Journal of Machine Learning Research 11 (2010) 1957-2000 Submitted 6/08; Revised 11/09; Published 7/10

Practical Approaches to Principal Component Analysis in the Presence of Missing Values

Alexander Ilin ALEXANDER.ILIN@TKK.FI
Tapani Raiko TAPANI.RAIKO@TKK.FI
Adaptive Informatics Research Center
Aalto University School of Science and Technology
P.O. Box 15400, FI-00076 Aalto, Finland

Editor: Tommi Jaakkola

Abstract

Principal component analysis (PCA) is a classical data analysis technique that finds linear transformations of data that retain the maximal amount of variance. We study a case where some of the data values are missing, and show that this problem has many features which are usually associated with nonlinear models, such as overfitting and bad locally optimal solutions. A probabilistic formulation of PCA provides a good foundation for handling missing values, and we provide formulas for doing that. In case of high dimensional and very sparse data, overfitting becomes a severe problem and traditional algorithms for PCA are very slow. We introduce a novel fast algorithm and extend it to variational Bayesian learning. Different versions of PCA are compared in artificial experiments, demonstrating the effects of regularization and modeling of posterior variance. The scalability of the proposed algorithm is demonstrated by applying it to the Netflix problem.

Keywords: principal component analysis, missing values, overfitting, regularization, variational Bayes

1. Introduction

Principal component analysis (PCA) is a data analysis technique that can be traced back to Pearson (1901). It can be used to compress data sets of high dimensional vectors into lower dimensional ones. This is useful, for instance, in visualization and feature extraction. PCA has been extensively covered in the literature (e.g., Jolliffe, 2002; Bishop, 2006; Diamantaras and Kung, 1996; Haykin, 1989; Cichocki and Amari, 2002). PCA can be derived from a number of starting points and optimization criteria. The most important of these are minimization of the mean-square error in data compression, finding mutually orthogonal directions in the data having maximal variances, and decorrelation of the data using orthogonal transformations.

In the data compression formulation, PCA finds a smaller-dimensional linear representation of data vectors such that the original data could be reconstructed from the compressed representation with the minimum square error. Assume that we have n $d \times 1$ data vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n$ that are modeled as

$$\mathbf{y}_j \approx \mathbf{W}\mathbf{x}_j + \mathbf{m}, \tag{1}$$

where $\mathbf{W}$ is a $d \times c$ matrix, $\mathbf{x}_j$ are $c \times 1$ vectors of principal components, and $\mathbf{m}$ is a $d \times 1$ bias vector. We assume that $c \leq d \leq n$. Principal subspace methods (e.g., Cichocki and Amari, 2002;

Diamantaras and Kung, 1996) find $\mathbf{W}$, $\mathbf{x}_j$ and $\mathbf{m}$ such that the reconstruction error

$$C = \sum_{j=1}^{n} \|\mathbf{y}_j - \mathbf{W}\mathbf{x}_j - \mathbf{m}\|^2 \tag{2}$$

is minimized. In matrix notation, the data vectors and principal components can be compiled into $d \times n$ and $c \times n$ matrices $\mathbf{Y} = [\mathbf{y}_1\ \mathbf{y}_2\ \ldots\ \mathbf{y}_n]$ and $\mathbf{X} = [\mathbf{x}_1\ \mathbf{x}_2\ \ldots\ \mathbf{x}_n]$, and y_{ij}, w_{ik} and x_{kj} denote the elements of the matrices $\mathbf{Y}$, $\mathbf{W}$ and $\mathbf{X}$, respectively. The bias matrix $\mathbf{M}$ contains n copies of the bias vector $\mathbf{m}$ as its columns. Principal subspace methods find $\mathbf{W}$ and $\mathbf{X}$ such that $\mathbf{Y} \approx \mathbf{W}\mathbf{X} + \mathbf{M}$ and the minimized cost function is the sum of the squared elements (or Frobenius norm) of matrix $\mathbf{Y} - \mathbf{W}\mathbf{X} - \mathbf{M}$:

$$C = \|\mathbf{Y} - \mathbf{W}\mathbf{X} - \mathbf{M}\|_{\mathrm{F}}^2 . \tag{3}$$

Without any further constraints, there exist infinitely many ways to perform a decomposition that minimizes (2) or equivalently (3). This can be seen by noting that any rotation or scaling of $\mathbf{W}$ can be compensated by rotating or scaling $\mathbf{X}$ accordingly, leaving the product $\mathbf{W}\mathbf{X}$ the same. However, the subspace spanned by the column vectors of the matrix $\mathbf{W}$, called the *principal subspace*, is unique.

PCA finds a specific representation of the principal subspace. It is traditionally defined using the requirement that the column vectors of $\mathbf{W}$ are mutually orthogonal, have unit length and, furthermore, for each $k = 1, \ldots, c$, the first k vectors form the k-dimensional principal subspace. This makes the solution practically unique (except for changing the sign, and in the special case of having equal eigenvalues, e.g., Diamantaras and Kung, 1996; Jolliffe, 2002; Haykin, 1989). In this article, we use the term PCA for methods which seek representations (1) by minimizing the error (2). Thus we assume that once the principal subspace is found, it can be transformed into the PCA solution. This is indeed true for the case of fully observed vectors $\mathbf{y}_j$ but can be more difficult in the case with missing values, as we discuss in Section 4.

Let us now consider the same problem when the data matrix $\mathbf{Y}$ has missing entries. In this paper, we make the typical assumption that values are *missing at random* (MAR) (Little and Rubin, 1987), that is, given the observed data, the missingness does not depend on the unobserved data or latent variables. An example where the assumption does not hold is when out-of-scale measurements are marked missing. In the following example, the data matrix contains $N = 9$ observed values and 6 missing values (marked with a sign $\times$):

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} & \times \\ y_{21} & y_{22} & \times & \times & y_{25} \\ \times & \times & y_{33} & y_{34} & \times \end{bmatrix} .$$

A natural extension of PCA for the case with missing values would be to find a representation such that $\mathbf{Y} \approx \mathbf{W}\mathbf{X} + \mathbf{M}$ for the observed values. The rest of the matrix $\mathbf{W}\mathbf{X} + \mathbf{M}$ can be taken as the reconstruction of missing values.

Although the PCA problem in the presence of missing values seems to be as easy as classical PCA, there are some important distinctions: 1) There is no analytical solution available since even the estimation of the data covariance matrix is nontrivial. Therefore, iterative learning procedures must be exploited. 2) The optimized cost function typically has multiple local minima and thus finding the optimal solution is more difficult. 3) There is no analytical solution even for the bias

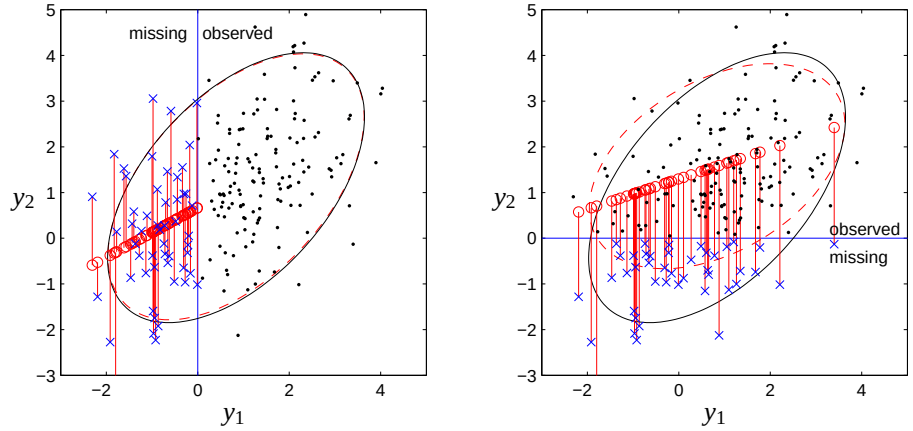


Figure 1: Two examples of PCA with missing values. In the left, the missing value mechanism is *missing at random (MAR)*, while on the right, it is *missing not at random (MNAR)*. The black dots represent fully observed samples, while the blue crosses represent observations where only the horizontal value has been observed. The red circles represent the reconstructions of the missing values produced by VBPCA. The dashed red ellipses represent the estimated data covariance. On the left, the estimated covariance is quite close to the data covariance (black ellipse).

term **m** in (1), which is not generally equal to the row-wise mean of the data matrix, as in classical PCA. 4) Standard PCA approaches can easily lead to overfitting, thus regularization is often required. 5) The algorithms may require heavy computations, especially for large-scale problems. 6) The concept of the PCA basis in the principal subspace is not easily generalized in the presence of missing values. 7) The choice of the dimensionality of the principal subspace is generally more difficult than in classical PCA. Thus, the PCA problem has many features which are usually associated with nonlinear models. This paper discusses some of these important issues providing illustrative examples and presenting ways to overcome possible difficulties.

All the methods studied in this work assume MAR, so they cannot be expected to work in the *missing not at random (MNAR)* case. Fig. 1 gives a first example of PCA with missing values. The data contain 200 samples generated from a Gaussian distribution represented by the black ellipse. In the first case (left subfigure), the vertical value is missing when the horizontal value is negative. This setting is MAR since the missingness depends only on the observed data. In the second case (right subfigure), the vertical value is missing when the vertical value itself is negative. Now the missingness depends on the missing data, and the missing value mechanism is thus MNAR. As expected, the variational Bayesian PCA (VBPCA, see Section 3.3) works well in the MAR case and gives reasonable reconstructions for the missing values. Also, the vertical part of the bias term **m** is estimated to be 1.13 which is much closer to the true value 1.10 than the row-wise mean 1.50 of the observed values in the data.

PCA in the presence of missing values can be relevant for many data sets which appear in practical applications. Sometimes, data sets contain relatively few missing observations and the

problem is to adjust standard PCA algorithms to handle partially observed instances. The choice of the algorithm is not very crucial in such a simple case, as most of the approaches would provide similar solutions. However, in other applications, the available observations can be very sparse and the modeling task is often to reconstruct the missing part from the observed data only. Examples include collaborative filtering problems which is the task of predicting preferences by using other people's preferences (e.g., Hofmann, 2004), and historical data reconstruction in climatic records (e.g., Kaplan et al., 1997).

Historically, the missing value problem in PCA was first studied by Dear (1959) who used only one component and just one imputation iteration (see below). It was based on the minimum mean-square error formulation of PCA introduced by Young (1941). Christoffersson (1970) also used a one-component model for reconstructing missing values. Wiberg (1976) first suggested to directly minimize the mean-square error of the observed part of the data. An algorithm by de Ligny et al. (1981) already worked with up to half of the values missing. The missing values problem using a multivariate normal distribution has been studied even earlier than using PCA, for instance, by Anderson (1957). More historical references can be found in the book by Jolliffe (2002).

More recently, PCA with missing values was studied by Grung and Manne (1998). They proposed using either the imputation or the faster alternating **W**–**X** algorithm (see below). They discussed the overfitting problem and suggested to delete troublesome rows or columns from data. Tipping and Bishop (1999) introduced the probabilistic formulation of PCA (PPCA). Although they mentioned shortly missing data, they did not provide the formulas to be used for incomplete data. Bishop (1999) introduced variational Bayesian PCA (VBPCA) for choosing the number of components in PCA. Raiko and Valpola (2001) reconstructed missing values with VBPCA to compare some nonlinear models to it. Oba et al. (2003) applied VBPCA for missing value estimation in gene expression profile data, and mentioned that it performs much better than the existing methods for missing value estimation.

The present article reviews possible approaches to the problem with the emphasis on probabilistic models and Bayesian methods. The rest of the article is organized as follows. In Section 2, we present classical algorithms for PCA based on minimizing the reconstruction error similar to (2) and the method based on estimation of the covariance matrix (imputation algorithm). We review how the algorithms are normally used for fully observed data and explain how they can be adapted to the case with missing values. We explain possible difficulties of the standard methods including bad locally optimal solutions, numerical problems and overfitting. Simple examples are given to illustrate these problems. In the same section, we discuss the properties of the imputation algorithm, such as implicit remedies against overfitting and overlearning and its expectation-maximization (EM) interpretation.

Section 3 presents probabilistic models for PCA which provide a good foundation for handling missing values. First, we introduce formulas for the probabilistic PCA model in the case with missing values. Then, we describe Bayesian regularization for handling problems that arise when data are sparse. We provide formulas for performing maximum a posteriori estimation and for variational Bayesian inference.

