations (26) and (28), and replacing the partial derivative (28) by the first term in (34). In this case, the matrix $\mathbf{Y}_k$ is computed from $\mathbf{Y}_k = \mathbf{W}_k^T\mathbf{X}$.

2.5 Least-squares type algorithms for PCA

In 1995, Yang [36] introduced so-called PAST (Projection Approximation Subspace Tracking) algorithm for adaptive estimation of the basis vectors (10) of the PCA subspace. Using the same notation as above, in deriving the PAST algorithm the unknown standard mean square-error $\mathrm{E}[\| \mathbf{x} - \mathbf{W}\mathbf{W}^T\mathbf{x} \|^2]$ used in (16) for deriving PCA is replaced by the respective weighted least-squares error criterion

$$J_{WLS}(k) = \sum_{i=1}^{k} \beta^{k-i} \| \mathbf{x}_i - \mathbf{W}_k\mathbf{W}_k^T\mathbf{x}_i \|^2 \tag{35}$$

where β, $0 < \beta \leq 1$, is the forgetting factor. A typical value of β is 0.99. If $\beta = 1$, no forgetting takes place, corresponding to the case in which all the data vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$ are weighted equally in the PAST algorithm. The forgetting factor is useful in tracking the PCA subspace of mildly non-stationary data whose statistical properties change slowly, as it weights the less each data vector $\mathbf{x}_i$ the longer back in time (or index value) it is from the current data vector $\mathbf{x}_k$. The index k denotes both iteration and data vector number. Note that if $k = N$ and $\beta = 1$, the criterion (35) coincides with the error criterion (25).

The key idea in developing the PAST algorithm [36] is to approximate the term $\mathbf{W}_k^T\mathbf{x}_i$ in (35) by $\mathbf{W}_{k-1}^T\mathbf{x}_i$. Because the weight matrix $\mathbf{W}_{k-1}$ from the previous iteration step $k-1$ is known, the quantity $\mathbf{W}_{k-1}^T\mathbf{x}_i$ can be computed and inserted into the criterion (35). It turns then to a standard weighted least-squares criterion which can be optimized recursively using the PAST algorithm below. Furthermore, the approximation error is usually small after the first iterations, because the weight matrices $\mathbf{W}_k$ and $\mathbf{W}_{k-1}$ from subsequent iterations are close to each other.

The k:th iteration of the PAST algorithm for estimating basis vectors of the standard PCA subspace is [36]

$$\begin{aligned}
\mathbf{v}_k &= \mathbf{W}_{k-1}^T\mathbf{x}_k \\
\mathbf{h}_k &= \mathbf{P}_{k-1}\mathbf{v}_k \\
\mathbf{m}_k &= \mathbf{h}_k/[\beta + \mathbf{v}_k^T\mathbf{h}_k] \\
\mathbf{P}_k &= \frac{1}{\beta}\mathrm{Tri}[\mathbf{P}_{k-1} - \mathbf{m}_k\mathbf{h}_k^T] \\
\mathbf{e}_k &= \mathbf{x}_k - \mathbf{W}_{k-1}\mathbf{v}_k \\
\mathbf{W}_k &= \mathbf{W}_{k-1} + \mathbf{e}_k\mathbf{m}_k^T
\end{aligned} \tag{36}$$

The notation Tri means that only the upper triangular part of the argument matrix is computed and copied to the lower triangular part, thus making the matrix $\mathbf{P}_k$ symmetric. The initial values of the matrices $\mathbf{W}_0$ and $\mathbf{P}_0$ can be taken $n \times m$ and $m \times m$ “unit” matrices, respectively.

The PAST algorithm (36) has become quite popular and more or less a standard choice in adaptive estimation of PCA subspace especially in signal processing due to its good properties. It converges faster than the respective steepest descent and especially stochastic gradient type algorithms. The computational load of the PAST algorithm per each iteration is quite tolerable though somewhat higher than for stochastic gradient type algorithms, because the most demanding operations needed in it are just matrix-vector multiplications. Furthermore, the PAST algorithm is truly iterative. One can attribute its good accuracy to the least-squares optimization of the criterion (35), which leads to replacing the scalar learning parameter μ_k in gradient algorithms by the gain matrix $\mathbf{P}_k$. Moreover, the asymptotic convergence of the PAST algorithm has been proved in [37].

The sequential version of the PAST algorithm, called by its developer as PASTd, estimates the principal PCA eigenvectors directly as follows [36]. On the k:th iteration,

$$\mathbf{x}_k(1) = \mathbf{x}_k; \tag{37}$$

For $j = 1, 2, \ldots, m$,

$$\begin{aligned} v_k(j) &= \mathbf{w}_{k-1}^T(j)\mathbf{x}_k(j) \\ d_k(j) &= \beta d_{k-1}(j) + [v_k(j)]^2 \\ \mathbf{e}_k(j) &= \mathbf{x}_k(j) - \mathbf{w}_{k-1}(j)v_k(j) \\ \mathbf{w}_k(j) &= \mathbf{w}_{k-1}(j) + \mathbf{e}_k(j)[y_k(j)/d_k(j)] \\ \mathbf{x}_k(j+1) &= \mathbf{x}_k(j) - \mathbf{w}_k(j)v_k(j) \end{aligned} \tag{38}$$

This is again a deflation type algorithm which estimates the PCA eigenvectors in a sequential manner, and may therefore be somewhat less accurate than the PCA subspace version (36) of the PAST algorithm. However, this is not so great problem as with the deflationary type gradient algorithms, because the algorithm (38) is clearly more accurate due to rougly optimal determination of the learning parameter from the minimization of the least-squares error criterion. Also the convergence of the PASTd algorithm is proved in [37].

2.6 Probabilistic PCA and Gaussianity

Roweis [28] and Tipping and Bishop [30] have shown that PCA emerges as the maximum likelihood solution of a probabilistic latent variable model in which all the stochastic variables involved are Gaussian. In this model, the data vectors are again represented in terms of the equation (21), but now it is assumed that $\mathbf{e}_i$ is a zero mean n-dimensional Gaussian noise vector with covariance matrix $\sigma^2\mathbf{I}$. That is, the components of the noise vector $\mathbf{e}_i$ are mutually uncorrelated and have equal variances σ^2. Also the n-dimensional data vector $\mathbf{x}_i$ and m-dimensional feature vector $\mathbf{y}_i$ are assumed to have Gaussian (marginal) distributions, see [7, 28, 30] for exact assumptions. Thus in the data model (21) $\mathbf{x}_i$ is represented in the subspace spanned by the non-random weight vectors $\mathbf{w}_j$, $j = 1, 2, \ldots, m$.

In [28, 30], an EM algorithm is derived for computing a basis of the PCA subspace of chosen dimensionality m. It is obtained as a limiting case from a slightly more general EM algorithm when the noise variance σ^2 approaches zero, see [7]. Using the matrix data model (22), this EM algorithm can be expressed compactly as

$$\text{E-step:} \quad \mathbf{Y} = [\mathbf{W}^T\mathbf{W}]^{-1}\mathbf{W}^T\mathbf{X} \tag{39}$$

$$\text{M-step:} \quad \mathbf{W}_{new} = \mathbf{X}\mathbf{Y}^T[\mathbf{Y}\mathbf{Y}^T]^{-1} \tag{40}$$

This algorithm converges to a weight matrix (10) whose column vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_m$ are estimates of basis vectors of the m-dimensional PCA subspace of the data $\mathbf{X}$. In general, these estimated basis vectors are not mutually orthonormal, and therefore the inverse matrix $[\mathbf{W}^T\mathbf{W}]^{-1}$ is needed to form the $m \times n$ projection matrix $[\mathbf{W}^T\mathbf{W}]^{-1}\mathbf{W}^T$ onto the estimated PCA subspace. Furthermore, because the EM algorithm (39)–(40) estimates a basis of the PCA subspace only, post-processing is needed for estimating the PCA eigenvectors themselves. However, Ahn and Oh [1] as well as Choi [8] have developed modified EM algorithm for PCA which estimate the first PCA eigenvectors directly.

A prominent feature of the data model (21) used in probabilistic PCA is that all the involved random vectors $\mathbf{x}_i, \mathbf{y}_i$, and $\mathbf{n}_i$ are assumed to have Gaussian distributions. Thus PCA is optimal for Gaussian data, but generally non-optimal for otherwise distributed data. Intuitively, this can be understood from the fact that PCA relies only on the second-order statistics, namely the covariance matrix of the data [19, 20] for zero mean data. This suffices for Gaussian data, because its probability distribution is determined completely by its covariance matrix for zero mean data. However, non-Gaussian data carries a lot of useful information in its higher-order statistics which should also be utilized for getting better results [16, 23].

3 Robust PCA

3.1 Transforming the data vectors more robust

A rather simple and straightforward method of reducing the effect of outliers and impulsive noise in the data is to suppress large elements in the data vectors themselves. After this preprocessing step, one can use the suppressed data vectors quite similarly as the original ones in any methods and algorithms. One could simply cut large values which exceed a suitably chosen threshold value [9], but a more refined method which avoids drawbacks of a strict threshold value is to suppress them. In the following, a simple but useful method for doing this is presented.

First, we estimate the standard deviation of each component of the data vectors $\mathbf{x}$. For zero-mean data vectors

$$\mathbf{x}_j = [x_{1j}, x_{2j}, \dots, x_{nj}]^T, \qquad j = 1, 2, \dots, N \tag{41}$$

the estimated variances $\hat{\sigma}_i^2$, $i = 1, \dots, n$, of their components are the respective diagonal values of the sample covariance matrix (3), and the respective estimated standard deviations are of course their square roots $\hat{\sigma}_i$, $i = 1, \dots, n$. The data vectors $\mathbf{x}_j$, $j = 1, 2, \dots, N$, are now transformed to suppressed data vectors $\mathbf{z}_j$ whose components are

$$z_{ij} = \alpha_i \tanh(x_{ij}/\alpha_i), \qquad i = 1, 2, \dots, n \tag{42}$$

There the scaling constant α_i of each component is

$$\alpha_i = c\hat{\sigma}_i \tag{43}$$

where the constant c is typically chosen from the interval $[2, 3]$.

The scaling function (42) is plotted in Figure 1. There $\alpha_i = 3$, $c = 3$ and $\hat{\sigma}_i = 1$. Note that the curves in Figure 1 behave qualitatively similarly for any $\hat{\sigma}_i$.

After the transformation (42), the covariance matrix $\mathbf{C}_{zz} = \mathrm{E}[\mathbf{z}\mathbf{z}]^T$ of the transformed data vectors $\mathbf{z}_j$ is estimated quite similarly as for the original data vectors $\mathbf{x}_j$ from

$$\hat{\mathbf{C}}_{zz} = \frac{1}{N}\mathbf{Z}\mathbf{Z}^T = \frac{1}{N}\sum_{i=1}^{N} \mathbf{z}_j \mathbf{z}_j^T \tag{44}$$

where

$$\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N] \tag{45}$$

is the entire transformed data matrix having as its columns the transformed data vectors $\mathbf{z}_j$, $j = 1, \dots, N$.