In Section 4, we propose an extension of the PCA-basis notion to the case of incomplete data. We also show how to find the PCA basis in a principal subspace estimated by different methods. Section 5 provides formulas for computing missing value reconstructions and the variance of the predicted values. We briefly discuss possible ways to select the right number of principal components to obtain reliable reconstructions.

Section 6 discusses large-scale problems and computational complexity. In case of high dimensional and very sparse data, overfitting becomes a severe problem and traditional algorithms for PCA are very slow. We introduce a novel fast optimization algorithm and show how to use it in Bayesian models.

In Section 7, different versions of PCA are compared in artificial experiments, demonstrating the effects of regularization and modeling of posterior variance. We demonstrate the importance of tricks such as fixing hyperparameters for the early learning in variational Bayesian methods. The scalability of the proposed algorithm is demonstrated by applying it to the Netflix problem. Finally, we conclude in Section 8.

1.1 Notation

The following notation is used throughout the article. Bold capital letters denote matrices, bold lower-case letters denote vectors, and normal lower-case letters denote scalars. The basic model equation is $\mathbf{y}_j \approx \mathbf{W}\mathbf{x}_j + \mathbf{m}$, where column vectors $\mathbf{y}_j$ are the data cases, $\mathbf{W}$ is the matrix that maps the principal components $\mathbf{x}_j$ to the data, and $\mathbf{m}$ is the bias vector. Indices $i = 1,\ldots,d$ and $j = 1,\ldots,n$ go over the rows and columns of $\mathbf{Y}$, respectively. The index of a principal component is denoted by $k = 1,\ldots,c$. Notation ij is used for the index of y_{ij}, the ij-th element of matrix $\mathbf{Y}$. O is the set of indices ij corresponding to *observed* values y_{ij}, O_i is the set of indices j (similarly O_j is the set of indices i) for which y_{ij} is observed. $|O_i|$ is the number of elements in O_i and $N = |O|$ is the number of observed data elements. Notation $\mathbf{A}_{i:}$ and $\mathbf{A}_{:j}$ is used for the i-th row and j-th column of a matrix $\mathbf{A}$, respectively. Both $\mathbf{A}_{i:}$ and $\mathbf{A}_{:j}$ are column vectors. The i-th row of $\mathbf{W}$ is denoted by $\mathbf{w}_i = \mathbf{W}_{i:}$, and the j-th column of $\mathbf{X}$ is $\mathbf{x}_j = \mathbf{X}_{:j}$. Table 1 lists the symbols used in the paper.

2. Least Squares Techniques

In this section, we present classical algorithms for PCA based on minimizing the reconstruction error similar to (2) and the method based on estimation of the covariance matrix.

2.1 The Cost Function

The minimum mean-square error compression of data is the formulation for PCA (Young, 1941) that can be generalized to the case with missing values in a very straightforward manner. The cost function (2) is adapted such that the sum is taken over only those indices i and j for which the data entry y_{ij} is observed (Wiberg, 1976):

$$C = \sum_{ij \in O} (y_{ij} - \hat{y}_{ij})^2 , \tag{4}$$

$$\hat{y}_{ij} = \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j + m_i = \sum_{k=1}^{c} w_{ik} x_{kj} + m_i . \tag{5}$$

In this section, we consider algorithms optimizing cost function (4), which we call the least squares (LS) approach to PCA of incomplete data.

The cost function (2) in the fully observed case has practically unique (up to an arbitrary rotation in the principal subspace) global minimum w.r.t. model parameters $\mathbf{W}$, $\mathbf{X}$, $\mathbf{m}$ provided that all eigenvalues of the sample covariance matrix are distinct. Srebro and Jaakkola (2003) showed that

d	dimensionality of the data vectors (indexed by i)
n	number of data vectors (indexed by j)
c	number of principal components (indexed by k)
$\mathbf{Y} = \{y_{ij}\}$	$d \times n$ data matrix
$\mathbf{y}_j$	d-dimensional data vector (j-th column of $\mathbf{Y}$)
$\hat{y}_{ij}$	reconstruction of the data element y_{ij}
$\mathbf{m} = \{m_i\}$	d-dimensional bias vector
$\mathbf{M}$	$= [\mathbf{m}\ \mathbf{m}\ \ldots\ \mathbf{m}]$, $d \times n$ bias matrix
$\overline{m}_i$	posterior mean of m_i (scalar)
$\widetilde{m}_i$	posterior variance of m_i (scalar)
$\mathbf{W} = \{w_{ik}\}$	$d \times c$ matrix for the mapping from principal components to the data
$\mathbf{W}_{:k}$	k-th column of matrix $\mathbf{W}$ (d-dimensional vector)
$\mathbf{w}_i$	i-th row of $\mathbf{W}$ (c-dimensional vector)
$\overline{\mathbf{w}}_i$	posterior mean of $\mathbf{w}_i$ (c-dimensional vector)
$\overline{\mathbf{W}}$	$= [\overline{\mathbf{w}}_1\ \overline{\mathbf{w}}_2\ \ldots\ \overline{\mathbf{w}}_d]^{\mathrm{T}}$, $d \times c$ matrix of posterior means of $\mathbf{w}_i$
$\mathbf{\Sigma}_{\mathbf{w}_i}$	posterior covariance of $\mathbf{w}_i$ ($c \times c$ matrix)
$\widetilde{w}_{ik}$	posterior variance of w_{ik} (scalar)
$\mathbf{X} = \{x_{kj}\}$	$c \times n$ matrix of principal components
$\mathbf{x}_j$	c-dimensional principal component vector (j-th column of $\mathbf{X}$)
$\overline{\mathbf{x}}_j$	posterior mean of $\mathbf{x}_j$ (c-dimensional vector)
$\overline{\mathbf{X}}$	$= [\overline{\mathbf{x}}_1\ \overline{\mathbf{x}}_2\ \ldots\ \overline{\mathbf{x}}_n]$, $c \times n$ matrix of posterior means of $\mathbf{x}_j$
$\mathbf{\Sigma}_{\mathbf{x}}$	posterior covariance of $\mathbf{x}_j$ (same $c \times c$ matrix for each j)
$\mathbf{\Sigma}_{\mathbf{x}_j}$	posterior covariance of $\mathbf{x}_j$ (separate $c \times c$ matrix for each j)
$\widetilde{x}_{kj}$	posterior variance of x_{kj} (scalar)
C	cost function to be minimized
O	set of indices i, j for which y_{ij} is observed
O_i	set of indices j for which y_{ij} is observed
O_j	set of indices i for which y_{ij} is observed
γ	learning rate (positive scalar)
ε_j	d-dimensional noise vector (j-th out of n vectors)
v_y	noise variance (scalar)
$\mathcal{N}(\mathbf{x}; \mu, \Sigma)$	Gaussian pdf over variable $\mathbf{x}$ with mean μ and covariance Σ
v_m	prior variance for the elements of $\mathbf{m}$ (scalar)
$v_{w,k}$	prior variance for the elements of $\mathbf{W}_{:k}$ (scalar for each k)
ξ	$= (v_y, v_{w,k}, v_m)$ hyperparameters
θ	parameters, typically $\theta = (\mathbf{W}, \mathbf{X}, \mathbf{m})$

Table 1: Symbols used in the paper. See also Section 1.1 about notation.

an extension to the weighted case, where each observation has a weight varying from 0 to 1, is a difficult non-convex problem where the cost function can have multiple local minima. We demonstrate this phenomenon for a simple data set with missing values, that is with weights either 0 (missing value) or 1 (observed value). In our example, model (1) with one principal component and fixed $\mathbf{m} = 0$ is fitted to the data

$$\mathbf{Y} = \begin{bmatrix} 0.8 & 0.8 \\ 1 & \times \\ \times & 1 \end{bmatrix} \approx \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \end{bmatrix}.$$

To discard the scaling ambiguity of the PCA solution, the column vector $\mathbf{W}$ can be restricted to have unit length. Thus, the minimized error can be represented as a function of two parameters which define a unit length vector in the three-dimensional space. The cost function in this simple example has three local minima, as shown in Fig. 2: The global minimum $\mathbf{W} = \pm[0.49\ 0.62\ 0.62]^{\mathrm{T}}$ corresponds to zero cost while the other two minima (close to $\pm[0\ 1\ 0]^{\mathrm{T}}$ and $\pm[0\ 0\ 1]^{\mathrm{T}}$) provide a non-zero error. Each of the two sub-optimal solutions reconstructs perfectly only three out of four observed values in $\mathbf{Y}$.

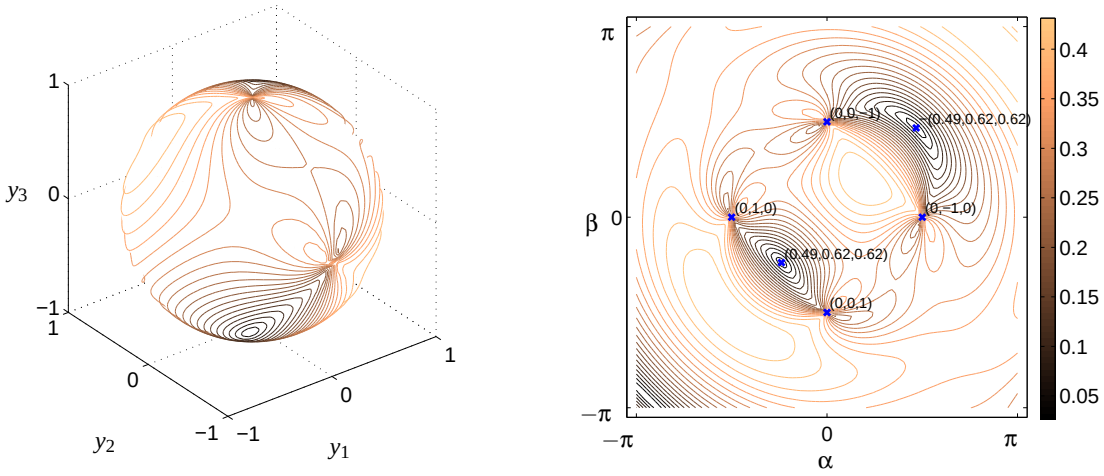


Figure 2: Example of local minima for the cost function (4). Data are three-dimensional ($d = 3$), the model has one principal component ($c = 1$) and therefore the PCA solution can be defined by a unit length vector $\mathbf{W}$. Left: The cost function plotted on a surface of unit length vectors. Right: The same plot using the Euler vector representation: The matrix $\mathbf{W}$ is constructed from α and β as $\mathbf{W} = e^{\mathbf{A}}[1\ 0\ 0]^{\mathrm{T}}$ with $\mathbf{A}$ a 3×3 matrix with four nonzero elements $a_{12} = -a_{21} = \alpha$ and $a_{13} = -a_{31} = \beta$. Coordinates in brackets correspond to coordinates in Euclidian space in the left hand side plot.

The cost function (4) can be minimized using any optimization procedure. Two possible approaches are presented in the following.

2.2 Alternating W–X Algorithm

Complete data. It is possible to optimize the cost function (3), and therefore to find the principal subspace, by updating $\mathbf{W}$ and $\mathbf{X}$ alternately. When either of these matrices is fixed, the other one

can be obtained from an ordinary least squares problem. We will further refer to this approach as the *alternating algorithm*.

The algorithm alternates between the updates

$$\mathbf{X} = (\mathbf{W}^{\mathrm{T}}\mathbf{W})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{Y}, \tag{6}$$

$$\mathbf{W} = \mathbf{Y}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}. \tag{7}$$

This iteration can be efficient when only a few principal components are needed, that is $c \ll d$ (Roweis, 1998). The bias vector $\mathbf{m}$ is the row-wise mean of the data and the above equations assume that it has been subtracted from the data as a preprocessing step.

Incomplete data. Grung and Manne (1998) studied the alternating algorithm in the case of missing values. In order to get the accurate least squares solution, we include the bias term into the estimation procedure, yielding the update rules

$$\mathbf{x}_j = \Big(\sum_{i\in O_j} \mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}\Big)^{-1} \sum_{i\in O_j} \mathbf{w}_i(y_{ij} - m_i)\,, \quad j = 1,\ldots,n, \tag{8}$$

$$m_i = \frac{1}{|O_i|}\sum_{j\in O_i}\left[y_{ij} - \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j\right], \tag{9}$$

$$\mathbf{w}_i = \Big(\sum_{j\in O_i} \mathbf{x}_j\mathbf{x}_j^{\mathrm{T}}\Big)^{-1} \sum_{j\in O_i} \mathbf{x}_j(y_{ij} - m_i)\,, \quad i = 1,\ldots,d\,, \tag{10}$$

where m_i is the i-th element of $\mathbf{m}$.