The limiting maximum value of the transformed component z_{ij} in (42) is $\alpha_i = c\hat{\sigma}_i$ when $x_{ij} \to \infty$, and minimum value $-c\hat{\sigma}_i$ when $x_{ij} \to -\infty$. If for example $c = 3$, these values are equal to $\pm 3\hat{\sigma}_i$.

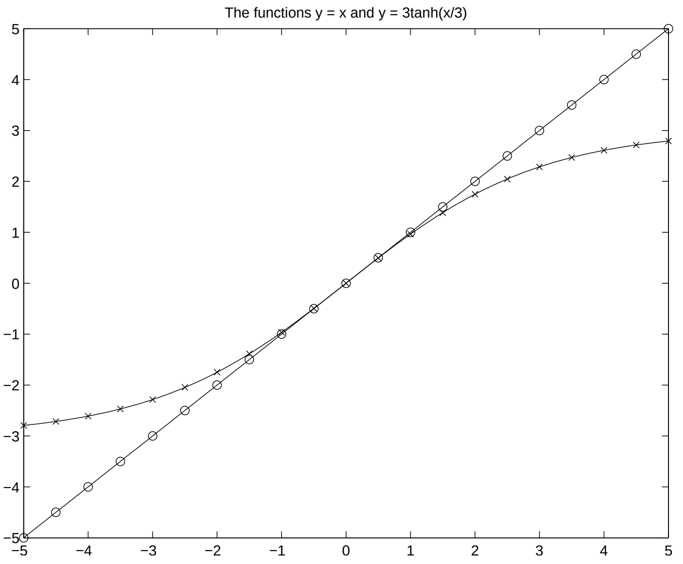

Figure 1: The sigmoidal scaling function $z = 3\tanh(x/3)$ (marked by x) compared with the straight line $z = x$ (marked by o) corresponding to no scaling.

Now recall that if z_{ij} were Gaussian distributed, the probability that it has a value which is farther away than $\pm 3\sigma$ from its mean is only 0.0026. Thus such values are highly probably outliers, and one can in practice suppress also values that are in the interval $[\pm 2\sigma, \pm 3\sigma]$ from the mean. However, if there are many outliers, there could be a local maximum in the distribution of z_i's near the values $z_i = \pm c\hat{\sigma}_i$, resulting into a somewhat U-shaped distribution of z_i. This can be avoided by using in (42) instead of the sigmoidal tanh function a suitably scaled logarithmic function which does not saturate when $x_{ij} \to \pm\infty$. An example of such a function is the inverse hyperbolic sine function

$$\sinh^{-1}(x) = \ln(x + \sqrt{x^2 + 1}) \tag{46}$$

3.2 Robust generalization of variance maximization

A robust generalization of the variance maximization problem (9), (11) can be obtained by replacing the variance in (9) by a more robust cost function. For the i:th weight vector $\mathbf{w}_i$, the generalized criterion is then [19, 20]

$$\mathcal{J}_{genvar}(\mathbf{w}_i) = \mathrm{E}[f(\mathbf{x}^T\mathbf{w}_i)] \tag{47}$$

under the constraints (11) that $\mathbf{w}_i$ is normalized to unity and orthogonal to all the other weight vectors $\mathbf{w}_j$, $j \neq i$, in the symmetric case. The function $f(t)$ in (47) is assumed to be a valid cost function that grows less than quadratically (that is, proportionally to t^2) at least for large t. More specifically, $f(t)$ is even, nonnegative, and continously differentiable almost everywhere. Examples of such cost functions are $f(t) = \mathrm{lncosh}(t)$ and $f(t) = |t|$ suitably scaled.

In the symmetric case, the criterion (47) can be expressed more compactly in matrix-vector form as [19]

$$\mathcal{J}_{genvar}(\mathbf{W}) = \mathbf{1}^T\mathrm{E}[f(\mathbf{W}^T\mathbf{x})] + \frac{1}{2}\mathrm{tr}[\mathbf{\Lambda}(\mathbf{W}^T\mathbf{W} - \mathbf{I})] \tag{48}$$

where the elements of the matrix $\mathbf{\Lambda}$ are Lagrange multipliers λ_{ij}, and $\mathbf{1} = [1, 1, \ldots, 1]^T$ is a 'ones' vector of compatible length, in this case m. When the argument of a scalar function such as $f(\cdot)$

above is a vector or matrix, it means here and later on that this function is applied separately to each element of its argument vector or matrix.

Minimization of the criterion (48) with respect to the weight matrix $\mathbf{W}$ and the Lagrange multipliers $\mathbf{\Lambda}$ leads to the following simple generalization of Oja's PCA subspace rule (14)–(15) [19, 20]:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k \mathbf{e}_k g(\mathbf{x}_k^T \mathbf{W}_k) = \mathbf{W}_k + \mu_k [\mathbf{I} - \mathbf{W}_k \mathbf{W}_k^T] \mathbf{x}_k g(\mathbf{x}_k^T \mathbf{W}_k) \tag{49}$$

where the error vector $\mathbf{e}_k$ at iteration k is defined by (15). The function $g(t)$ is scaled suitably and grows less than linearly. It is the derivative of the function $f(t)$. Typically for example $g(t)$ = tanh(t) or $g(t) = +1$, $t \geq 0$; $g(t) = -1$, $t < 0$. The term $\mathbf{I} - \mathbf{W}_k \mathbf{W}_k^T$ keeps the matrix $\mathbf{W}$ orthonormal, so that the condition (11) is satisfied.

The batch steepest descent version of the stochastic gradient algorithm (49), which uses the entire data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N]$ in each iteration, can be written as

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k \widetilde{\mathbf{E}}_k g(\widetilde{\mathbf{X}}^T \mathbf{W}_k) \tag{50}$$

where the normalized data matrix

$$\widetilde{\mathbf{X}} = \frac{1}{\sqrt{N}} \mathbf{X} \tag{51}$$

and the normalized error matrix

$$\widetilde{\mathbf{E}}_k = \widetilde{\mathbf{X}} - \mathbf{W}_k \mathbf{W}_k^T \widetilde{\mathbf{X}} = \frac{1}{\sqrt{N}} (\mathbf{X} - \mathbf{W}_k \mathbf{W}_k^T \mathbf{X}) \tag{52}$$

The scaling factor $1/N$ of the data covariance matrix (3) is thus divided evenly to the data matrices $\mathbf{X}$ appearing in these two update formulas. One should preprocess the data matrix $\mathbf{X}$ before any iterations by normalizing it using Eq. (51).

The variance maximization problem can be generalized in a similar manner in the sequential (deflationary) case as well [20]. This sequential version of the algorithm (49) is

$$\mathbf{w}_{k+1}(i) = \mathbf{w}_k(i) + \mu_k \mathbf{e}_k(i) g(\mathbf{x}_k^T \mathbf{w}_k(i)), \qquad i = 1, \dots, m \tag{53}$$

$$\mathbf{e}_k(i) = \mathbf{x}_k - \sum_{j=1}^{i} [\mathbf{w}_k^T(j) \mathbf{x}_k] \mathbf{w}_k(j) = \mathbf{e}_k(i-1) - [\mathbf{w}_k^T(i) \mathbf{x}_k] \mathbf{w}_k(i) \tag{54}$$

In these equations, $\mathbf{w}_k(i)$ denotes the estimate of i:th basis vector (PCA eigenvector) at iteration k, and $\mathbf{e}_k(i)$ is the respective error vector. Actually, this is nothing but a generalization of Sanger's generalized Hebbian algorithm (GHA) [29] for robust variance maximization.

The batch steepest descent version of the algorithm (53)-(54) is obviously for $i = 1, \dots, m$

$$\widetilde{\mathbf{E}}_k(i) = \widetilde{\mathbf{X}} - \sum_{j=1}^{i} \mathbf{w}_k(j) \mathbf{w}_k^T(j) \widetilde{\mathbf{X}} = \widetilde{\mathbf{E}}_k(i-1) - \mathbf{w}_k(i) \mathbf{w}_k^T(i) \widetilde{\mathbf{X}}, \tag{55}$$

$$\mathbf{w}_{k+1}(i) = \mathbf{w}_k(i) + \mu_k \widetilde{\mathbf{E}}_k(i) g(\widetilde{\mathbf{X}}^T \mathbf{w}_k(i)) \tag{56}$$

In the linear case $g(t) = t$ corresponding to the maximization of standard variance in (53) and (56), the algorithm (53)-(54) becomes Sanger's GHA algorithm for standard linear PCA, and the algorithm (55)-(56) is simply the batch steepest descent version of the GHA algorithm.

A topic which would deserve attention is which are the optima of the criteria (47) and (48), but this has not yet been studied. Obviously these optima are different from PCA solutions, because the derived algorithms perform in practice worse than standard PCA in impulsive noise.

3.3 Robust generalization of minimization of MSE approximation error

The mean-square approximation error in (16) can also be taken as a starting point for deriving robust algorithms [19, 20, 34]. The robust criterion corresponding to the standard MSE error (16) is in the symmetric case [19, 20]

$$\mathcal{J}_{GMSE} = \mathbf{1}^T \mathrm{E}[f(\mathbf{e})] = \mathbf{1}^T \mathrm{E}[f(\mathbf{x} - \widehat{\mathbf{x}})] = \mathbf{1}^T \mathrm{E}[f(\mathbf{x} - \mathbf{W}\mathbf{W}^T\mathbf{x})] \tag{57}$$

where the robust cost function $f(t)$ satisfies similar properties as before. Thus in (57) the mean-square approximation error has been replaced by a robust approximation error.

Minimization of the robust approximation error criterion (57) leads to the following algorithm:

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k[\mathbf{x}_k g(\mathbf{e}_k^T)\mathbf{W}_k + g(\mathbf{e}_k)\mathbf{x}_k^T\mathbf{W}_k] \tag{58}$$

where again $g(t)$ is the derivative of $f(t)$, which grows less than linearly.

Comparing the robust algorithm (58) with its standard PCA subspace estimation counterpart (20), we see that the only difference is that the error vector $\mathbf{e}_k$ defined already in (15) is replaced by robust error vector $g(\mathbf{e}_k)$. In early experiments [20] it was found that the algorithm (58) is prone to local minima. Similarly as (20), the algorithm (58) can be approximated by dropping the first term $\mathbf{x}_k g(\mathbf{e}_k^T)\mathbf{W}_k$ from its update inside the square brackets out. The resulting algorithm

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k g(\mathbf{e}_k)\mathbf{x}_k^T\mathbf{W}_k \tag{59}$$

$$\mathbf{e}_k = \mathbf{x}_k - \mathbf{W}_k\mathbf{W}_k^T\mathbf{x}_k \tag{60}$$

where the error vector $\mathbf{e}_k$ in (60) is reproduced for convenience, is less prone to local minima and converges at least roughly to the same solution as the exact algorithm (58).

One should note here an important difference: in the approximative algorithm (59) for minimizing the robust approximation error (57), the nonlinearity $g(t)$ is applied to the error vector $\mathbf{e}_k$, while in the algorithm (49) derived by generalizing the variance maximization criterion it is applied to the transposed linear response vector $\mathbf{x}_k^T\mathbf{W}_k$. Thus these robust algorithms are different and yield different results, while in estimating the standard PCA subspace they coincide because then $g(t) = t$.

The batch steepest descent version of the algorithm (59)-(60) is obviously

$$\mathbf{W}_{k+1} = \mathbf{W}_k + \mu_k g(\widetilde{\mathbf{E}}_k)\widehat{\mathbf{X}}^T\mathbf{W}_k, \tag{61}$$

where the error matrix $\widetilde{\mathbf{E}}_k$ for the entire data set $\mathbf{X}$ is computed on each iteration k from the formula (52).

The sequential version of the algorithm (59)-(60) is

$$\mathbf{w}_{k+1}(i) = \mathbf{w}_k(i) + \mu_k g(\mathbf{e}_k(i))\mathbf{x}_k^T\mathbf{w}_k(i), \qquad i = 1, \dots, m \tag{62}$$

where the error vector $\mathbf{e}_k(i)$ corresponding to the estimate of i:th basis vector $\mathbf{w}_k(i)$ (robust PCA eigenvector) at iteration k is computed from (54). The algorithm (62) is a generalization of Sanger's generalized Hebbian algorithm (GHA) [29] for minimizing robust error [20].

The batch steepest descent version of the algorithm (62), (54) is obviously for $i = 1, \dots, m$

$$\mathbf{w}_{k+1}(i) = \mathbf{w}_k(i) + \mu_k g(\widetilde{\mathbf{E}}_k(i))\widehat{\mathbf{X}}^T\mathbf{w}_k(i), \tag{63}$$

where the error matrix $\widetilde{\mathbf{E}}_k(i)$ for the i:th basis vector $\mathbf{w}_k(i)$ at iteration k is computed from (55).

3.4 Robust Newton type algorithms

Robust versions of the algorithms discussed in subsection 2.4 can be developed by replacing the squared norm $\| \cdot \|^2$ in (25) by a suitable cost function $f(t)$ which grows less than quadratically, for example

$$f(t) = \frac{1}{\alpha}\mathrm{lncosh}(\alpha t) \tag{64}$$

where α is a suitably chosen scaling parameter. The derivative of this function is

$$g(t) = \frac{df(t)}{dt} = \tanh(\alpha t) \tag{65}$$

and its derivative is in turn

$$g'(t) = \frac{dg(t)}{dt} = \alpha[1 + g(t)][1 - g(t)] \tag{66}$$

The robust versions of the steepest descent Oja's PCA subspace rule and the modified algorithm (30)-(31) are now obtained simply by replacing the error matrix $\mathbf{E}_k$ by $g(\mathbf{E}_k)$. Scaling the data matrix yields then the steepest descent version (61) of robust Oja's PCA subspace rule for generalized error minimization, and similarly for the modified algorithm (30)-(31). For clarity, we give the update formulas for this latter algorithm here. They are

$$\begin{aligned} \mathbf{W}_{k+1} &= \mathbf{W}_k + \mu_k g(\widetilde{\mathbf{E}}_k)\widetilde{\mathbf{Y}}_k^T & (67) \\ \widetilde{\mathbf{Y}}_{k+1} &= \widetilde{\mathbf{Y}}_k + \mu_k \mathbf{W}_k^T g(\widetilde{\mathbf{E}}_k) & (68) \end{aligned}$$

where the normalized error matrix is

$$\widetilde{\mathbf{E}}_k = \widetilde{\mathbf{X}} - \mathbf{W}_k\widetilde{\mathbf{Y}} \tag{69}$$

and $\widetilde{\mathbf{X}} = \frac{1}{\sqrt{N}}\mathbf{X}$. In robust Oja's PCA subspace rule, only the first update equation (67) is used, and $\widetilde{\mathbf{Y}}_k$ is computed instead of (68) from $\widetilde{\mathbf{Y}}_k = \mathbf{W}_k^T\widetilde{\mathbf{X}}$. Of course, one can use the scaled quantities $\widetilde{\mathbf{X}}$, $\widetilde{\mathbf{E}}_k$, and $\widetilde{\mathbf{Y}}$ in the standard PCA algorithm (30)-(31), too, which can be then obtained by dropping the nonlinearity $g(\cdot)$ off from the equations (67)-(68).

It is straightforward to see that when the quadratic error in (25) is replaced by a more general robust error criterion $f(x_{ij} - \hat{x}_{ij})$, the second partial derivatives in (33) are replaced by

$$\frac{\partial^2 \mathcal{C}}{\partial w_{il}^2} = 2\sum_{j=1}^{N} g'(x_{ij} - \hat{x}_{ij})y_{lj}^2, \qquad \frac{\partial^2 \mathcal{C}}{\partial y_{lj}^2} = 2\sum_{i=1}^{n} g'(x_{ij} - \hat{x}_{ij})w_{il}^2 \tag{70}$$

and the Newton type counterpart of the robust PCA subspace algorithm (67)-(68) takes the form

$$\begin{aligned} \Delta w_{il} &= \left(\frac{\partial^2 \mathcal{C}}{\partial w_{il}^2}\right)^{-1} \sum_{j=1}^{N} g(x_{ij} - \hat{x}_{ij})y_{lj} & (71) \\ \Delta y_{lj} &= \left(\frac{\partial^2 \mathcal{C}}{\partial y_{lj}^2}\right)^{-1} \sum_{j=1}^{N} g(x_{ij} - \hat{x}_{ij})w_{il} & (72) \end{aligned}$$

The respective Newton type robust Oja's PCA subspace algorithm is obtained by using only the first update formula (71), and computing values of y_{lj} from the relationship $\mathbf{Y} = \mathbf{W}^T\mathbf{X}$. In practice, we use the normalized data matrix $\widetilde{\mathbf{X}}$ from (51), so that also $\mathbf{Y}$ is replaced by $\widetilde{\mathbf{Y}}$ and $\widehat{\mathbf{X}}$ is computed from the formula $\widehat{\mathbf{X}} = \mathbf{W}\widetilde{\mathbf{Y}}$.

However, this algorithm did not work well in preliminary experiments, because it zigzags between iterations and becomes rather easily unstable. Therefore, we used approximative robust Newton algorithms where the second partial derivatives in (71) and (72) are computed from (33) instead of (70). These approximative robust Newton algorithms perform much better, maybe because the values of the second derivatives in (33) do not depend on the index i and j, respectively.

3.5 Robust least-squares type algorithms

Similarly as for the mean-square error (16), we can generalize the weighted least-squares error criterion (35) by replacing the squared error $\| \mathbf{x}_i - \mathbf{W}_k\mathbf{W}_k^T\mathbf{x}_i \|^2$ in (35) with its robust counterpart. This leads to the criterion

$$J_{GWLS}(k) = \sum_{i=1}^{k} \beta^{k-i} f(\mathbf{x}_i - \mathbf{W}_k\mathbf{W}_k^T\mathbf{x}_i) \tag{73}$$

where the function $f(t)$ satisfies the same properties as in (47). That is, $f(t)$ is even, nonnegative, and continously differentiable almost everywhere, and grows less quadratically at least for large t. Our selection for such a cost function is $f(t) = \text{lncosh}(t)$ scaled suitably.

A robust version of the subspace PAST algorithm is now obtained simply by replacing the error vector $\mathbf{e}_k$ in (36) by the nonlinear error

$$g(\mathbf{e}_k) = g(\mathbf{x}_k - \mathbf{W}_{k-1}\mathbf{v}_k) = g(\mathbf{x}_k - \mathbf{W}_{k-1}\mathbf{W}_{k-1}^T\mathbf{x}_k) \tag{74}$$

where the nonlinearity $g(t)$ grows less than linearly with $| t |$. If $f(t) = \text{lncosh}(t)$, $g(t) = \tanh(t)$. Otherwise, the robust version of the subspace PAST algorithm remains the same as the original least-squares algorithm (36). This robust PAST algorithm performs usually clearly better than standard PCA when impulsive noise is present in the data.

This algorithm can be justified as follows. Due to the properties of the cost function $f(t)$, $f(\mathbf{e})$ is minimized when the norm $\| \mathbf{e} \|$ (or the squared norm $\| \mathbf{e} \|^2$) of the linear error $\mathbf{e}$ is minimized. Thus we can minimize the robust cost function (73) with respect to this linear error using the standard PAST algorithm (36), and take into account the nonlinearity $f(t)$ by replacing the error vector $\mathbf{e}_k$ in (36) by its robust counterpart $g(\mathbf{e}_k)$ in (74), where $g(t)$ is the derivative of $f(t)$. While this derivation of the robust PAST algorithm is somewhat heuristic, the algorithm is well justified by its good performance in practice.

A robust version of the sequential PAST algorithm is obtained quite similarly by replacing the linear error vector $\mathbf{e}_k(j)$ in (38) by its nonlinear (robust) counterpart $g(\mathbf{e}_k(j))$. The other steps of this algorithm remain the same, too.

We have earlier developed a nonlinear PCA version of the PAST algorithm for independent component analysis and blind source separation, which also converges much faster than respective stochastic gradient type algorithms, see [16, 24].

4 Missing values in PCA

General remarks. In real-world data sets there are often missing values. This problem and methods for handling it are considered on a general level in [2, 21]. Various methods for treating missing values in PCA are discussed in Section 13.6 in [18]. If there are only a few missing values, the usual method is to omit completely from the computations the data vectors for which at least one of their components is missing. But if the data contains a significant amount of missing values, this simple method wastes too much information. Another simple customary method is to estimate each element of the sample covariance matrix only over such component pairs of the data vectors for which both the components have values. However, this method can yield a sample covariance matrix that is not valid because it is no longer positive (semi)definite [17].

Although the PCA problem in the presence of missing values may at first sight seem as easy as standard PCA, there are some important distinctions [17]. Because of them the situation actually resembles in many ways nonlinear modeling problems. First, there is no analytical solution available for PCA with missing values because even the estimation of the data covariance matrix is nontrivial. Second, the optimized cost function typically has local minima, which makes finding of the global optimum more difficult. Regularization is often required in probabilistic approaches because standard PCA approaches can easily lead to overfitting. And algorithms may have a heavy computational load especially in large-scale problems. There are still a few more problems that are mentioned in [17].

All the methods studied in this paper assume that missing values appear at random in each component of a data vector. Similarly, outliers appear at random in each component of a data vector. Contrary to this, in some probabilistic methods at least [4] entire data vectors are assumed to be outliers. One should use a correct model for the presence of missing values and outliers in the data set analyzed, because this is critical for the performance of a method; see for example our experimental results in [22].

Imputation method. A typical method for handling missing values is so-called imputation method [2, 21, 18]. In its simplest form, this means in PCA replacing the missing values in the data by the mean value of that component. Because we use data preprocessed to have zero mean, in our case the missing values are replaced by zeros. After this, one can estimate the sample covariance matrix and compute its PCA in the usual manner.

A more advanced imputation method uses suitably designed iterations for replacing the missing values. Let the current estimate of the data covariance matrix be $\widehat{\mathbf{C}}_{xx}$ in (3), where the missing values in the data vectors have been substituted by their values obtained using the imputation algorithm. Then new estimates of the missing values in the data vectors are computed from (8). Only the missing values are replaced by the values estimated from this formula, while the existing non-missing components of the data vectors are retained to have their original values. The data vectors completed in this way are then used to re-estimate the sample covariance matrix in (3), and this iteration is continued until it converges. This method was proposed already in [3], but using only the first (principal) PCA eigenvector.