2.3 Gradient Descent Algorithm

Complete data. The basis of neural network implementation of PCA learning rules is a gradient descent optimization procedure. Such algorithms work online, processing only one input vector $\mathbf{y}_j$ at once. The learning rules implement *stochastic* gradient descent and the algorithms will eventually converge to a basis in the principal subspace. Each data vector may have to be processed several times for convergence. The same can also be implemented in a batch procedure.

Using gradient descent for minimization of (4) w.r.t. $\mathbf{W}$ yields the update rule

$$\mathbf{W} \leftarrow \mathbf{W} + \gamma(\mathbf{Y} - \mathbf{W}\mathbf{X})\mathbf{X}^{\mathrm{T}}\,, \tag{11}$$

where $\gamma > 0$ is called the learning rate. Minimization w.r.t. matrix $\mathbf{X}$ can be performed using the least squares solution in (6). Some neural algorithms use learning rules which either explicitly orthogonalize $\mathbf{W}$ or which yield an orthogonal $\mathbf{W}$ at the convergence. Then the update of $\mathbf{X}$ can be simplified to $\mathbf{X} = \mathbf{W}^{\mathrm{T}}\mathbf{Y}$, which together with (11) is the batch version of the Oja learning algorithm (Oja, 1983; Diamantaras and Kung, 1996). The bias $\mathbf{m}$ is again removed from the data as a preprocessing step.

Incomplete data. In the presence of missing values, the gradient descent update rule for $\mathbf{W}$ is

$$\mathbf{W} \leftarrow \mathbf{W} - \gamma\frac{\partial C}{\partial \mathbf{W}}\,, \qquad \text{with} \qquad \frac{\partial C}{\partial w_{ik}} = -2\sum_{j\in O_i}(y_{ij} - \hat{y}_{ij})x_{kj}\,,$$

where $\hat{y}_{ij}$ is given in (5). Matrix $\mathbf{X}$ could be updated using (8) but a gradient-based update can also be used:

$$\mathbf{X} \leftarrow \mathbf{X} - \gamma \frac{\partial C}{\partial \mathbf{X}}, \qquad \text{with} \qquad \frac{\partial C}{\partial x_{kj}} = -2 \sum_{i \in O_j} (y_{ij} - \hat{y}_{ij}) w_{ik}.$$

The bias term $\mathbf{m}$ can be updated using (9). As we discuss in Section 6.1, gradient-based learning can be computationally more efficient than the alternating algorithm because there is no need to compute matrices $\left(\sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}}\right)^{-1}$ and $\left(\sum_{j \in O_i} \mathbf{x}_j \mathbf{x}_j^{\mathrm{T}}\right)^{-1}$ in (8) and (10).

2.4 Numerical Problems and Overfitting

Least squares PCA methods can work well for data sets with few missing values but they may not be applicable to sparse data sets because of severe problems with overfitting. Suppose that for some j the number $|O_j|$ of observed measurements y_{ij} is smaller than the number of principal components c. Then, the corresponding least square problem is ill posed: the matrix $\sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}}$ is rank deficient and Equation (8) cannot be used. Moreover, even if $|O_j|$ is greater than c, matrix $\sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}}$ may be badly conditioned and the corresponding $\mathbf{x}_j$ can become infinitely large. This means that the parameters would be overfitted to explain well a few observed values but the generalization ability of the model would be poor (e.g., reconstruction of missing data would be very inaccurate). This is exactly what happens in the vicinity of the two local minima in the example in Fig. 2.

The same problem can happen when the rows of $\mathbf{W}$ are estimated using, for example, (10): matrix $\sum_{j \in O_i} \mathbf{x}_j \mathbf{x}_j^{\mathrm{T}}$ can be rank deficient or badly conditioned. This is more probable when some rows of $\mathbf{Y}$ contain very few measurements. However, these problems can generally appear for any data set and for any algorithm optimizing the cost function (4).

Overfitting problems can be avoided by using proper regularization. A common way to prevent unbounded growth of model parameters is to add terms which penalize large parameter values into the cost function. The amount of regularization can be determined, for instance, by cross-validation. Another possibility is to use probabilistic methods (see Section 3) in which the extra penalty terms come naturally from a probabilistic model.

2.5 Method Using Singular Value Decomposition

Complete data. Perhaps the most popular approach to PCA is based on singular value decomposition (SVD) of the data matrix or (equivalently) eigen-decomposition of the sample covariance matrix. SVD of the data matrix is given by:

$$\mathbf{Y} = \mathbf{U} \Sigma \mathbf{V}^{\mathrm{T}},$$

where $\mathbf{U}$ is a $d \times d$ orthogonal matrix, $\mathbf{V}$ is an $n \times n$ orthogonal matrix and Σ is a $d \times n$ pseudo-diagonal matrix (diagonal if $d = n$) with the singular values on the main diagonal (e.g., Haykin, 1989). The PCA solution is obtained by selecting the c largest singular values from Σ, by forming $\mathbf{W}$ from the corresponding c columns of $\mathbf{U}$, and $\mathbf{X}$ from the corresponding c rows of $\Sigma \mathbf{V}^{\mathrm{T}}$ (Jolliffe, 2002). The bias $\mathbf{m}$ is again removed from the data as a preprocessing step.

PCA can equivalently be defined using the eigen-decomposition of the $d \times d$ covariance matrix $\mathbf{C}$ of the column vectors of the data matrix $\mathbf{Y}$:

$$\mathbf{C} = \frac{1}{n} \mathbf{Y} \mathbf{Y}^{\mathrm{T}} = \mathbf{U} \mathbf{D} \mathbf{U}^{\mathrm{T}}.$$

The diagonal matrix **D** contains the eigenvalues of **C**, and the columns of the matrix **U** contain the unit-length eigenvectors of **C** in the same order (Cichocki and Amari, 2002; Diamantaras and Kung, 1996; Jolliffe, 2002; Haykin, 1989). Again, the columns of **U** corresponding to the largest eigenvalues are taken as **W**, and **X** is computed as $\mathbf{W}^{\mathrm{T}}\mathbf{Y}$. This approach can be more efficient for cases where $d \ll n$, since it avoids the computation of the $n \times n$ matrix **V**.

Incomplete data. The same approach cannot be directly used in the presence of missing values. Estimating the covariance matrix **C** becomes difficult. Let us consider a simple (but incorrect) estimate where we just leave out terms with missing values from the average for each element of **C**. For the data matrix below, we get

$$\mathbf{Y} = \begin{bmatrix} -1 & +1 & 0 & 0 & \times \\ -1 & +1 & \times & \times & 0 \\ \times & \times & -1 & +1 & \times \end{bmatrix}, \quad \mathbf{C} = \frac{1}{n}\mathbf{Y}\mathbf{Y}^{\mathrm{T}} = \begin{bmatrix} 0.5 & 1 & 0 \\ 1 & 0.667 & \times \\ 0 & \times & 1 \end{bmatrix}.$$

There are at least two problems. First, the estimated covariance 1 between the first and second components is larger than their estimated variances 0.5 and 0.667. This clearly leads to the situation where the covariance matrix is not positive (semi)definite and some of its eigenvalues are negative. Secondly, the covariance between the second and the third component could not be estimated at all. For these reasons, this is a viable option only if the number of missing values is not significant. There are many algorithms for finding a proper estimate of the covariance matrix (e.g., Ghahramani and Jordan, 1994; Boscardin and Zhang, 2004) but they are computationally intensive iterative algorithms.

A simple alternative is an iterative procedure which performs PCA more directly. It alternates between imputing the missing values in **Y** and applying standard PCA to the infilled (complete) data matrix $\mathbf{Y}_c$ (e.g., Jolliffe, 2002). Initially, the missing values can be replaced, for example, by the row-wise means of **Y**. The covariance matrix of the complete data $\mathbf{Y}_c$ can be estimated without the problems mentioned above, and **W** can be computed using its eigen-decomposition. The bias term **m** can be updated as the row-wise mean of $\mathbf{Y}_c$. Next, the principal components **X** are calculated:

$$\mathbf{X} = \mathbf{W}^{\mathrm{T}}(\mathbf{Y}_c - \mathbf{M}),$$

the reconstructions can be used as a better estimate for the missing values:

$$\mathbf{Y}_c = \begin{cases} \mathbf{Y} & \text{for observed values} \\ \mathbf{W}\mathbf{X} + \mathbf{M} & \text{for missing values} \end{cases}$$

and PCA can be applied again to the updated data matrix $\mathbf{Y}_c$. This process can be iterated until convergence. We will further refer to this approach as the *imputation algorithm*. In Appendix A, we show that the imputation algorithm also minimizes the cost function (4) and that it implements the EM steps for a simple probabilistic model. This approach requires the use of the complete data matrix, and therefore it is computationally very expensive for large-scale problems.

Although the imputation algorithm belongs to the class of least-squares PCA algorithms (i.e., it does not use explicit regularization), it still provides some remedy against overfitting. Performing SVD of a complete data matrix is equivalent to minimizing a cost function which is a sum of two terms: the reconstruction error for the observed values (which is the true minimized error) and the reconstruction error for the infilled missing data (which is a penalty term forcing the new reconstructions of missing data to be close to the infilled values). Initialization of the reconstructions

with the row-wise means of the data matrix introduces a bias in favor of the most "conservative" solutions. In practice, there can be infinitely many solutions that provide the same reconstruction error for the observed part but that have different reconstructions for the missing part. In such cases, the effect of the initialization can last even if the algorithm is iterated indefinitely.

Our experiments show that the imputation algorithm also has resistance against overlearning. By overlearning we mean situations when the training error decreases but the generalization ability of the model gets worse. Badly overfitted solutions correspond to regions in the parameter space where a small decrease of the training error can cause a large increase of the test error. However, the penalty term (which keeps the reconstructions of missing data close to the infilled values) makes the steps shorter and learning can practically stop once the algorithm enters regions where the training error changes very little. Thus, this regularization effect is due to early stopping.

3. Probabilistic Models for PCA

In this section, we presents probabilistic models for PCA which provide a good foundation for handling missing values.

3.1 Probabilistic PCA

The probabilistic formulation of PCA offers a number of benefits, including well-founded regularization, model comparison, interpretation of results, and extendability. *Probabilistic PCA* (Tipping and Bishop, 1999) explicitly includes the noise term in a generative model

$$\mathbf{y}_j = \mathbf{W}\mathbf{x}_j + \mathbf{m} + \varepsilon_j \,. \tag{12}$$

Both the principal components $\mathbf{x}_j$ and the noise ε_j are assumed normally distributed:

$$p(\mathbf{x}_j) = \mathcal{N}(\mathbf{x}_j; 0, \mathbf{I}) \,, \tag{13}$$

$$p(\varepsilon_j) = \mathcal{N}(\varepsilon_j; 0, v_y\mathbf{I}) \,, \tag{14}$$

where $\mathcal{N}(\mathbf{x}; \mu, \Sigma)$ denotes the normal probability density function (pdf) over variable $\mathbf{x}$ with mean μ and covariance Σ. The parameters of the model include $\mathbf{W}$, $\mathbf{m}$ and v_y. The model can be identified by finding the maximum likelihood (ML) estimate for the model parameters using the EM algorithm (Dempster et al., 1977).

Complete data. The E-step estimates the conditional distribution of the hidden variables $\mathbf{X}$ given the data and the current values of the model parameters:

$$p(\mathbf{X}|\mathbf{Y}, \mathbf{W}, v_y) = \prod_{j=1}^{n} \mathcal{N}\left(\mathbf{x}_j; \overline{\mathbf{x}}_j, \Sigma_{\mathbf{x}_j}\right) \,,$$

where the covariance matrix $\Sigma_{\mathbf{x}_j}$ is same for all $\mathbf{x}_j$:

$$\Sigma_{\mathbf{x}_j} = \Sigma_{\mathbf{x}} = v_y\left(v_y\mathbf{I} + \mathbf{W}^{\mathrm{T}}\mathbf{W}\right)^{-1}, \quad j = 1, \ldots n,$$

and the means $\overline{\mathbf{x}}_j$ can be summarized in the columns of matrix

$$\overline{\mathbf{X}} = \frac{1}{v_y}\Sigma_{\mathbf{x}}\mathbf{W}^{\mathrm{T}}\mathbf{Y} \,. \tag{15}$$

The M-step re-estimates the model parameters as

$$\mathbf{W} = \mathbf{Y}\overline{\mathbf{X}}^{\mathrm{T}}(\overline{\mathbf{X}\mathbf{X}}^{\mathrm{T}} + n\Sigma_{\mathbf{x}})^{-1}, \tag{16}$$
$$v_y = \frac{1}{nd}\sum_{i=1}^{d}\sum_{j=1}^{n}\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}}\overline{\mathbf{x}}_j\right)^2 + \frac{1}{d}\,\mathrm{tr}(\mathbf{W}\Sigma_{\mathbf{x}}\mathbf{W}^{\mathrm{T}})\,.$$

Note that v_y is updated as the mean-square error plus the extra term which accounts for the uncertainty in $\mathbf{X}$.