Nearest neighbor method. One method for handling missing values in real-world data sets is to look for such a data vector that is as similar as possible to the data vector to be handled which has missing value(s), and then replace the missing values with the respective components of that most similar another data vector. A natural method is to seek for such a data vector by computing the Euclidean distances between the data vector to be handled and the other data vectors in the data set, and then select the nearest neighbor that has minimum distance. This simple idea was developed by the current author, but similar techniques have been used in collaborative filtering [6]. Another possibility is to compute the angles between the data vectors, and replace the missing component from the respective one of the vector that has the smallest angle to the data vector to be handled.

Probabilistic methods. We have developed probabilistic methods for missing values in PCA, and applied them successfully to the huge Netflix data set in [26, 27]. This work is summarized in the long journal paper [17]. There one can find also a general discussion of the problems arising with missing values in PCA, as well as more references. Treatment of missing values is in probabilistic and Bayesian methods usually straightforward, but these methods are often both mathematically demanding and have a heavy computational load. Furthermore, they require tailoring of the developed method to an appropriate data model with assumptions on the probability distributions of the random variables involved.

5 Experimental results

We do not show results for the robust steepest descent algorithms (50) and (55)-(56) obtained by generalizing the variance maximization problem in a symmetric and hierarchic manner, respectively, because they performed worse than standard PCA in impulsive noise. The reason is obviously that the optimum of the criterion (47) or more completely (48) which these algorithms try to achieve is different from PCA.

We also tried robust version of the EM algorithm (39)-(40), where the E-step (39) is replaced by the respective robust step $\mathbf{Y} = [\mathbf{W}^T\mathbf{W}]^{-1}g(\mathbf{W}^T\mathbf{X})$. The resulting algorithm requires orthonormalization of the weight matrix $\mathbf{W}$ after each iteration (40) to perform appropriately. But it suffers from the same problem as robust generalizations of the variance maximization problem, namely that the robust EM algorithm performs worse than standard PCA in impulsive noise. Obviously the basic reason for this is also the same, that is, the optimum of this algorithm is something else than the PCA subspace. The same remarks apply to similar robust generalizations of the EM type PCA algorithm introduced by Ahn and Oh [1] as well as the wake-sleep PCA algorithm by Choi [8]. For the robust generalization of Ahn's and Oh's algorithm it suffices to normalize the weight vectors (column vectors of the weight matrix $\mathbf{W}$), and for the robust wake-sleep PCA algorithm not even this normalization is needed.

We do not show any experimental results for the robust Newton type algorithm (71)-(72) either, because it seems to be too complicated, zigzagging between the subsequent iterations and becoming unstable rather easily. For the remaining algorithms, we present in the following experimental results for simulated and real-world data vectors containing impulsive noise, and after this when also missing values are present in addition to the impulsive noise.

5.1 Simulated data

Experiment 1: Estimation of PCA subspace in impulsive noise

In this experiment, the data vectors $\mathbf{x}_i$ are 5-dimensional. They are first generated so that their components are zero-mean statistically independent Gaussian distributed random numbers with variances $\sigma_1^2 = 5.0$, $\sigma_2^2 = 3.0$, $\sigma_3^2 = 2.0$, $\sigma_4^2 = 1.0$, and $\sigma_5^2 = 0.6$. The covariance matrix of the data is thus a diagonal matrix with the above variances, and its eigenvectors are simply unit vectors. Hence the two first PCA eigenvectors are $\mathbf{u}_1 = [1, 0, 0, 0, 0]$ and $\mathbf{u}_2 = [0, 1, 0, 0, 0]$, and the projection of the normalized weight vectors $\mathbf{w}_j$ onto the two-dimensional PCA subspace consist simply of the two first components of the weight vectors $\mathbf{w}_j$. However, the algorithms tested do of course not utilize and know this information, but use the generated data vectors only, and treat them as data vectors having any possible distribution. In this experiment, we tried to estimate this two-dimensional PCA subspace using different methods.

Furthermore, a certain amount of impulsive noise is added to the data. The impulsive noise is modeled and generated by adding with a certain fixed probability p_v uniformly distributed random numbers lying in the interval $[-a, +a]$ independently to each component of the data vectors, which increases theoretically the variance of all the components of the data vectors by the same amount $p_v a^2/3$. In this experiment, $p_v = 0.05$ and $a = 10$, so that the amount of 1.67 is added to all the variances above. This makes reliable estimation of the eigenvectors clearly more difficult than in the case where there is no impulsive noise, because now the the eigenvalues are relatively closer to each other. The number of data vectors is $n = 400$. For getting statistically more reliable results, 100 different randomly generated realizations of the data set $\mathbf{X}$ having these statistical properties were used in this experiment. The results are averages for these 100 data sets.

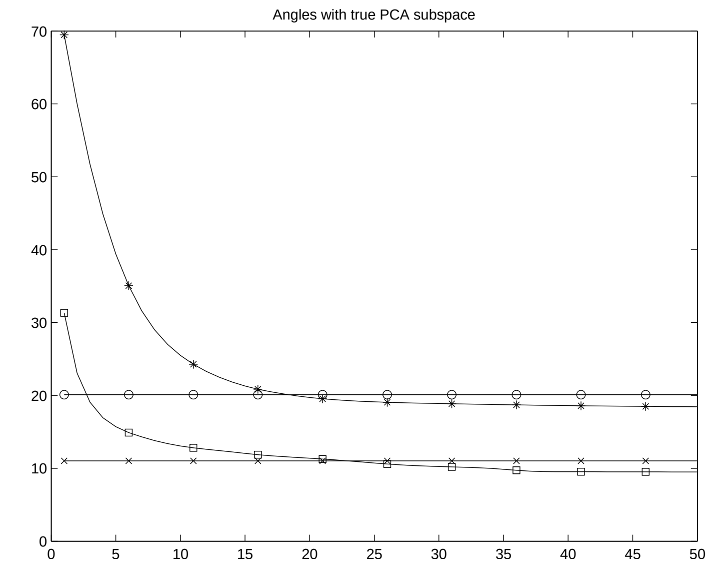

Figure 2: Angles between the true PCA subspace and estimated ones in Experiment 1 for different methods versus the number of sweeps. Standard PCA (line marked with circles), PCA from robust data vectors (line marked with crosses), approximative robust Oja's PCA subspace rule (curve marked with asterisks), and robust subspace PAST algorithm (curve marked with squares).

Figure 2 shows the angles between the true PCA subspace and its estimates provided by different methods as a function of sweeps. One sweep means using the data $\mathbf{X}$ once. The number of sweeps was limited to 50, by which also the iterative methods had been converged. We compare only the PCA subspace, because the symmetric iterative algorithms in this experiment estimate some orthonormal basis of the PCA subspace only, and therefore comparing their basis vector estimates to the true PCA eigenvectors is not meaningful.

The scaling coefficient in the robust method where large components of the data vectors were first suppressed was $c = 2.5$ in (43). It seems that this choice is not so critical, because the results are almost the same for example if $c = 2.0$ or $c = 3.0$. Figure 2 shows that this simple method improves clearly the accuracy of the estimated PCA subspace, and it also improves the accuracy of the estimated PCA eigenvectors. For the iterative methods, the scaling coefficient in (65) was $\alpha = 0.8$. Also this choice is not so critical. The learning parameter in the approximative robust Oja's PCA subspace rule was computed during each sweep k from the formula

$$\mu_k = \frac{\mu_0}{1 + k/\tau} \tag{75}$$

where the search time constant τ guarantees that the learning parameter μ_k does not approach zero too quickly, allowing for the algorithm sufficient time to converge close to the optimum [13]. In this experiment, we chose $\mu_0 = 1$ and $\tau = 20$. Finally, the forgetting factor in the robust subspace PAST algorithm was $\beta = 0.99$.

From Figure 2 one can see that robust subspace PAST algorithm (marked with squares) performs the best, but PCA from robust data vectors (marked with crosses) is close to it. Approximative robust Oja's PCA subspace rule (marked with asterisks) performs only slightly better than standard PCA estimated by computing the eigenvectors of the sample covariance matrix (marked with circles), which gives the least accurate estimated PCA subspace. In more difficult cases where the proportion of data vectors corrupted by impulsive noise is larger and there are less data vectors the results are qualitatively fairly similar, but the robust subspace PAST algorithm is most accurate with a larger difference in performance compared with PCA from robust data vectors, and standard PCA may perform slightly better than approximative robust Newton type Oja's PCA subspace rule.

Experiment 2: Estimation of PCA eigenvectors in impulsive noise

In another experiment, the data vectors were first generated quite similarly as in Experiment 1. That is, they were 5-dimensional zero-mean Gaussian random vectors with the same variances as in Experiment 1. The data vectors were corrupted by impulsive noise generated from the same uniform distribution as in Experiment 1. But now the estimation task was considerably more difficult as in the first experiment as the probability of impulsive noise was three times higher, $p_v = 0.15$, and there were 200 data vectors only. The scaling coefficient of the hyperbolic tanh function in (65) was now $\alpha = 1.0$, $\mu_0 = 0.5$, and the total number of sweeps was 60. The other parameters were the same as in Experiment 1.

This time we compared sequential (deflationary) type methods that estimate the PCA eigenvectors directly, not only the PCA subspace spanned by them. The methods compared were standard PCA estimated by computing the eigenvectors of the sample covariance matrix (lines marked with circles in Figures 3 - 5), the same method using robust data vectors whose large components were suppressed (lines with crosses), robust steepest descent GHA algorithm (curves marked with asterisks), and robust sequential PAST algorithm (curves marked with squares). From Figures 3 - 5 one can see that robust sequential PAST algorithm performs the best, then PCA from robust data vectors, followed by robust steepest descent GHA algorithm, and standard PCA estimation method is again the least accurate. The PCA subspace estimates follow the same order, even though the difference between the accuracy of the robust steepest descent GHA algorithm and standard PCA estimation is now smaller, obviously because the PCA eigenvectors estimated using the standard method are somewhat mixed, containing prominent components of the other estimated eigenvector.

The results provided by these sequential methods were qualitatively similar in other experiments that we carried out with different variances of the 5 component data vectors, different amounts of impulsive noise, and different numbers of data vectors.

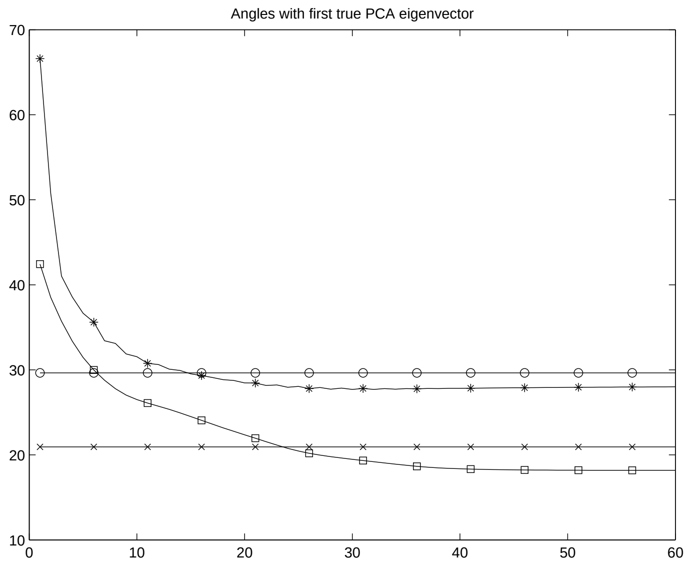

Figure 3: Angles between the first true PCA eigenvector and its estimates in Experiment 2 for different methods versus the number of sweeps: standard PCA (circles), PCA from robust data vectors (crosses), robust steepest descent GHA algorithm (asterisks), and robust sequential PAST algorithm (squares).

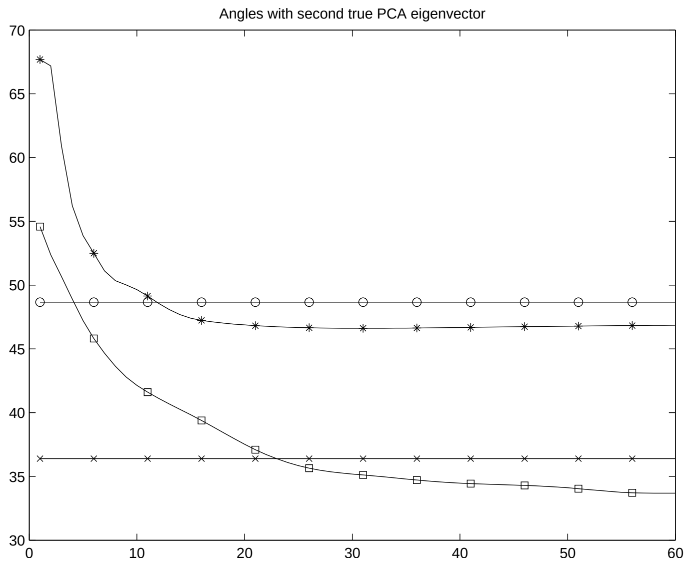

Figure 4: Angles between the second true PCA eigenvector and its estimates in Experiment 2 for different methods versus the number of sweeps: standard PCA (circles), PCA from robust data vectors (crosses), robust steepest descent GHA algorithm (asterisks), and robust sequential PAST algorithm (squares).

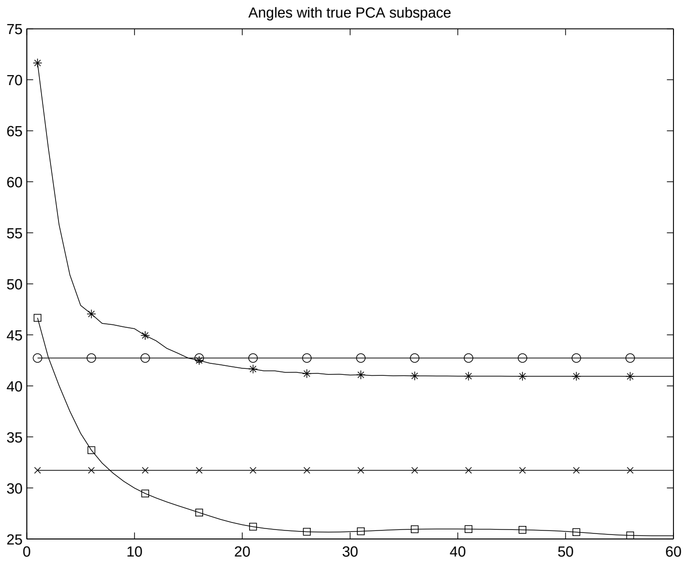

Figure 5: Angles between the true PCA subspace and estimated ones in Experiment 2 for different methods versus the number of sweeps: standard PCA (circles), PCA from robust data vectors (crosses), robust steepest descent GHA algorithm (asterisks), and robust sequential PAST algorithm (squares).

5.2 Real-world forest fires data

For testing our robust PCA methods with real-world data, we chose the data set 'Forest Fires' which is publicly available in the well-known UCI Machine Learning Repository [32]. This data set was originally introduced and used for a difficult regression task in [10]. The aim there is to predict the burned area of forest fires in northeastern Portugal by using meteorological and other data. Our aim is to estimate the principal component subspace or the first PCA eigenvectors of this data set after a certain amount of impulsive noise and outliers were added to the data.

The Forest Fires data set contains 517 data vectors $\mathbf{x}_i, i = 1, \dots, 517$, which are 13-dimensional. Most of their components are real numbers, and the two first components which are integers can be treated as real numbers, too. The third and fourth component are the labels of the month of the year and the day of the week, respectively. They were transformed to numbers so that the month label “jan” was replaced by 1, and so on so that the month label “dec” was replaced by the number 12. The labels of the days of the week were transformed similarly so that “mon” corresponds to the numerical value 1 and “sun” to 7. Furthermore, the last 13th component $x_{i,13}$ (burned forest area) varies very much in different data vectors, ranging from 0.0 hectars to 1091 hectars, and is very skewed towards 0.0. Therefore it was transformed using a $\ln(x_{i,13} + 1)$ function according to the authors' recommendation in [10].

A problem with this and almost all real-world data sets is that no one knows which are the true principal components and PCA eigenvectors. The best that we can have is to first estimate the sample covariance matrix (3) from from the Forest Fires data set without impulsive noise, and then use its eigenvectors as 'true' values after impulsive noise etc. have been added to the data set. Our experimental results show that this may favor batch PCA compared with robust batch PCA and with the iterative robust PCA algorithms that we have proposed when impulsive noise is present. At least the results are qualitatively different from those obtained when using artificially generated data for which the theoretically correct results are known.

Applying batch PCA directly to the Forest Fires data set reveals that the PCA eigenvalues vary very much in magnitude, most of them being small or very small. And even the dominant eigenvalues have highly different values. On the other hand, scaling the variances of all the 13 components in the Forest Fires data set to unity provides PCA eigenvalues which are quite close to each other and unity. In such a case reliable estimatation of the PCA eigenvectors becomes impossible when impulsive noise or outliers are added to the data. For getting a more suitable range of PCA eigenvalues we therefore scaled the Forest Fires data set somewhat heuristically so that the 5th, 6th, and 10th component were divided by 10 and the 7th component by 50, as well as the 12th component was multiplied by 10 and the 13th component by 5 after the logarithmic transformation described above. After this, the 13 eigenvalues of the sample covariance matrix $\widehat{\mathbf{C}}_{xx}$ in (3) estimated using standard batch PCA were 76.95, 48.37, 23.01, 16.06, 11.06, 8.75, 5.73, 4.27, 2.84, 1.38, 1.00, 0.72, and 0.18.

Experiment 3: Estimation of PCA subspace in impulsive noise

Figure 6 shows the results provided by different methods when the estimated PCA subspace was 4-dimensional. That is, we estimated its four basis vectors which should ideally lie in the same subspace as the four first principal eigenvectors of the covariance matrix of the data. The probability of the impulsive noise was $p_v = 0.10$, and its average magnitude was the same for all components of the data vectors. The impulsive noise was the generated from the formula $n_j = 40u$, where u is an uniformly distributed random number lying in the interval $[-0.5, +0.5]$. The scaling coefficients in the robust batch PCA and iterative methods were the same $c = a = 1.5$ for a fair comparison, and the forgetting factor in the robust subspace PAST algorithm was $\beta = 0.999$.

The results are somewhat different compared with simulated data. This time batch PCA estimated from robust data vectors (line marked with crosses) yields the worst result, with standard batch PCA (circles) performing somewhat better. Robust subspace PAST algorithm (marked with squares) is the best, but also approximative robust Newton type Oja's PCA subspace rule (asterisks) provides somewhat better estimate of the PCA subspace than the two compared batch PCA methods.

Experiment 4: Estimation of PCA eigenvectors in impulsive noise

In this experiment, we tried to estimate 4 first PCA eigenvectors in impulsive noise for the forest fires data. The experimental setting was similar to the Experiment 3 but there were some differences in the parameter values. The probability of impulsive noise was $p_v = 0.15$, and it was generated from the formula $n_j = 5\sigma_j u$. That is, the magnitude of the impulsive noise for each component x_j of the data vectors was proportional to the standard deviation σ_j of that component. We could as well have generated impulsive noise in the same manner as in the previous experiment. The scaling coefficients in the robust batch PCA and iterative methods were again the same $c = a = 1.5$ for a fair comparison. The parameters in the search-and-converge strategy (75) were $\mu_0 = 0.03$ and $\tau = 30$. This was used both in the steepest descent GHA algorithm (63) and in the robust PASTd algorithm, where in (38) the new weight vector estimate is computed from the formula

$$\mathbf{w}_k(j) = \mathbf{w}_{k-1}(j) + \mu_k g(\mathbf{e}_k(j))[y_k(j)/d_k(j)] \tag{76}$$

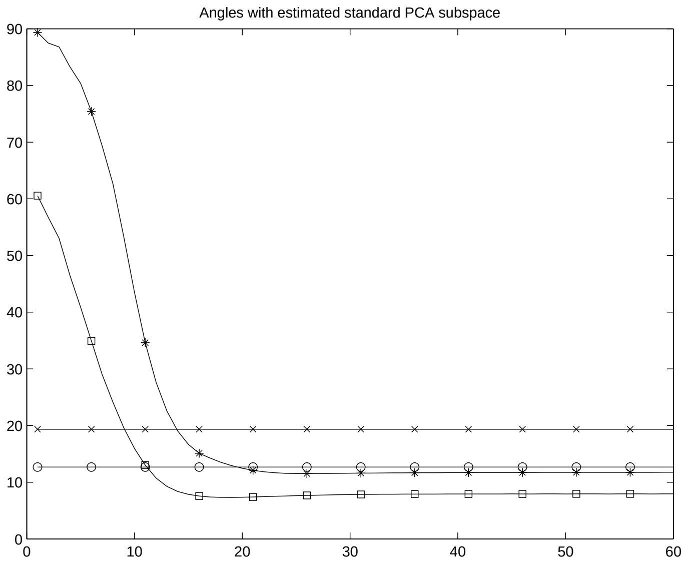

Figure 6: Angles between the “true” 4-dimensional PCA subspace and its estimates for the Forest Fires data set using different methods versus the number of sweeps. Impulsive noise was added to the data set before estimation. Line marked with circles: batch PCA. Crosses: batch PCA from robust data vectors. Asterisks: approximative robust Newton type Oja’s PCA subspace rule. Squares: robust subspace PAST algorithm.

and μ_k is computed from the formula

$$\mu_k = \frac{1}{1 + k/\tau} \tag{77}$$

This changes the robust PASTd algorithm only slightly, so that the updates will be somewhat smaller during the later sweeps. The forgetting factor in the robust PASTd algorithm was $\beta = 0.999$.

Figures 7-11 show the results. The accuracy of the estimated PCA subspace in Figure 7 is best for the steepest descent GHA algorithm (curve marked with asterisks) and for the robust PASTd algorithm (curve marked with squares), and worst for the batch PCA estimated from robust data vectors (line marked with crosses). It is still unclear why the intermediate results for the robust iterative algorithm around sweeps 20 to 30 are better than their final ones. Figures 8-11 show that the accuracies of the estimates of the first to fourth PCA eigenvectors vary somewhat for different methods. They are rather similar for the batch PCA, steepest descent GHA and robust PASTd methods, even though the second and third eigenvector estimates converge somewhat slowly for the robust PASTd method (curves with squares). On the other hand, batch PCA from robust data vectors (lines marked with crosses) provides clearly different results. Its estimates for the first, second and third PCA eigenvectors are most accurate, but for the fourth PCA eigenvector and the PCA subspace it yields clearly the worst estimates. These results vary somewhat with the parameters used, and occasionally batch PCA from robust data vectors may perform better than plain batch PCA.