Tipping and Bishop (1999) showed that $\mathbf{W}$ converges to the principal subspace. In the zero-noise limit, that is for $v_y \to 0$, the iterations (15)–(16) reduce to the least squares projections in (6)–(7) (Roweis, 1998). Note also that the ML estimator for $\mathbf{m}$ is given by the row-wise mean of the data (Tipping and Bishop, 1999). Therefore, it can be computed once and removed from the data in the preprocessing step.

Incomplete data. The generalization to the case with missing values happens as follows. First we write an element-wise version of the probabilistic model in (12)–(14):

$$\begin{aligned} y_{ij} &= \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j + m_i + \varepsilon_{ij}\,, \qquad \forall ij \in O,\\ p(\mathbf{x}_j) &= \mathcal{N}(\mathbf{x}_j; 0, \mathbf{I})\,,\\ p(\varepsilon_{ij}) &= \mathcal{N}(\varepsilon_{ij}; 0, v_y)\,. \end{aligned}$$

We treat $\mathbf{w}_i$, m_i, and v_y as model parameters and $\mathbf{x}_j$ as latent variables and apply the standard EM algorithm to arrive at the following update rules:

$$\Sigma_{\mathbf{x}_j} = v_y\Big(v_y\mathbf{I} + \sum_{i\in O_j}\mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}\Big)^{-1}, \tag{17}$$
$$\overline{\mathbf{x}}_j = \frac{1}{v_y}\Sigma_{\mathbf{x}_j}\sum_{i\in O_j}\mathbf{w}_i(y_{ij} - m_i)\,, \qquad j = 1,\ldots,n,$$
$$m_i = \frac{1}{|O_i|}\sum_{j\in O_i}\left[y_{ij} - \mathbf{w}_i^{\mathrm{T}}\overline{\mathbf{x}}_j\right],$$
$$\mathbf{w}_i = \Big(\sum_{j\in O_i}\left[\overline{\mathbf{x}}_j\overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j}\right]\Big)^{-1}\sum_{j\in O_i}\overline{\mathbf{x}}_j(y_{ij} - m_i)\,, \qquad i = 1,\ldots,d, \tag{18}$$
$$v_y = \frac{1}{N}\sum_{ij\in O}\left[\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}}\overline{\mathbf{x}}_j - m_i\right)^2 + \mathbf{w}_i^{\mathrm{T}}\Sigma_{\mathbf{x}_j}\mathbf{w}_i\right].$$

There are several distinctions compared to the fully observed data: 1) The optimal $\mathbf{m}$ depends on other parameters and therefore it has to be updated in the iterative procedure. 2) The covariance $\Sigma_{\mathbf{x}_j}$ is different for each $\mathbf{x}_j$. 3) Each row of $\mathbf{W}$ is recomputed based only on those columns of $\overline{\mathbf{X}}$ which contribute to the reconstruction of the *observed* values in the corresponding row of the data matrix. The same applies to the computation of the columns of $\overline{\mathbf{X}}$. Thus, the computations required for incomplete data are generally heavier.

3.2 Examples of Overfitting with Probabilistic PCA

PPCA provides some remedy against overfitting, in contrast to the least squares approach. First, a nonzero noise level v_y regularizes badly conditioned matrices $\sum_{i\in O_j}\mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}$ in (17). Second, even

if the noise level is small, badly conditioned matrices $\sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}}$ result in large values both in the means $\bar{\mathbf{x}}_j$ and in the covariance matrices $\mathbf{\Sigma}_{\mathbf{x}_j}$. This diminishes the effect of large values in $\bar{\mathbf{x}}_j$ when $\mathbf{W}$ is re-estimated using (18).

However, PPCA can overfit, for example, if the estimated number of principal components is unreasonably large. In the example in Fig. 3, PPCA with one ($c = 1$) principal component is applied to two-dimensional ($d = 2$) data and each data vector is only partly observed (i.e., either y_{1j} or y_{2j} is known for all js). The observations are represented by triangles placed on the two axes in Fig. 3. The data points reconstructed with a trained PPCA model lie on a line, thus the model invents correlations that are not observed in the data. In fact, there are infinitely many PPCA models which are equally good and the final solution (and hence the reconstructions) depends on initialization. The situation would not change significantly if there were a few fully observed data vectors. Then, the trained model would be defined by those few samples and it would not generalize well for new data.

A similar overfitting example is presented in Fig. 4. There, three-dimensional data are described well by a model with $c = 1$ principal component. The first two rows of $\mathbf{Y}$ are fully observed while there are only two measurements in the third row (Fig. 4a). The reconstructions of the third row of $\mathbf{Y}$ are very inaccurate because the model relies only on the two available observations.

3.3 Variational Bayesian PCA (VBPCA)

A common way to cope with the overfitting problem is to penalize parameter values which correspond to more complex explanations of the data. A natural regularization in PCA is using penalization of large values in matrices $\mathbf{W}$ and $\mathbf{X}$. In the Bayesian formulation, this is equivalent to introducing a prior over the model parameters. For example, the PPCA model in (12)–(14) can be complemented with Gaussian priors over the elements of vector $\mathbf{m}$ and matrix $\mathbf{W}$:

$$p(\mathbf{m}) = \mathcal{N}(\mathbf{m}; 0, v_m \mathbf{I}), \tag{19}$$

$$p(\mathbf{W}) = \prod_{k=1}^{c} \mathcal{N}(\mathbf{W}_{:k}; 0, v_{w,k} \mathbf{I}). \tag{20}$$

Here, we use a zero mean prior for $\mathbf{m}$ for the sake of simplicity. Including a mean hyperparameter μ, that is $p(\mathbf{m}) = \prod_i \mathcal{N}(m_i; \mu, v_m)$, can be useful in practice.

The model (20) uses a shared prior for all elements in the same column of $\mathbf{W}$, parameterized with $v_{w,k}$. This is done to allow automatic selection of the right number of components needed for PCA. The hyperparameters v_m, $v_{w,k}$ can also be updated during learning (e.g., using the evidence framework or variational approximations). If the evidence of the relevance of the k-th principal component for reliable data modeling is weak, the corresponding $v_{w,k}$ should tend to zero. This is called *automatic relevance determination* (e.g., Bishop, 2006). The prior in (19)–(20) also introduces a bias towards the PCA basis within the principal subspace (Luttinen and Ilin, 2010).

Let us assume that we perform ML estimation of the hyperparameters $\xi = (v_y, v_{w,k}, v_m)$ in the probabilistic model defined in (12)–(14) and (19)–(20). This can be done using the EM-algorithm if one treats the model parameters $\theta = (\mathbf{W}, \mathbf{X}, \mathbf{m})$ as hidden random variables. Implementation of the EM algorithm would require the computation of the posterior of the hidden variables $p(\theta|\mathbf{Y}, \xi)$ on the E-step. Unfortunately, the true posterior $p(\theta|\mathbf{Y}, \xi)$ does not have an analytic form and one possible solution is to approximate it with a simpler pdf $q(\theta)$. This is justified by the variational view of the EM algorithm (Neal and Hinton, 1999).

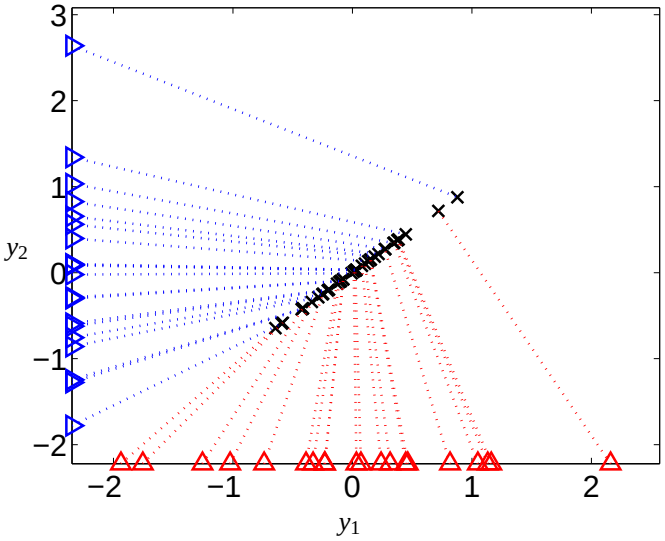


Figure 3: A simple example of overfitting with PPCA: data are two-dimensional and each sample contains only one observed value. The measurements are marked with blue and red triangles at the two axes. The reconstructions provided by a trained PPCA model are shown with crosses.

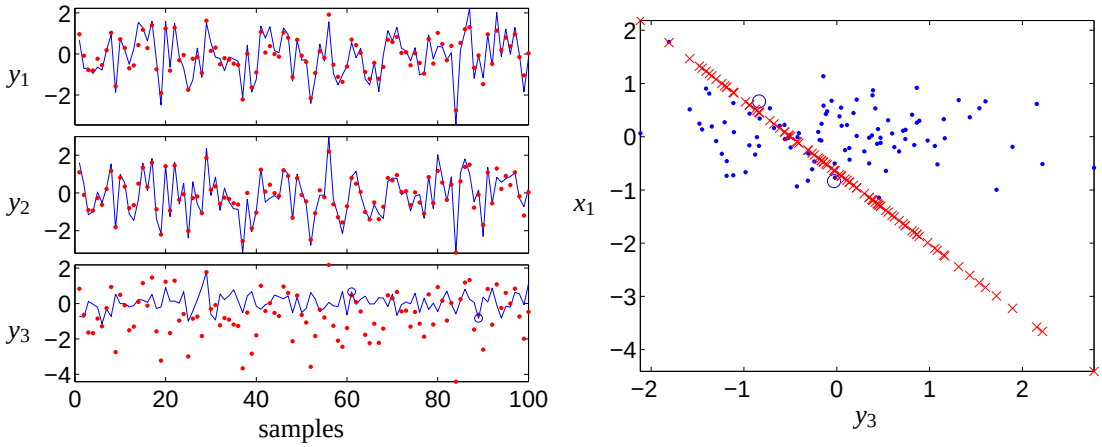


Figure 4: Example of inaccurate reconstruction with PPCA. Left: The "true" values in the three rows of $\mathbf{Y}$ are shown with blue lines. The first two rows are fully observed, the only two observations in the third row are marked with circles. The PPCA reconstruction is shown with red dots. Right: The blue dots is the scatter plot of the third row of $\mathbf{Y}$ (x-axis) against the principal component estimated by PPCA (y-axis). The dots corresponding to the two observations are marked with circles. The red crosses is the reconstruction of the third row (x-axis) against the estimated principal component (y-axis). The estimated correlation are due to the two fully observed data vectors.

Using the variational approach (Wallace, 1990; Hinton and van Camp, 1993), the E-step is modified to update the approximation $q(\theta)$ so as to minimize the cost function

$$C(q(\theta),\xi) = \int q(\theta)\log\frac{q(\theta)}{p(\mathbf{Y},\theta|\xi)}d\theta = \int q(\theta)\log\frac{q(\theta)}{p(\theta|\mathbf{Y},\xi)}d\theta - \log p(\mathbf{Y}|\xi)\,. \tag{21}$$

On the M-step, the approximation $q(\theta)$ is used as it was the actual posterior $p(\theta|\mathbf{Y},\xi)$ in order to increase likelihood $p(\mathbf{Y}|\xi)$. This can be seen as minimization of (21) w.r.t. ξ (Neal and Hinton, 1999).

The first term in (21) is the Kullback-Leibler divergence between the true posterior and its approximation. Since it is always non-negative, the cost function provides a lower bound of the log-likelihood:

$$\log p(\mathbf{Y}|\xi) \geq -C(q(\theta),\xi)\,. \tag{22}$$

This property can be used to compare solutions corresponding to different local minima of (21). More details on variational methods can be found, for example, in the book by Bishop (2006).

The variational Bayesian (VB) approach to PCA can be implemented by minimization of (21) w.r.t. the approximating pdf $q(\theta)$ and ξ.[1] The complexity of the cost function (21) depends on the form of the approximating pdf $q(\theta)$. A computationally convenient form for the PCA model is

$$q(\theta) = \prod_{i=1}^{d} q(m_i)\prod_{i=1}^{d} q(\mathbf{w}_i)\prod_{j=1}^{n} q(\mathbf{x}_j)\,. \tag{23}$$

Then, the cost function can be minimized alternately w.r.t. one factor $q(\theta_i)$ in (23) while keeping the other ones fixed. Because we use conjugate priors in (19)–(20), it is possible to find *optimal* pdfs $q(\theta_i)$ on each step: The optimal $q(m_i)$, $q(\mathbf{w}_i)$ and $q(\mathbf{x}_j)$ are Gaussian and their update boils down to re-estimation of the corresponding means and covariance matrices.

In Appendix C, we present the update rules for the resulting algorithm, which we call VBPCA. There, we assume incomplete data sets, the update rules for the fully observed data can be found in the paper by Bishop (1999). Note the resemblance of the learning rules to the EM algorithm applied to PPCA. The mean parameters $\overline{\mathbf{W}}$, $\overline{\mathbf{X}}$, $\overline{\mathbf{m}}$ of the approximating pdfs can be used as estimates of the corresponding model parameters. The covariance matrices $\Sigma_{\mathbf{w}_i}$, $\Sigma_{\mathbf{x}_j}$ and variances $\widetilde{m}_i$ reflect the uncertainty about the corresponding quantities.

The advantages of the VB approach can be summarized in the following.

- VB learning is sensitive to posterior probability mass rather than posterior probability density, which makes it more resistant against overfitting compared to point estimation (e.g., PPCA or MAPPCA presented in Section 6.4). Typically, overfitted solutions correspond to high peaks in the posterior density while robust solutions contain much of the posterior probability mass in their neighborhood.

- The method provides information about uncertainty for the unknown quantities (in the estimated posterior covariance matrices). This can be useful to detect unreliable results similar to the one presented in Fig. 4. For example, the uncertainty of the reconstructions of missing values can be estimated as we show in Section 5.

1. In the fully Bayesian treatment, the hyperparameters ξ are also assigned priors and they are treated equally to the rest of the parameters θ. We omit the effect of the prior for the hyperparameters to simplify the equations.

- The VB cost function can be used for comparison between different solutions based on (22): A smaller cost yields a greater lower bound of the solution evidence. This is a useful feature for PCA of incomplete data as, in this problem, even simpler models can provide cost functions with multiple local minima. A greater lower bound does not guarantee a better solution, but still it seems to be a reliable measure in practice, see Figure 11 for an example.

4. Rotation in the Principal Subspace to the PCA Basis

A useful property of classical PCA is that the principal components are ordered by the amount of data variance they explain, which allows an intuitive interpretation of the results. Such a representation is quite trivial for complete data sets but the extensions to the case of incomplete data is not straightforward. In this section, we show how to find such a basis in the principal subspace estimated by different algorithms for incomplete data. We refer to this basis as the PCA basis in the principal subspace.

One can define the PCA basis for *complete data* as the solution which minimizes (2) and which satisfies the following conditions: 1) the principal components (in the rows of $\mathbf{X}$) are zero mean, mutually uncorrelated and scaled to unit variance:

$$\frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j = 0, \tag{24}$$

$$\frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j\mathbf{x}_j^{\mathrm{T}} = \frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}} = \mathbf{I} \tag{25}$$

and 2) matrix $\mathbf{W}$ has mutually orthogonal columns which are ordered according to their norms:

$$\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathrm{diag}(\mathbf{s}), \tag{26}$$

$$s_k \geq s_l\,, \quad k < l\,, \tag{27}$$

where $\mathrm{diag}(\mathbf{s})$ denotes a diagonal matrix with diagonal elements $s_k = \|\mathbf{W}_{:k}\|^2$. Note that the normalized columns of $\mathbf{W}$ and their squared norms s_k are the eigenvectors and eigenvalues of the data covariance matrix.

We propose to use the same conditions (24)–(27) to define the PCA basis in the principal subspace estimated for *incomplete data*. Note the important difference to the complete data case: The PCA basis for complete data is unique (assuming distinct eigenvalues of the data covariance matrix) and it does not depend on the dimensionality for the assumed principal subspace. This cannot be guaranteed for incomplete data case: The principal subspace estimated with the same algorithm using fewer components may differ from the leading directions of the PCA basis found in the subspace with more components.

In the following, we show how to transform a solution found with different algorithms to the PCA basis such that conditions (24)–(27) are satisfied.

4.1 Least Squares Approaches

The first step to transform a solution $\{\mathbf{W},\mathbf{m},\mathbf{X}\}$ found by a least squares algorithm (as presented in Sections 2.2-2.3) is to center the rows in $\mathbf{X}$ to zero mean:

$$\mathbf{x}_j \leftarrow \mathbf{x}_j - \mu, \tag{28}$$

$$\mathbf{m}_{\text{pca}} = \mathbf{m} + \mathbf{W}\mu, \tag{29}$$

with $\mu = \frac{1}{n}\sum_{j=1}^{n}\mathbf{x}_j$. This ensures (24). The next step is to transform linearly $\mathbf{W}$ and $\mathbf{X}$:

$$\mathbf{W}_{\text{pca}} = \mathbf{W}\mathbf{U}\mathbf{D}_x^{1/2}\mathbf{V}, \tag{30}$$

$$\mathbf{X}_{\text{pca}} = \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{X}, \tag{31}$$

where matrices $\mathbf{U}$, $\mathbf{D}_x$ are computed by eigen-decomposition

$$\frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}} = \mathbf{U}\mathbf{D}_x\mathbf{U}^{\mathrm{T}} \tag{32}$$

and $\mathbf{V}$ is calculated by eigen-decomposition

$$\mathbf{D}_x^{1/2}\mathbf{U}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{U}\mathbf{D}_x^{1/2} = \mathbf{V}\mathbf{D}_w\mathbf{V}^{\mathrm{T}}. \tag{33}$$

It is easy to show that the transformed solution satisfies (25)–(26):

$$\begin{aligned}
\frac{1}{n}\mathbf{X}_{\text{pca}}\mathbf{X}_{\text{pca}}^{\mathrm{T}} &= \frac{1}{n}\mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\mathbf{U}\mathbf{D}_x^{-1/2}\mathbf{V} \\
&= \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{D}_x\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{D}_x^{-1/2}\mathbf{V} = \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{D}_x\mathbf{D}_x^{-1/2}\mathbf{V} = \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}, \\
\mathbf{W}_{\text{pca}}^{\mathrm{T}}\mathbf{W}_{\text{pca}} &= \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{1/2}\mathbf{U}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{W}\mathbf{U}\mathbf{D}_x^{1/2}\mathbf{V} = \mathbf{V}^{\mathrm{T}}\mathbf{V}\mathbf{D}_w\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{D}_w.
\end{aligned}$$

4.2 Probabilistic PCA

We show in Appendix B that the following conditions hold at the convergence of PPCA:

$$\frac{1}{n}\sum_{j=1}^{n}\overline{\mathbf{x}}_j = 0, \tag{34}$$

$$\Sigma_* = \frac{1}{n}\sum_{j=1}^{n}\left[\overline{\mathbf{x}}_j\overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j}\right] = \mathbf{I}, \tag{35}$$

which can be seen as the analogue of (24)–(25). Note that Σ_* becomes the sample covariance matrix of the principal components in the noiseless limit, when $v_y \to 0$. The extra requirements (26)–(27) can be used to define the (practically) unique PCA basis for a PPCA solution. They resolve the rotational ambiguity caused by the fact that any orthogonal transformation of the principal subspace does not change Σ_* in (35).

One can easily perform a transformation of a PPCA solution such that (34)–(35) and (26)–(27) hold. This is done by first normalizing $\overline{\mathbf{x}}_j$ to zero mean and updating the bias term similarly to (28)–(29). Then, a rotation of the subspace is performed similarly to (30)–(33) with the exception that

$\frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}}$ is replaced with Σ_*. Finally, one can rotate the posterior covariance matrices of the estimated principal components:

$$\Sigma_{\mathbf{x}_j,\mathrm{pca}} = \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{-1/2}\mathbf{U}^{\mathrm{T}}\Sigma_{\mathbf{x}_j}\mathbf{U}\mathbf{D}_x^{-1/2}\mathbf{V}\,, \tag{36}$$

which follows from (31).

Although the optimal PPCA solution satisfy at least conditions (34)–(35), convergence of the learning algorithm to the optimal solution can be very slow. Therefore, performing the described transformations during learning can speed up learning. In the experiments, we use the transformations after each iteration.

4.3 VBPCA

It can be shown (Luttinen and Ilin, 2010) that VBPCA described in Section 3.3 converges to the solution which satisfies (34)–(35) and the condition analogous to (26):

$$\sum_{i=1}^{d}\left[\overline{\mathbf{w}_i}\overline{\mathbf{w}_i}^{\mathrm{T}} + \Sigma_{\mathbf{w}_i}\right] = \mathrm{diag}(\mathbf{s})\,, \tag{37}$$

where $\overline{\mathbf{w}_i}$, $\Sigma_{\mathbf{w}_i}$ are the posterior means and covariance matrices for the rows of matrix $\mathbf{W}$ (see Appendix C for notation details). Thus, one can simply use the ordering requirement (27) to define a meaningful basis for a VBPCA solution. Performing a transformation that ensures (34)–(35) and (37) during learning can speed up convergence (Luttinen and Ilin, 2010). The transformation can be performed similarly to (30)–(33) with the exception that $\mathbf{X}\mathbf{X}^{\mathrm{T}}$ is replaced with $\sum_{j=1}^{n}\left[\overline{\mathbf{x}}_j\overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j}\right]$ in (32) and $\mathbf{W}^{\mathrm{T}}\mathbf{W}$ is replaced with $\sum_{i=1}^{d}\left[\overline{\mathbf{w}_i}\overline{\mathbf{w}_i}^{\mathrm{T}} + \Sigma_{\mathbf{w}_i}\right]$ in (33). Finally, one can rotate the posterior covariance matrices using (36) and

$$\Sigma_{\mathbf{w}_i,\mathrm{pca}} = \mathbf{V}^{\mathrm{T}}\mathbf{D}_x^{1/2}\mathbf{U}^{\mathrm{T}}\Sigma_{\mathbf{w}_i}\mathbf{U}\mathbf{D}_x^{1/2}\mathbf{V}\,.$$

In the experiments, we use the transformations after each iteration.

5. Reconstruction of Missing Values and Selection of Model Rank

PCA is a popular approach to the problem of missing value reconstruction. The estimated principal components capture the correlations between different variables, which allows for reconstruction of missing values from the observed ones. The models discussed in this article compute reconstructions using (5), in which $\mathbf{w}_i$, m_i, $\mathbf{x}_j$ are replaced with the respective posterior means $\overline{\mathbf{w}}_i$, $\overline{m}_i$, $\overline{\mathbf{x}}_j$ when applicable. Additionally, one can estimate the uncertainty of the predictions for PPCA and VBPCA by computing its variance

$$\widetilde{y}_{ij} = \widetilde{m}_i + \overline{\mathbf{w}}_i^{\mathrm{T}}\Sigma_{\mathbf{x}_j}\overline{\mathbf{w}}_i + \overline{\mathbf{x}}_j^{\mathrm{T}}\Sigma_{\mathbf{w}_i}\overline{\mathbf{x}}_j + \mathrm{tr}(\Sigma_{\mathbf{x}_j}\Sigma_{\mathbf{w}_i})\,,$$

where $\widetilde{m}_i$, $\Sigma_{\mathbf{x}_j}$, $\Sigma_{\mathbf{w}_i}$ represent the posterior uncertainties of the respective parameters and $\Sigma_{\mathbf{w}_i}$ is zero for PPCA. This is a useful feature of the probabilistic methods compared to the least-squares approach.

The quality of reconstruction depends on the number of estimated principal components and therefore selection of the model rank is an important problem. In this section, we briefly outline possible solutions to this problem leaving out the detailed discussion.

The most straightforward way to define the rank of the model is to use cross-validation. The data set is divided into training and validation parts, models with different ranks are fitted to the training set and the the rank is selected based on the performance on the held-out validation set.