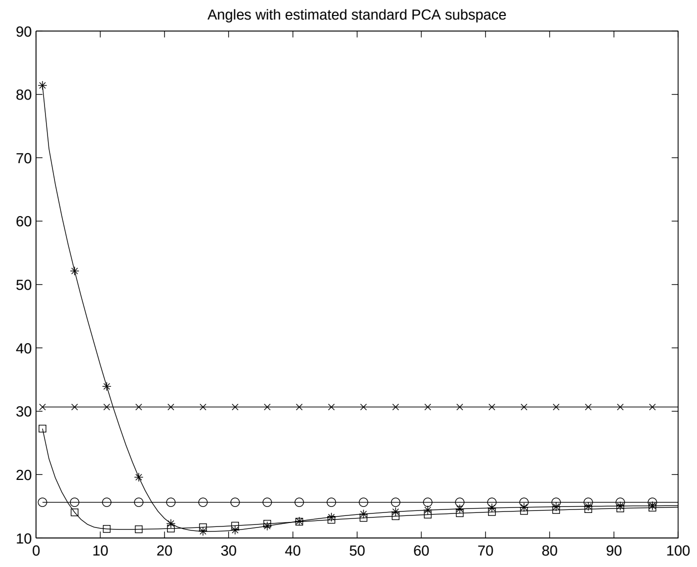

Figure 7: Angles between the “true” PCA subspace and estimated ones in an experiment with the Forest Fires data set for different methods versus the number of sweeps. The estimated PCA subspace was 4-dimensional. Impulsive noise was added to the data set before estimation. Line marked with circles: batch PCA. Crosses: batch PCA from robust data vectors. Asterisks: approximative robust Newton type Oja’s PCA subspace rule. Squares: robust subspace PAST algorithm.

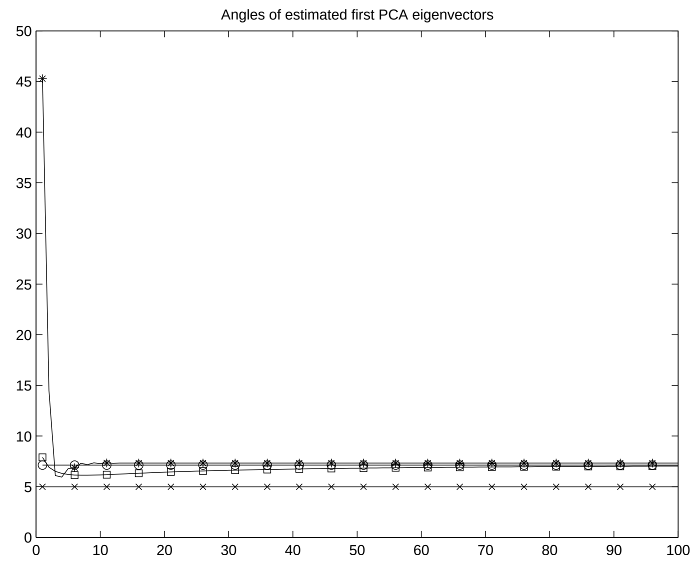

Figure 8: Angles between the first “true” PCA eigenvector and its estimates in an experiment with the Forest Fires data set in impulsive noise for different methods versus the number of sweeps. Line marked with circles: batch PCA. Crosses: batch PCA from robust data vectors. Asterisks: approximative robust Newton type Oja’s PCA subspace rule. Squares: robust subspace PAST algorithm.

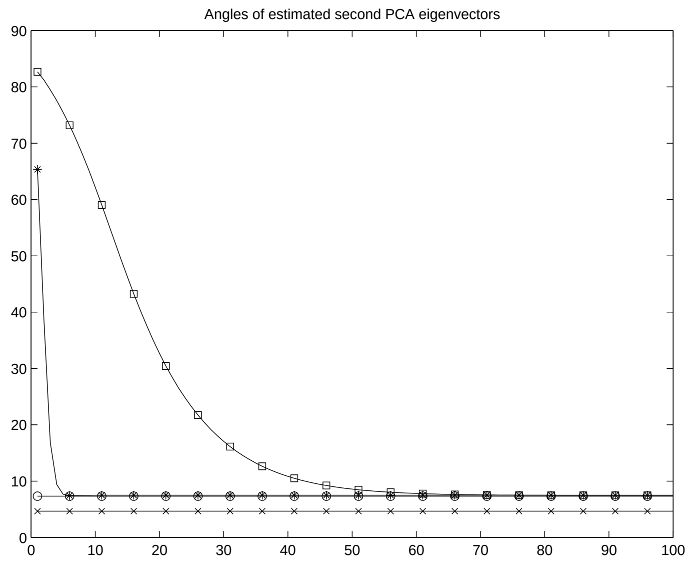

Figure 9: Angles between the second "true" PCA eigenvector and its estimates in an experiment with the Forest Fires data set in impulsive noise for different methods versus the number of sweeps. Line marked with circles: batch PCA. Crosses: batch PCA from robust data vectors. Asterisks: approximative robust Newton type Oja's PCA subspace rule. Squares: robust subspace PAST algorithm.

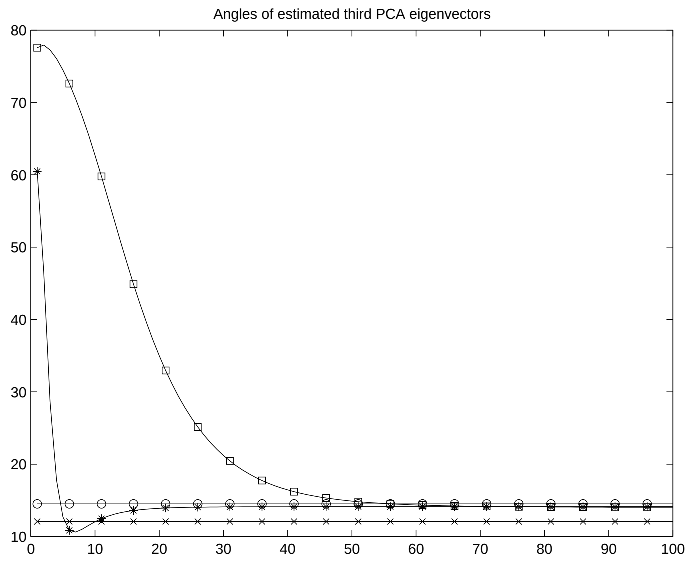

Figure 10: Angles between the third "true" PCA eigenvector and its estimates in an experiment with the Forest Fires data set in impulsive noise for different methods versus the number of sweeps. Line marked with circles: batch PCA. Crosses: batch PCA from robust data vectors. Asterisks: approximative robust Newton type Oja's PCA subspace rule. Squares: robust subspace PAST algorithm.

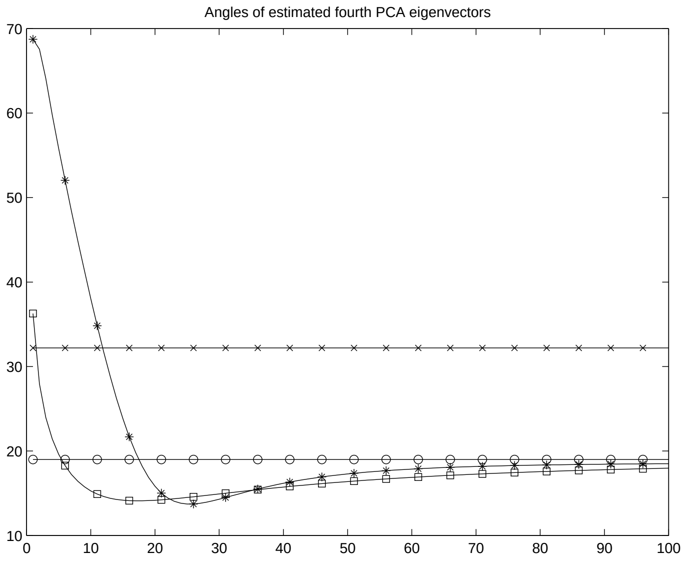

Figure 11: Angles between the fourth "true" PCA eigenvector and its estimates in an experiment with the Forest Fires data set in impulsive noise for different methods versus the number of sweeps. Line marked with circles: batch PCA. Crosses: batch PCA from robust data vectors. Asterisks: approximative robust Newton type Oja's PCA subspace rule. Squares: robust subspace PAST algorithm.

5.3 Missing values

Experiment 5: Imputation algorithm for missing values, simulated data

In this experiment, we studied the performance of the iterative imputation algorithm described in Section 4 for simulated data which contains both outliers and missing values. As before, the data vectors $\mathbf{x}_i$ were 5-dimensional. They were first generated so that their components were zero-mean statistically independent Gaussian distributed random numbers with variances $\sigma_1^2 = 5.0$, $\sigma_2^2 = 3.0$, $\sigma_3^2 = 2.0$, $\sigma_4^2 = 1.0$, and $\sigma_5^2 = 0.6$. The number of samples (data vectors) was 200. Both missing values and outliers were generated independently of each other randomly separately for each component of the data vectors with probability 0.1. Outliers were generated by adding to the the components of data vectors uniformly distributed random numbers in the interval $[-10, +10]$ with probability 0.1. The constant for robust data vectors in (43) was $c = 2.5$.

Figure 12 shows the angles between the true 2-dimensional PCA subspace and its estimates for different methods as a function of the number of sweeps. The imputation method performs for both standard batch PCA and robust batch PCA often the best when the missing values in the data are replaced simply by zeros, which is the mean of each component in the data. Iterations of the imputation algorithm provide during first sweeps much worse results. With more iterations, the results of the imputation method get gradually better, and in this simulation they become eventually a little better than for the initial values. The results of the imputation method vary greatly in different simulations even for the same parameters remain the same, and they are typically worse for the eigenvector estimates than for the PCA subspace. We emphasize that the results for batch PCA (line with circles) and robust batch PCA (line with crosses) are unrealistic and for reference and comparison purposes only, because they have been obtained with complete data which contains outliers but no missing values. In this particular simulation it is remarkable that the imputation algorithm using robust data vectors achieves better final results with missing values than the compared batch PCA methods without any missing values.