Another way is to use probabilistic methods for model selection. For example, Minka (2001) developed a means to detect the dimensionality of the principal subspace for *complete* data using Laplace's method and Bayesian information criterion (BIC) applied to the PPCA model. He also discussed other alternatives but his approach showed the best performance. However, the extension of Laplace's method to the case of *incomplete* data is not straightforward. An additional difficulty is the possibility that the posterior distribution over the model parameters has multiple modes corresponding to different principal subspaces. VBPCA can select the optimal number of components automatically by setting some columns of $\mathbf{W}$ to zero. However, the VB cost function may have many local minima which correspond to different model ranks and exploring different local solutions can be a tedious procedure. Hoff (2008) estimated the dimensionality of the principal subspace in the complete data using a sampling procedure.

6. Speeding Up Learning For Sparse High-Dimensional Data Sets

Obtaining a reasonably good solution in appropriate time is a very important issue for high-dimensional problems in which learning may take several days, as in the example considered in Section 7.4. The algorithms presented in the previous sections scale differently to problems with large dimensions and large degree of sparsity of observed values.

The computational complexities of different algorithms are summarized in Tables 2 and 3. For example, the alternating optimization algorithm for PPCA requires computation and inversion of matrices $\sum_{j\in O_i}\left[\mathbf{x}_j\mathbf{x}_j^{\mathrm{T}}+\boldsymbol{\Sigma}_{\mathbf{x}_j}\right]$ and $\left(v_y\mathbf{I}+\sum_{i\in O_j}\mathbf{w}_i\mathbf{w}_i^{\mathrm{T}}\right)$, which are generally unique for each row of $\mathbf{W}$ and each column of $\mathbf{X}$, respectively. The corresponding computational complexity is $O(Nc^2+nc^3)$ per iteration,[2] which is quite a bit heavier than $O(ndc)$ per iteration (Roweis, 1998) required for complete data.

The computational complexity $O(Nc+nc)$ of the gradient method is small compared to previously reported results. For instance, Srebro and Jaakkola (2003) reported the complexity $O(ndc^2)$ for a more general model which was a huge improvement over the complexity $O(n^3c^3)$ of the algorithms proposed even earlier. Using the numbers from the Netflix problem considered in Section 7.4 ($N\approx 10^8$, $n\approx 480189$, $d=17770$, $c=30$), we get rough estimates of the differences in practice: Imputation algorithm $nd^2\approx 10^{14}$, (Srebro and Jaakkola, 2003) $ndc^2\approx 10^{12}$, alternating $\mathbf{W}$–$\mathbf{X}$ $Nc^2+nc^3\approx 10^{11}$, and gradient $Nc+nc\approx 10^{9.5}$.

Once the model has been learned and fixed, doing inference on a new sample is fast. Using the alternating algorithm, it requires only one E-step with $O(N_0c^2+c^3)$, where $N_0\leq d$ is the number of observed values in the new sample, while imputation and gradient algorithms require a number of iterations with $O(dc)$ and $O(N_0c)$ each.

Efficient memory usage is an important issue for sparse data sets. It is preferable that the amount of used memory scales linearly with the number of observed values regardless of the data matrix dimensionalities. In Table 2, we provide the number of parameters estimated by different models, which gives the lower bound for the amount of the required memory. Only the imputation algorithm

2. The computational complexity of the alternating $\mathbf{W}$–$\mathbf{X}$ scheme can be reduced if the matrices requiring inversion are computed once for all rows (columns) of $\mathbf{Y}$ that have observed values at exactly the same columns (rows respectively).

	Section	Number of parameters	Imputation with SVD	Alternating **W**–**X**	Gradient descent
LS	2	$(d+n)c$	✓	✓	✓
MAPPCA	6.4	$(d+n)c$		✓	✓
PPCA	3.1	$(d+n)c+nc^2$		✓	
VBPCA	3.3	$(d+n)(c+c^2)$		✓	
PPCAd	6.3	$(d+2n)c$			✓
VBPCAd	6.3	$(2d+2n)c$			✓

Table 2: Memory requirements and applicability of optimization methods to different models. We mark only the optimization schemes which can easily be implemented using the formulas provided in this paper. Some of the methods mentioned here will be presented in the upcoming sections.

Imputation with SVD	Alternating **W**–**X**	Gradient descent
$O(nd^2)$	$O(Nc^2+nc^3)$	$O(Nc+nc)$

Table 3: Summary of computational complexities (per iteration) of different optimization methods, assuming naïve computation of products and inverses of matrices and ignoring the computation of SVD in the imputation algorithm.

requires memory for the complete data matrix, and even that requirement can be diminished at a cost of greater time complexity.

6.1 Gradient-Based Learning

The gradient-based optimization scheme, extending the Oja learning algorithm to regularized PCA models, can be very efficient for large-scale problems and sparse data sets. The computational complexity of one iteration scales very well with dimensionalities, which can lead to faster learning in practice. The gradient-based approach can also be advantageous compared to the alternating optimization scheme as the latter discards the joint effect of parameters **W** and **X** on the changes in the cost function. This results in slow convergence without proper speed-up procedures (e.g., Honkela et al., 2003).

We also propose to use a speed-up to the gradient descent algorithm. In Newton's method for optimization, the gradient is multiplied by the inverse of the Hessian matrix. Newton's method is known to be fast-converging, but using the full Hessian is computationally costly in high-dimensional problems ($d \gg 1$). We propose a simplified approach which uses only the diagonal part of the Hessian matrix and includes a control parameter α that allows the learning algorithm to vary from the standard gradient descent ($\alpha = 0$) to the diagonal Newton's method ($\alpha = 1$). The final learning rules then take the form

$$\theta_i \leftarrow \theta_i - \gamma \left(\frac{\partial^2 C}{\partial \theta_i^2} \right)^{-\alpha} \frac{\partial C}{\partial \theta_i}.$$

For example, the cost function (4) has the second-order derivatives

$$\frac{\partial^2 C}{\partial w_{ik}^2} = \sum_{j \in O_i} x_{kj}^2 , \qquad \frac{\partial^2 C}{\partial x_{kj}^2} = \sum_{i \in O_j} w_{ik}^2 ,$$

which can be computed efficiently for sparse data. We showed the efficiency of the proposed optimization scheme in the application to large-scale PCA problems in our conference papers (Raiko et al., 2008, 2007a).

6.2 Online Learning

In cases where the number of data samples is very large, $n \gg d$, it is wasteful to go through the whole data set before updating $\mathbf{W}$ (and $\mathbf{m}$). Using online learning (Oja, 1983), one sample j is processed at a time as follows: The principal components $\mathbf{x}_j$ are inferred using (8) (or the E-step in general), and $\mathbf{W}$ is updated using the gradient approach. When sweeping through the data, updates concerning the latter samples benefit from using the $\mathbf{W}$ that has already been updated using the earlier ones. This reduces the required number of iterations. Downsides of this approach include that determining the step size, or using speed-ups such as the one in Section 6.1 or the conjugate gradient, becomes difficult. Also, for enabling parallel implementation, one should process at least a small batch of samples at a time before updating $\mathbf{W}$ (e.g., Salakhutdinov et al., 2007). Online learning was not considered in this paper.

6.3 Factorial Variational Approximations

The overfitting problem can be more severe for high-dimensional sparse data sets. The situations discussed in Sections 2.4 and 3.2 are more probable when data points are sparsely distributed in high dimensions. Thus, methods penalizing large values of model parameters and taking into account their posterior uncertainty are especially relevant here.

The PPCA and VBPCA approaches, which take into account the posterior uncertainty at least in $\mathbf{X}$, are hardly applicable to large-scale problems. First, the computational burden of one iteration of the alternating $\mathbf{W}$–$\mathbf{X}$ scheme is very large (see Table 3) and application of the gradient-based optimization is cumbersome because of the need to compute $c \times c$ covariance matrices. Second, the required memory is at least nc^2 elements for storing posterior correlations only in $\mathbf{X}$. This becomes infeasible for many large-scale problems even for a decent number of principal components.

A possible solution is to take into account only some posterior correlations and to use variational techniques for learning. For example, the posterior approximation $q(\theta)$ in VBPCA can be made fully factorial leading to

$$q(\theta) = \prod_{i=1}^{d} q(m_i) \prod_{i=1}^{d} \prod_{k=1}^{c} q(w_{ik}) \prod_{c=1}^{k} \prod_{j=1}^{n} q(x_{kj}) . \tag{38}$$

instead of (23). Such posterior approximation was used, for example, by Raiko et al. (2007b) for PCA in the presence of missing values. The implementation proposed there was based on the imputation algorithm and thus not easily scalable to high-dimensional sparse problems.

The fully factorial approximation (38) reduces significantly the number of variational parameters. They now include the mean parameters $\overline{m}_i$, $\overline{w}_{ik}$ and $\overline{x}_{kj}$ and the variance parameters $\widetilde{m}_i$, $\widetilde{w}_{ik}$, $\widetilde{x}_{kj}$ in addition to hyperparameters v_y, $v_{w,k}$, v_m, which can be point-estimated. The corresponding

cost function is given in Appendix D. It can be minimized in different ways, for example, by using the gradient-based optimization scheme explained in Section 6.1. The complete algorithm works by alternating four update steps: $\{\widetilde{w}_{ik}, \forall i,k\}$, $\{\widetilde{x}_{kj}, \forall k,j\}$, $\{\overline{w}_{ik}, \overline{x}_{kj}, \forall i,k,j\}$, and $\{v_y, v_{w,k}, v_m, \forall k\}$. The required derivatives are reported in Appendix D. We will further refer to this algorithm as *VBPCAd*.

The idea of fully factorial approximation can also be used for reducing the complexity of PPCA. The posterior of the hidden states $p(\mathbf{X}|\mathbf{Y},\mathbf{W},\mathbf{m},v_y)$ can be approximated to be fully factorial on the E-step. The approximating pdf (38) can be fitted to the true posterior by minimizing the cost function

$$C_{\mathrm{PPCA}} = \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{Y},\mathbf{X}|\mathbf{W},\mathbf{m},v_y)} d\mathbf{X},$$

which is motivated by the variational view of the EM algorithm. The resulting update rules resemble the ones of VBPCAd with the exceptions outlined in Fig. 5. We refer to this approach as *PPCAd*.

Note that the approximation used in PPCAd does not restrict the generality of the PPCA model when it is applied to *complete* data. The variational approximation introduces a bias in favor of solutions for which the form of the posterior approximation agrees with the form of the true posterior (e.g., Ilin and Valpola, 2005). The posterior covariance matrix $\mathbf{\Sigma}_{\mathbf{x}_j}$, which is the same for each j in the fully observed case, is diagonal if and only if the columns of $\mathbf{W}$ are mutually orthogonal. Therefore, restricting the posterior covariance to be diagonal guarantees that (26) holds at convergence. Requirements (24)–(25) are fulfilled because of the assumed prior model for $\mathbf{X}$. Thus, PPCAd for complete data should converge to the PCA basis. PPCAd applied to *incomplete* data is biased in favor of solutions in which the true posterior covariance matrices are closer to being diagonal. Note also that the idea of speeding up learning by using transformations as described in Section 4.3 for VBPCA, cannot be used for VBPCAd, because the transformations would break up the diagonality of the posterior covariance matrices.

There is an interesting connection between the speed-up proposed in Section 6.1 and the fully diagonal posterior approximation. The second order derivatives needed for the speed-up coincide with the inverse of the posterior variance of each parameter, and thus their computation is practically free.

6.4 Ignoring Posterior Uncertainty (MAPPCA)

A further simplification is to use maximum a posteriori (MAP) estimation for the model parameters. The minimized cost function is then minus log-posterior of the parameters. Assuming uniform prior for hyperparameters v_y, $v_{w,k}$, v_m, it is proportional to

$$C_{\mathrm{MAP}} = \frac{1}{v_y} \sum_{ij \in O} (y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i)^2 + N \log 2\pi v_y + \frac{1}{v_m} \sum_{i=1}^{d} m_i^2 + d \log 2\pi v_m \\ + \sum_{k=1}^{c} \left[\frac{1}{v_{w,k}} \sum_{i=1}^{d} w_{ik}^2 + d \log 2\pi v_{w,k} + \sum_{j=1}^{n} x_{kj}^2 + n \log 2\pi \right]. \quad (39)$$

and it should be minimized w.r.t. the unknown quantities $\mathbf{W}$, $\mathbf{X}$, v_y, $v_{w,k}$, v_m. We refer to this approach as MAPPCA.