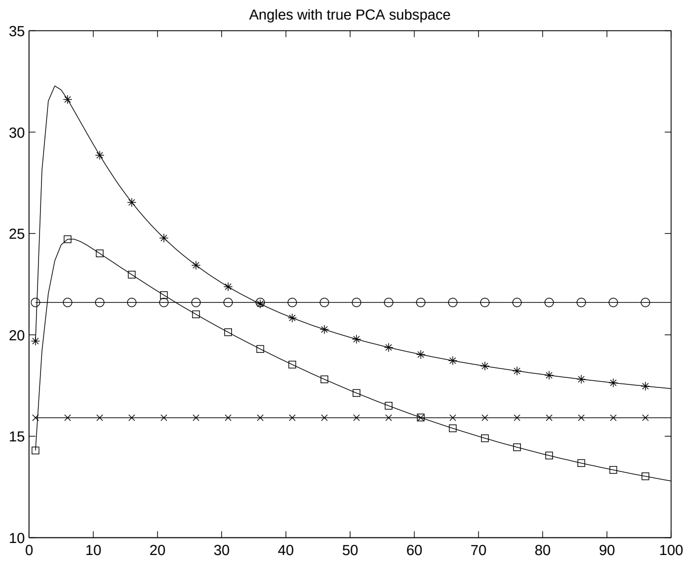

Figure 12: Angles between the true PCA subspace and estimated ones for different methods versus the number of sweeps. Two-dimensional PCA subspace was estimated from simulated 5-dimensional data. The probability of both missing values and outliers was 0.1. Line marked with circles: batch PCA with outliers but no missing values. Crosses: batch PCA from robust data vectors with outliers but no missing values. Asterisks: Batch PCA using the imputation algorithm for the missing values. Squares: robust batch PCA using the imputation algorithm for the missing values.

Experiment 6: Treatment of missing values for forest fires data

We studied the performance of the iterative imputation algorithm and the nearest neighbor method in handling missing values described in Section 4 also for the forest fires data set, which was preprocessed similarly as in previously. Note that the nearest neighbour method is not meaningful for the simulated data used in the previous experiment, because the components of its data vectors are statistically independent Gaussian random number and thus mutually uncorrelated. By finding the nearest neighbor of a data vector nothing can be deduced about the value of its missing component(s). However, for the forest fires data the nearest neighbor method is more meaningful, and we compare it in this experiment with the imputation method in handling missing values.

In this experiment, we added both outliers and missing values to the forest fires data set randomly for each component with probability 0.15 and independently of each other. The constant for robust data vectors in (43) was $c = 2.5$. The “true” 4-dimensional PCA subspace was computed from the forest fires data set with no missing values and outliers.

Figure 13 shows the angles between the “true” PCA subspace and its estimates for different methods as a function of the number of sweeps. It should be noted that the results of batch PCA (line marked with circles) and batch PCA from robust data vectors (line marked with crosses) are unrealistic if there are missing values, because they have been computed by assuming that there are no missing values in the data set. These results have been shown as a reference for the quality of results obtained using the imputation and nearest neighbor method when missing

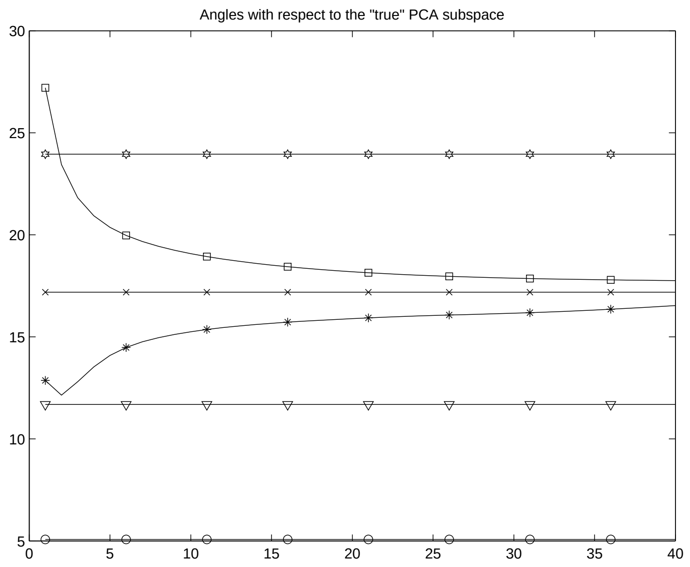

Figure 13: Angles between the “true” PCA subspace and estimated ones for different methods versus the number of sweeps. Four-dimensional PCA subspace was estimated from the forest fires data. The probability of both missing values and outliers was 0.15. Circles: batch PCA with outliers but no missing values. Crosses: batch PCA from robust data vectors with outliers but no missing values. Asterisks: Batch PCA using the imputation algorithm for the missing values. Squares: robust batch PCA using the imputation algorithm for the missing values. Triangles: Nearest neighbor approach for missing values in batch PCA. Stars: Nearest neighbor approach for missing values in batch PCA from robust data vectors.

values are present. The results are somewhat surprising especially compared with artificially generated data. First, standard batch PCA performs always better than batch PCA computed from robust data vectors. This holds for the results with no missing values (line marked with circles versus line marked with crosses), imputation method (curve marked with asterisks versus curve marked with squares), and nearest neighbor method for missing values (line marked with triangles versus line marked by stars). Note that for artificially generated data just the contrary is true. The reason is probably that because the true theoretical PCA subspace is not known, we estimated it using standard batch PCA from the data vectors where there are no missing values and outliers. This favors batch PCA also when missing values and outliers are present compared with the case in which we would estimate the “true” PCA subspace from robust data vectors using batch PCA.

It is also interesting and somewhat surprising that the imputation method behaves qualitatively differently for standard batch PCA and batch PCA estimated from robust data vectors. For standard batch PCA, the results get worse with iterations (curve marked with asterisks), while for robust batch PCA (curve marked by squares) they improve. Of the methods for handling missing values the nearest neighbor method for standard batch PCA (line marked with triangles) performs the best, but when using robust data vectors the imputation method (curve marked with squares) is clearly better than the nearest neigbor method (line marked with stars).

Figures 14-17 show the results provided by the same methods for the first 4 PCA eigenvectors. For the two first PCA eigenvectors, batch PCA with no missing values (line marked with circles) and batch PCA from robust data vectors with no missing values (line marked with crosses) have almost the same accuracy, and this holds even more when the nearest neighbor method is used for missing values. Then the differences in performance between these two methods are very small,

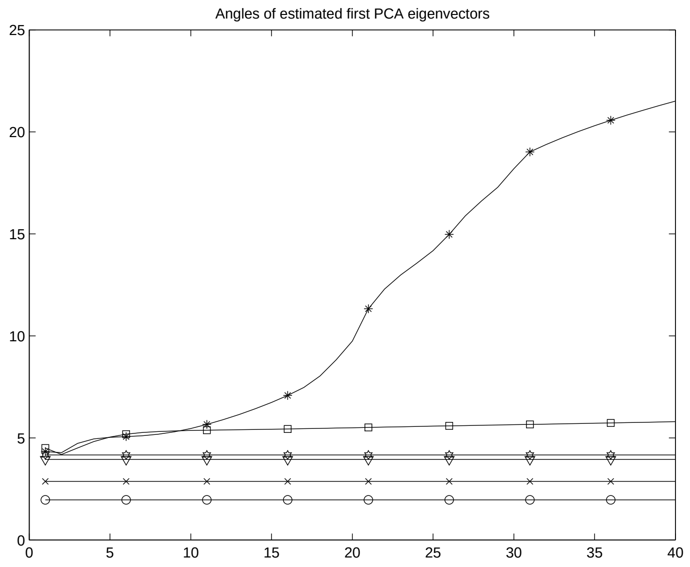

Figure 14: Angles between the “true” first PCA eigenvector and its estimates for different methods versus the number of sweeps for the forest fires data. Circles: batch PCA with outliers but no missing values. Crosses: batch PCA from robust data vectors with outliers but no missing values. Asterisks: Batch PCA using the imputation algorithm for the missing values. Squares: robust batch PCA using the imputation algorithm for the missing values. Triangles: Nearest neighbor approach for missing values in batch PCA. Stars: Nearest neighbor approach for missing values in batch PCA from robust data vectors.

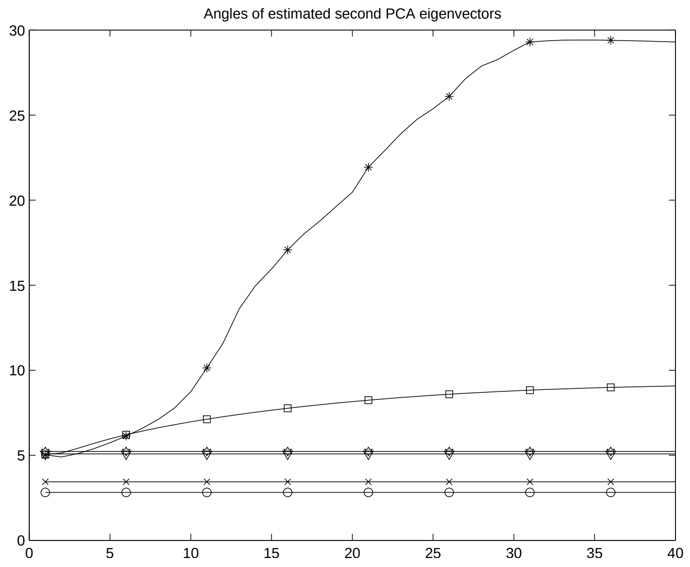

Figure 15: Angles between the “true” second PCA eigenvector and its estimates for different methods versus the number of sweeps for the forest fires data. Circles: batch PCA with outliers but no missing values. Crosses: batch PCA from robust data vectors with outliers but no missing values. Asterisks: Batch PCA using the imputation algorithm for the missing values. Squares: robust batch PCA using the imputation algorithm for the missing values. Triangles: Nearest neighbor approach for missing values in batch PCA. Stars: Nearest neighbor approach for missing values in batch PCA from robust data vectors.

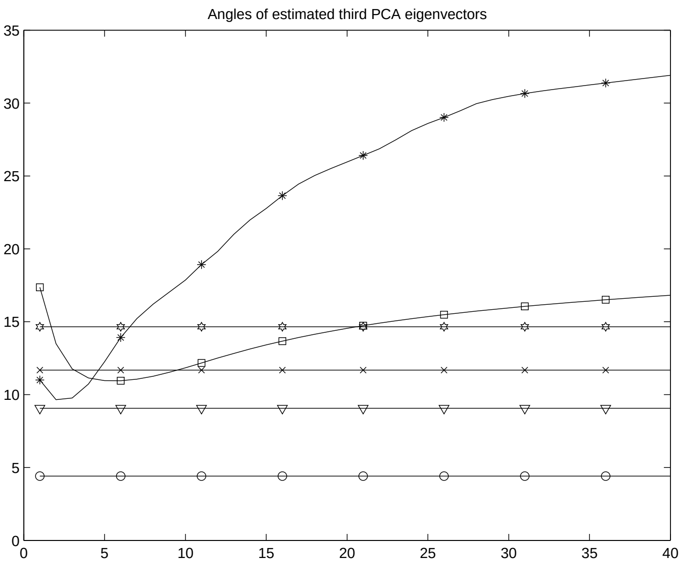

Figure 16: Angles between the “true” third PCA eigenvector and its estimates for different methods versus the number of sweeps for the forest fires data. Dark blue line: batch PCA with outliers but no missing values. Circles: batch PCA with outliers but no missing values. Crosses: batch PCA from robust data vectors with outliers but no missing values. Asterisks: Batch PCA using the imputation algorithm for the missing values. Squares: robust batch PCA using the imputation algorithm for the missing values. Triangles: Nearest neighbor approach for missing values in batch PCA. Stars: Nearest neighbor approach for missing values in batch PCA from robust data vectors.