Minimization of (39) can be done by any suitable optimization procedure. Derivatives required for gradient-based optimization are given in Appendix E. However, some difficulties should be

avoided. First, using an improper prior for hyperparameters leads to a situation where the posterior pdf is infinitely large when $v_{w,k} \to 0$, $\sum_{i=1}^{d} w_{ik}^2 \to 0$. This can be overcome by penalizing too small values of the variance hyperparameters. For example, a hyperparameter v_z, which is a variance parameter of the prior for a set of zero-mean variables $\{z_i, i = 1, \ldots, M\}$, can be updated as $v_z = \frac{2\beta + \sum_{i=1}^{M} \langle z_i^2 \rangle}{2\alpha + M}$, where $\langle z_i^2 \rangle$ denotes the expectation of z_i^2 over the posterior, and α and β are some small values (we used $\alpha = 10^{-3}$ and $\beta = 10^{-3}$). This update rule corresponds to broad priors restricting v_z to be positive and it guarantees that $v_z \geq \frac{\beta}{\alpha + M/2}$. The maximum likelihood (ML) estimate $v_z = \frac{1}{M} \sum_{i=1}^{M} \langle z_i^2 \rangle$ is achieved when $\alpha \to 0$, $\beta \to 0$.[3]

Another issue is the non-identifiability of scaling between $\mathbf{W}$ and $\mathbf{X}$. A practical way to avoid this problem is to fix the scale by normalizing the rows of $\mathbf{X}$ to zero mean and unit variance (after each update) and compensating by scaling the columns of $\mathbf{W}$ accordingly. This guarantees the fulfilment of (24)–(25), which are the conditions justified by the prior model.

6.5 Summary of the PCA Variants

The differences between the six described variants of the PCA models are summarized in Table 4 and Fig. 5. The approaches differ on the posterior models and on the use of prior for $\mathbf{W}$ and $\mathbf{m}$. All the variants can be derived as special cases of VBPCA by making simplifying assumptions, which is shown in Fig. 5.

7. Experiments

We study the properties of the discussed algorithms first using artificial data and then presenting a case study using the Netflix competition data (Netflix, 2007). In the experiments with artificial data, we generated data matrices $\mathbf{Y}$ according to model (12)–(14) with fixed dimensionalities d, c and n. The mixing matrix $\mathbf{W}$ was generated by taking a random orthogonal matrix and scaling its columns by $1, 2, \ldots, c$. The bias term $\mathbf{m}$ was selected randomly from a Gaussian distribution with variance 10. For one of the data sets (10-5-mlp), we nonlinearly transformed the resulting data. The observed values were selected randomly in $\mathbf{Y}$ and the rest of the data matrix was used to compute the test reconstruction error.

We used four types of data sets in the experiments:

- *Data Set 10-5-g*: The data set is for the case where $d \ll n$ and Gaussian components. We generated $n = 1000$ samples from a 10-dimensional Gaussian distribution with $c = 5$ principal directions. The standard deviation of the noise in (14) was $\sqrt{v_y} = 0.5$.

- *Data Set 10-5-mlp*: The data lie within a nonlinear manifold and $d \ll n$. The data were generated by a random nonlinear transformation of five-dimensional Gaussian data to the 10-dimensional space and adding relatively small Gaussian noise. The transformation was performed by randomly initialized multi-layer perceptron networks. The number of samples was $n = 1000$.

- *Data Set 10-4-mog*: The data set is a linear combination (12) of non-Gaussian $\mathbf{x}_j$ and $d \ll n$. We used $c = 4$ independently distributed non-Gaussian components (elements of $\mathbf{x}_j$). The

3. In the formulas reported in the article, we always present the ML update rules, although more robust update rules were used in practice.

	prior for **W**, **m**	posterior for **W**, **m**	posterior for **X**
LS	uniform	point	point
PPCA	uniform	point	full
PPCAd	uniform	point	diagonal
VBPCA	yes	full	full
VBPCAd	yes	diagonal	diagonal
MAPPCA	yes	point	point

Table 4: Variants of probabilistic approaches to PCA.

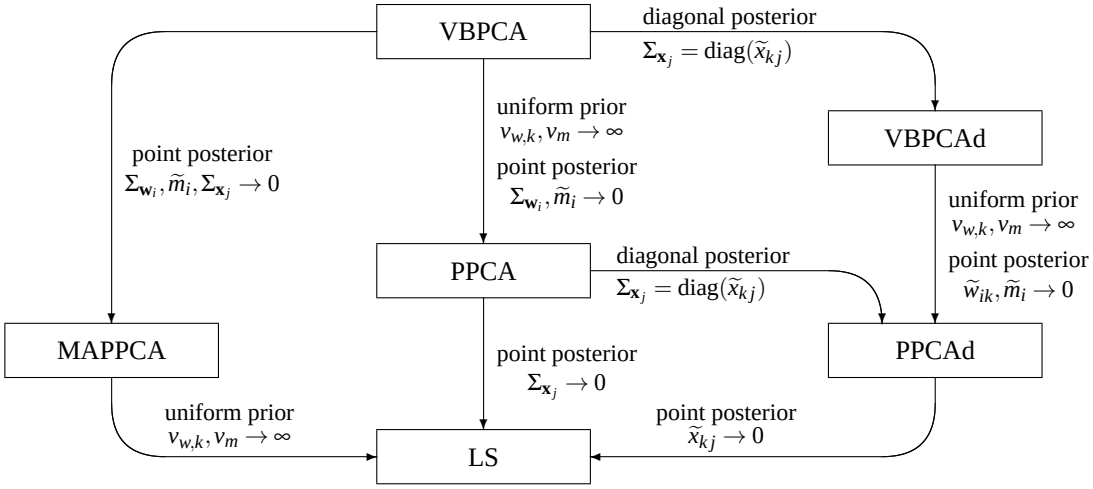


Figure 5: Relationships between the PCA variants are easily described by deriving each of them as a special case of VBPCA. The corresponding update rules can also be derived from the VBPCA update rules using the given restrictions.

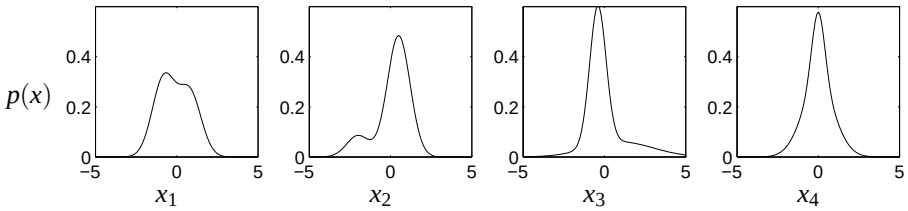


Figure 6: The distributions of the four components used to generate the 10-4-mog data set.

Data Set	d	n	c	Gaussianity	Square root of variances
10-5-g	10	1000	5	yes	[5 4 3 2 1 0.5 ... 0.5]
10-5-mlp	10	1000	5	no	varying by nonlinear transformation
10-4-mog	10	1000	4	no	[4 3 2 1 0.5 ... 0.5]
100-10-g	100	100	10	yes	[10 9 ... 1 0.1 ... 0.1]

Table 5: Characteristics of four types of data sets used in artificial experiments.

distributions of the four components are presented in Fig. 6. The standard deviation of the noise in (14) was $\sqrt{v_y} = 0.5$. The number of samples was $n = 1000$.

- *Data Set 100-10-g*: The data set is for the case where $d \approx n$ and small noise. The data were generated from a 100-dimensional Gaussian distribution with $c = 10$ principal directions. The standard deviation of the noise in (14) was $\sqrt{v_y} = 0.5$. The number of samples was $n = 100$.

The ratio of missing values was varied for all data sets. We used data sets with relatively small dimensionalities in order to test the performance of many alternative algorithms in a reasonable time. The summary of the data sets is shown in Table 5.

Different PCA algorithms considered in this article were used to estimate the same number c of principal components that was used for generating the data (see Table 5). The algorithms were initialized randomly except for the imputation algorithm which has a standard procedure of infilling missing values with the row-wise means of $\mathbf{Y}$.

The performance of the algorithms was assessed by the speed of their convergence and the accuracy in the task of missing value reconstruction. We computed three quantities: the average training root mean square (RMS) reconstruction error $\sqrt{\frac{1}{N}\sum_{ij\in O}(y_{ij} - \hat{y}_{ij})^2}$, the average test RMS reconstruction error and the average time to convergence. Averaging was done across 30 different data sets of the same type and the same ratio of missing values. The algorithms were run until the deviation of the training error was less than 0.0001 during 100 iterations. This stopping criterion was used to estimate the time to convergence.

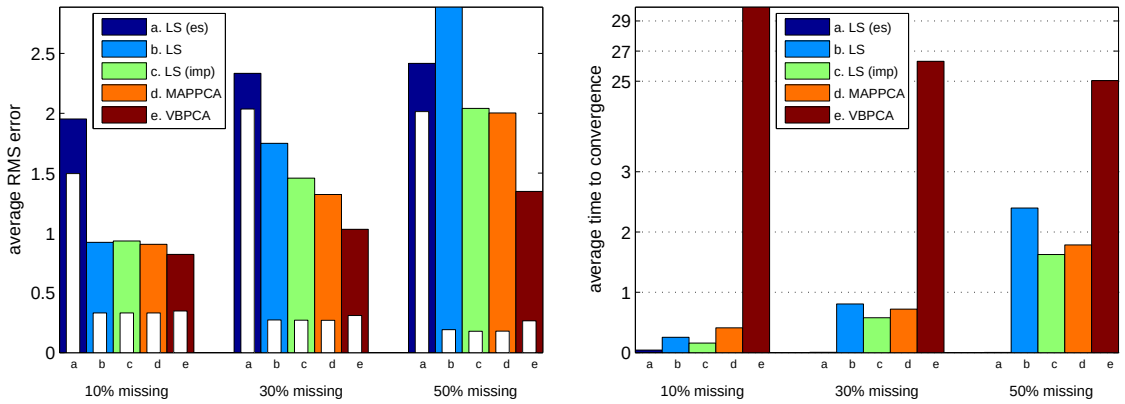
7.1 Performance of Least Squares Approaches and MAPPCA When $d \ll n$

In this section, we present the performance of three techniques:

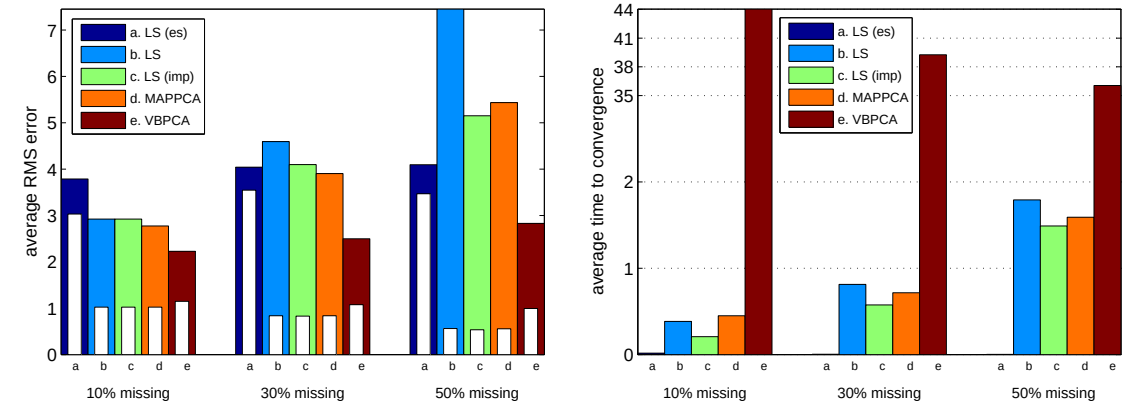
- LS: the least squares approach which optimizes the cost function (4) using the gradient-based procedure with the speed-up explained in Section 6.1
- LS (es): the same algorithm which also used early stopping: 20% of data was reserved as validation set and learning was stopped when the reconstruction error for the validation set started to increase
- LS (imp): the imputation algorithm presented in Section 2.5
- MAPPCA: method presented in Section 6.4. We used the gradient-based optimization procedure with the speed-up explained in Section 6.1.

The estimated performance of the four algorithms is presented in Fig. 7. There, we also show the results of VBPCA which turned out to be the most accurate method. We summarize the observed results in the following.