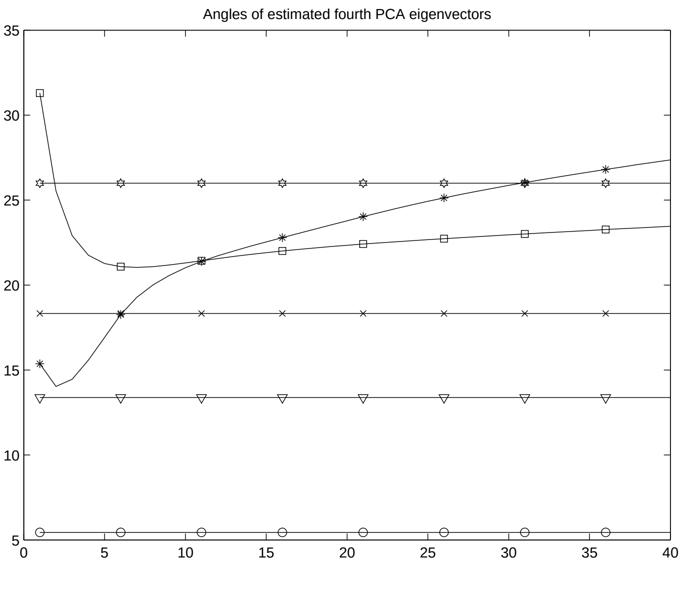

Figure 17: Angles between the “true” fourth PCA eigenvector and its estimates for different methods versus the number of sweeps for the forest fires data. Circles: batch PCA with outliers but no missing values. Crosses: batch PCA from robust data vectors with outliers but no missing values. Asterisks: Batch PCA using the imputation algorithm for the missing values. Squares: robust batch PCA using the imputation algorithm for the missing values. Triangles: Nearest neighbor approach for missing values in batch PCA. Stars: Nearest neighbor approach for missing values in batch PCA from robust data vectors.

hardly distinguishable. But for the third and fourth PCA eigenvectors the differences in the accuracy between batch PCA and robust batch PCA grow large both without missing values and when missing values are handled using the nearest neighbor method. The imputation method perform poorly especially in context with standard batch PCA when estimating the individual eigenvectors. Because its results are better in estimating the PCA subspace, the estimated eigenvectors obviously contain large portions of the other eigenvectors to be estimated. If one chooses to use the imputation method in context with batch PCA, it is best to just replace the missing elements with zeros and perform only one iteration. Finally, the performance of the imputation method in context with robust data vectors is contradictory in the sense that the accuracy of the estimated PCA subspace increases with more sweeps, but for individual PCA eigenvectors the accuracy of their estimates decreases with sufficiently many sweeps. The results were qualitatively similar in the other experiments that we carried out with different parameter values.

6 Conclusions

Principal component analysis (PCA) is a well-known standard technique based on the second-order statistics of the data represented by its covariance matrix. PCA is commonly used for preprocessing and compressing data because of its optimal properties, namely maximization of variance and minimization of the mean-square representation error. PCA is used also for whitening the data, which normalizes the data with respect to its first-order (mean vector subtracted) and second-order statistics (covariance matrix becomes unit matrix). This is useful and necessary for example in many methods designed for independent component analysis, allowing them to concentrate on the higher-order statistics of the data.

However, problems arise in practical situations when the data contains outliers and/or missing values. Several methods have been introduced for robust PCA when the data contains outliers and on the other hand for handling missing values, but both these cases have usually not been considered simultaneously. In this paper, we introduce several methods for robust PCA, of which the robust version of the least-squares based PAST algorithm performs the best. Also a simple method for processing the components of the data vectors directly for increasing their robustness is useful, while generalization of variance maximization does not lead to useful methods.

For handling missing values, we consider two methods, an imputation algorithm and nearest neighbor method. The performance of these method varies for the simulated data set and real-world forest fires data set that we have used for testing our methods. In general, the nearest neighbor method seems to be more useful for real-world data.

Acknowledgments

The author is grateful to Drs. Tapani Raiko and Alexander Ilin for useful comments and discussions on the topics of this paper. This work was supported by the Adaptive Informatics Research Centre of Excellence of the Academy of Finland.

References

[1] J. Ahn and J. Oh, “A constrained EM algorithm for principal component analysis”, Neural Computation, vol. 15, pp. 57–65, 2003.

[2] P. Allison, *Missing Data*, Quantitative Applications in Social Sciences, vol. 136, SAGE Publ., Thousand Oaks, California, USA, 2001.

[3] A. Anderson, A. Basilevsky, and D. Hum, “Missing data: a review of the literature”. In P. Rossi, J. Wright, and A. Anderson, *Handbook of Survey Research*, pp. 415–494, 1983.

[4] C. Archambeau, N. Delannay, and M.Verleysen, “Robust probabilistic projections”, in Proc. of the 23rd Int. Conf. on Machine Learning (ICML2006), Pittsburgh, Pennsylvania, pp. 33–40, 2006.

[5] P. Baldi and K. Hornik, “Neural networks for principal component analysis: Learning from examples without local minima”, Neural Networks, vol. 2, 1989, pp. 53–58.

[6] R. Bell and Y. Koren, “Scalable collaborative filtering with jointly derived neighborhood interpolation weights”, in Proc. of the 2007 Seventh IEEE Int. Conf. on Data Mining, October 2007, pp. 43–52. IEEE Computer Society, Washington, DC, USA, 2007.

[7] C. Bishop, *Pattern Recognition and Machine Learning,* Springer 2006.

[8] S. Choi, “Wake-Sleep PCA”, in Proc. of the IEEE Int. J. Conf. on Neural Networks, Orlando, Florida, USA, August 2007, pp. 2432–2435.

[9] A. Cichocki and S. Amari, *Adaptive Blind Signal and Image Processing - Learning Algorithms and Applications*, Wiley 2002.

[10] P. Cortez and A. Morais, “A data mining approach to predict forest fires using meteorological data”. In J. Neves, M. Santos, and J. Machado (Eds.), New Trends in Artificial Intelligence, Proc. of the 13th Portuguese Conference on Artificial Intelligence (EPIA 2007), Guimaraes, Portugal, December 2007, pp. 512–523. Available at `http://www3.dsi.uminho.pt/pcortez/fires.pdf`.

[11] C. Ding, D. Zhou, X. He, and H. Zha, “R_1-PCA: rotational invariant L_1-norm principal component analysis for robust subspace factorization”, in Proc. of the 23rd Int. Conf. on Machine Learning (ICML2006), Pittsburgh, PA, USA, 2006.

[12] J. Gao, “Robust L1 principal component analysis and its Bayesian variational inference”, Neural Computation, Vol. 20, 2008, pp. 555–572.

[13] F. Ham and I. Kostanic, *Principles of Neurocomputing for Science and Engineering*, McGraw-Hill 2001.

[14] F. Hampel, E. Ronchetti, P. Rousseouw, and W. Stahel, *Robust Statistics*, Wiley 1986.

[15] P. Huber, *Robust Statistics*, Wiley 1981.

[16] A. Hyvärinen, J. Karhunen, and A. Ilin, *Independent Component Analysis*, Wiley 2001.

[17] A. Ilin and T. Raiko, “Practical approaches to principal component analysis in the presence of missing values”, J. of Machine Learning Research, Vol. 11, 2010, pp. 1957-2000.

[18] I. Jolliffe, *Principal Component Analysis, 2nd ed.*, Springer 2002.

[19] J. Karhunen and J. Joutsensalo, “Representation and separation of signals using nonlinear PCA type learning”, Neural Networks, Vol. 7, No. 1, pp. 113–127, 1994.

[20] J. Karhunen and J. Joutsensalo, “Generalizations of principal component analysis, optimization problems, and neural networks”, Neural Networks, Vol. 8, No. 4, pp. 549–562, 1995.

[21] R. Little and D. Rubin, *Statistical Analysis with Missing Data*, Wiley, New York, USA, 2002.

[22] J. Luttinen, A. Ilin, and J. Karhunen, “Bayesian robust PCA for incomplete data”, in Proc. of the 8th Int. Conf. on Independent Component Analysis and Signal Separation (ICA2009). Lecture Notes in Computer Science, vol. 5441, pp. 66–73, Springer-Verlag, 2009.

[23] C. Nikias and J. Mendel, “Signal processing with higher-order spectra”, IEEE Signal Processing Magazine, vol. 10, pp. 10–37, 1993.

[24] P. Pajunen and J. Karhunen, “Least-squares methods for blind source separation based on nonlinear PCA”, Int. J. Neural Systems, Vol. 8, Nos 5-6, pp. 601–612, 1998.

[25] F. Palmieri and J. Zhu, “Self-association and Hebbian learning in linear neural networks”, IEEE Trans. on Neural Networks, Vol. 6, no. 5, pp. 1165–1184, 1995.

[26] T.Raiko, A. Ilin, and J. Karhunen, “Principal component analysis for large scale problems with lots of missing values”, in Proc. of the 12th European Conf. on Machine Learning (ECML 2007). Lecture Notes in Artificial Intelligence, vol. 4701, pp. 691–698, Springer-Verlag, 2007.

[27] T.Raiko, A. Ilin, and J. Karhunen, “Principal component analysis for sparse high-dimensional data”, in Proc. of the 14th Int. Conf. on Neural Information Processing (ICONIP2007). Lecture Notes in Computer Science, vol. 4985, pp. 566–575, Springer-Verlag, 2008.

[28] S. Roweis, “EM algorithm for PCA and SPCA”. In M. Jordan et al. (Eds.), Advances in Neural Information Processing Systems, Vol. 10, pp. 626–632, MIT Press, 1998.

[29] T. Sanger, “Optimal unsupervised learning in a single-layer linear feedforward network”, Neural Networks, Vol. 2, pp. 459–473, 1989.

[30] M. Tipping and C. Bishop, “Mixtures of probabilistic principal component analyzers”, Neural Computation, Vol. 11, No. 2, pp. 443–482, 1999.

[31] F. de la Torre and M. Black, “A framework for robust subspace learning”, Int. J. of Computer Vision, Vol. 54, pp. 117–142, 2003.

[32] UCI (Univ. of California at Irvine) Machine Learning Repository, Center for Machine Learning and Intelligent Systems. Available at `http://archive.ics.uci.edu/ml/`.

[33] A. Weingessel and K. Hornik, “A robust subspace algorithm for principal component analysis”, Int. J. Neural Systems, Vol. 13, No. 5, 2003, pp. 307–313.

[34] L. Xu, “Linear mean-square error reconstruction principle for self-organizing neural-nets”, Neural Networks, Vol. 6, pp. 627–648, 1993.

[35] L. Xu and A. Yuille, “Robust principal component analysis by self-organizing rules based on statistical physics approach”, IEEE Trans. on Neural Networks, Vol. 6, No. 1, pp. 131–143, 1995.

[36] B. Yang, “Projection approximation subspace tracking”, IEEE Trans. on Signal Processing, Vol. 43, pp. 95–107, 1995.

[37] B. Yang, “Asymptotic convergence analysis of the projection approximation subspace tracking algorithm”, Signal Processing, vol. 50, pp. 129-136, 1996.

[38] J. Zhao and Q. Jiang, “Probabilistic PCA for t distributions”, Neurocomputing, Vol. 69, 2006, pp. 2217–2226.