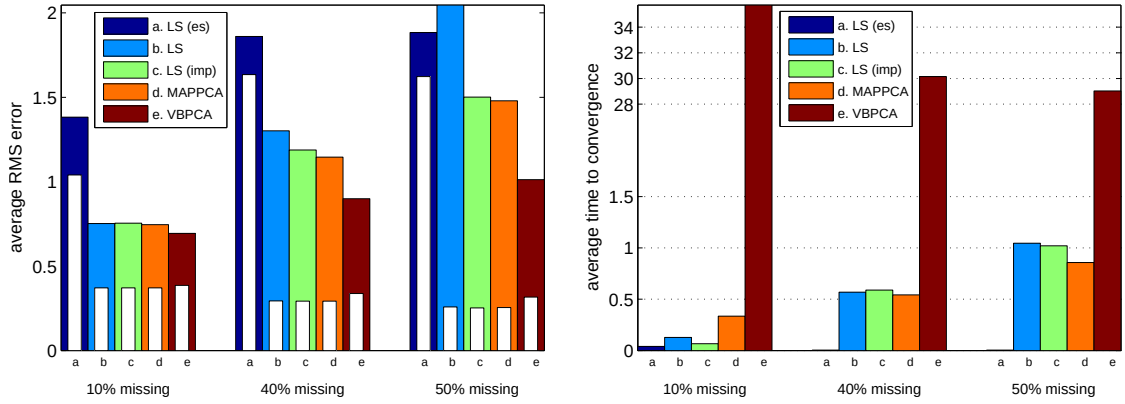
The gradient-based LS approach can be used for data sets with a relatively small amount of missing data. For such data sets, it provides reasonable accuracy, converges fast and scales well to high-dimensional data. However, its performance deteriorates fast with the increase of the ratio of missing values. The algorithm can get stuck at regions where learning proceeds very slowly and overfitting problems become very frequent. When overfitting happens, some parameter values can grow very large to explain perfectly only part of data. This results in very bad generalization and large test reconstruction errors (see discussion in Section 2.1). Thus, this method can badly overfit



(a) Data Set 10-5-g



(b) Data Set 10-5-mlp



(c) Data Set 10-4-mog

Figure 7: Left: Average test RMS errors (colored bars) and training RMS errors (white bars) obtained from 30 realizations of data sets of three types. Right: Average time to convergence, in seconds, estimated in the same way. Note the different scale on the r.h.s. plots for VBPCA.

even in relatively simple problems. We observed similar performance for the alternating optimization (see Section 2.2) of the sum-squared cost function: The alternating optimization procedure provided similar reconstruction errors but it seemed to get stuck in regions with slow convergence more often.

The simplest strategy of *early stopping* did not seem to improve the performance of the LS algorithms. On the contrary, the obtained accuracy was the worst among the considered approaches.

The imputation algorithm can also be a good choice for data sets with a relatively small amount of missing data. The algorithm converges fast, provides good accuracy and it has less problems with overfitting compared to explicit optimization of the sum-squared error. However, the complexity of the required computations grows fast with the increase of the dimensionalities, and therefore the algorithm is hardly applicable to large-scale data sets.

MAPPCA is a fast algorithm which provides reasonable accuracy for data sets with a relatively small amount of missing data. In many cases, it slightly outperforms the LS approaches in terms of the reconstruction accuracy. It is also less prone to serious overfitting problems compared to explicit optimization of (4). Again, similar performance was observed for MAPPCA optimized with the alternating optimization procedure explained in Section 2.2.

7.2 Performance of More Advanced Probabilistic Approaches When $d \ll n$

In this section, we present the performance of the more advanced probabilistic approaches:

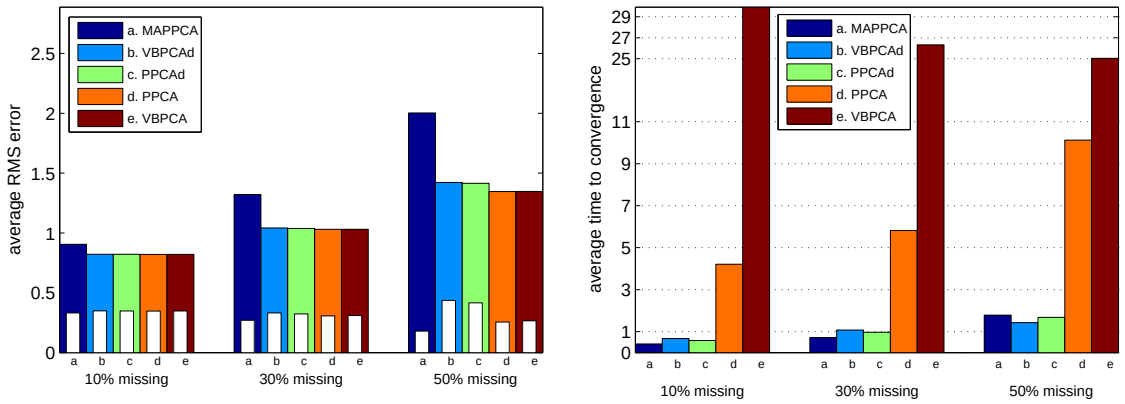
- PPCA presented in Section 3.1
- PPCAd which is PPCA with fully factorial variational approximation explained in Section 6.3
- VBPCA presented in Section 3.3
- VBPCAd which is VBPCA with fully factorial variational approximation explained in Section 6.3.

We used the gradient-based optimization procedure with the speed-up explained in Section 6.1 for training PPCAd and VBPCAd.

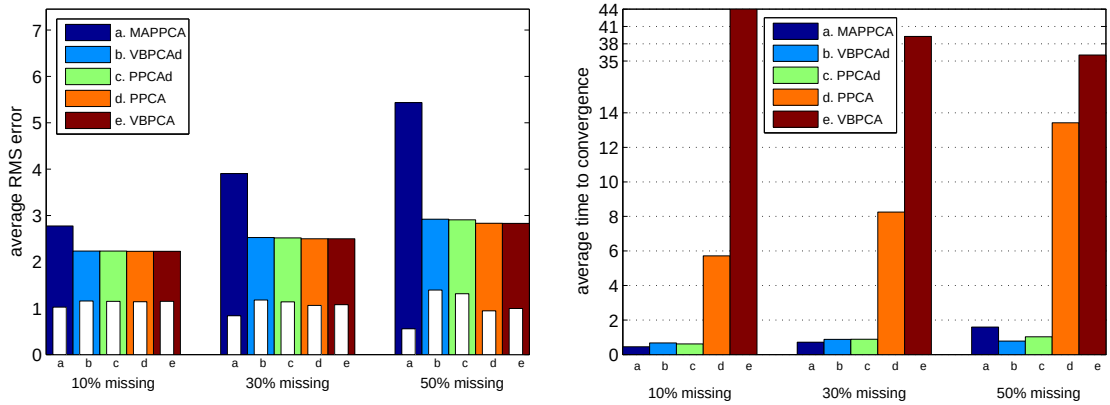
Fig. 8 presents the estimated performance of the algorithms for three data sets with $d \ll n$. We also show the results of MAPPCA for comparison. We summarize the results in the following.

Approaches using variational approximations, especially VBPCA and VBPCAd, may suffer from the underfitting problem. The algorithms may converge to a sub-optimal solution (corresponding to a local minimum of the VB cost function) in which some potentially useful principal components are not used. The algorithms seem to find such sub-optimal solutions more often for sparse data. Special care has to be taken to avoid such local minima. For example, in the presented experiments we fixed priors for $\mathbf{W}$ to be very broad at the beginning of learning and this helped improve the performance of VBPCA and VBPCAd significantly. See also the experiment showing the effect of the broad priors in Section 7.3.

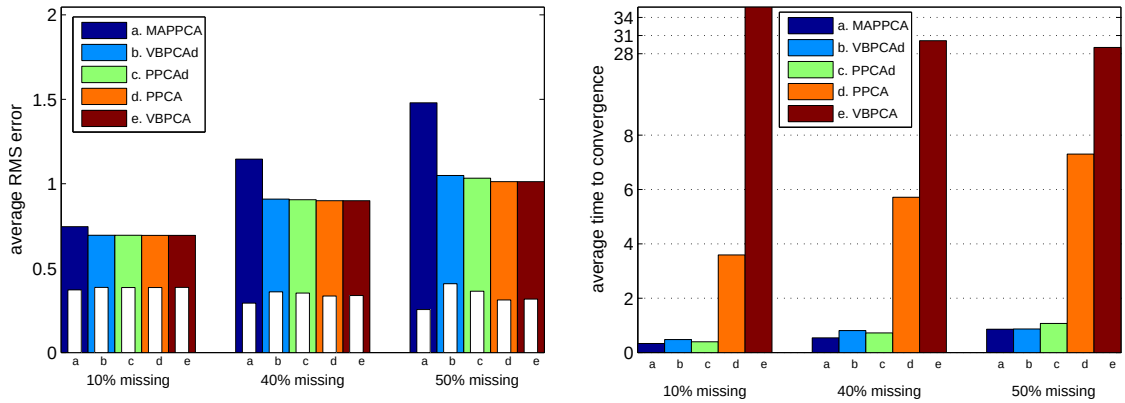
PPCAd is a very efficient algorithm for data sets with $d \ll n$. Its reconstruction accuracy is comparable to the best results (obtained with VBPCA) but the algorithm converges fast and scales well to high-dimensional problems. The difference of the accuracy of PPCAd compared to the best results of VBPCA becomes more noticeable when the ratio of missing values or the number of estimated principle components increase.



(a) Data Set 10-5-g



(b) Data Set 10-5-mlp



(c) Data Set 10-4-mog

Figure 8: Left: Average test RMS errors (colored bars) and training RMS errors (white bars) obtained from 30 realizations of data sets of three types. Right: Average time to convergence, in seconds, estimated in the same way. Note the different scale on the r.h.s. plots for VBPCA.

VBPCAd provides similar performance to PPCAd for data sets with $d \ll n$ but it is more prone to the underfitting problem mentioned earlier.

VBPCA and *PPCA* are the most accurate algorithms in these experiments. However, they are also very slow because of the required computations of the posterior covariance matrices. Thus, the methods are computationally very expensive to use in high-dimensional problems but if they are feasible, they are worth trying. The difference in the accuracy of VBPCA and PPCA is not noticeable in these experiments. This is likely due to the fact that the number n of samples was very large and the effect of priors was very small. VBPCA is also more prone to the underfitting problem compared to PPCA.

7.3 Performance in Difficult Sparse Problems

In this section, we test the performance of the considered algorithms on a more challenging data set 100-10-g. The data set is quite sparse (up to 75% of values are missing) and the number n of samples is very small compared to the data dimensionality d. Sparse data sets with $d \approx n$ can appear in some practical applications. For example, in collaborative filtering the number of users can be comparable to the number of ranked items and the number of missing values can be large. Here we used Gaussian data with $n = d = 100$, $c = 10$ principal directions and with relatively small noise.

Fig. 9 presents the performance of the considered algorithms for these data. The LS algorithms and MAPPCA do not provide good performance because of overfitting problems. The probabilistic approaches perform much better and the best results are obtained with VBPCA. The advantage of VBPCA becomes more significant for data sets with many missing values. The experiments also show a larger effect of using priors in VBPCA and VBPCAd (in comparison with PPCA and PPCAd) for data sets with a relatively small number of samples.

One of the reasons for the superior performance of the VB approach is its ability to select automatically the optimal rank of the model by cutting off useless principal components. Model selection is useful even in such artificial tests (when the number of principal components used for data generation is known) because the number of components which are useful for reliable reconstruction of missing data can turn out to be smaller than the model rank used for generating the data. Thus, the reported accuracy of the algorithms alternative to VBPCA and VBPCAd might be improved if model selection was explicitly performed for them (e.g., by cross-validation).

In Section 7.2, we pointed out that the downside of the VB model selection is the existence of multiple sub-optimal solutions corresponding to different local minima of the VB cost function. Solutions in which the effect of some components is set to zero are potentially attractive for the method. In order to avoid sub-optimal solutions, we fixed the priors for $\mathbf{W}$ to be very broad at the beginning of learning for VBPCA and VBPCAd in our experiments. We show the effect of the number of iterations with fixed broad priors on the resulting accuracy in Fig. 10.

A useful feature of the VB approach is that the value of the VB cost function at a local minimum can be used to estimate the accuracy of the corresponding model. This follows from the fact that the VB cost gives the lower bound of the likelihood, as shown in (22). In Fig. 11, we demonstrate the correlations between the accuracy of a trained VBPCA model and the corresponding value of the cost function. In that experiment, we initialized VBPCA in two different ways and applied it to 30 realizations of the 100-10-g data set with 80% of missing data. The negative result was that different initializations led to different VBPCA solutions for most of the 30 realizations. The positive result

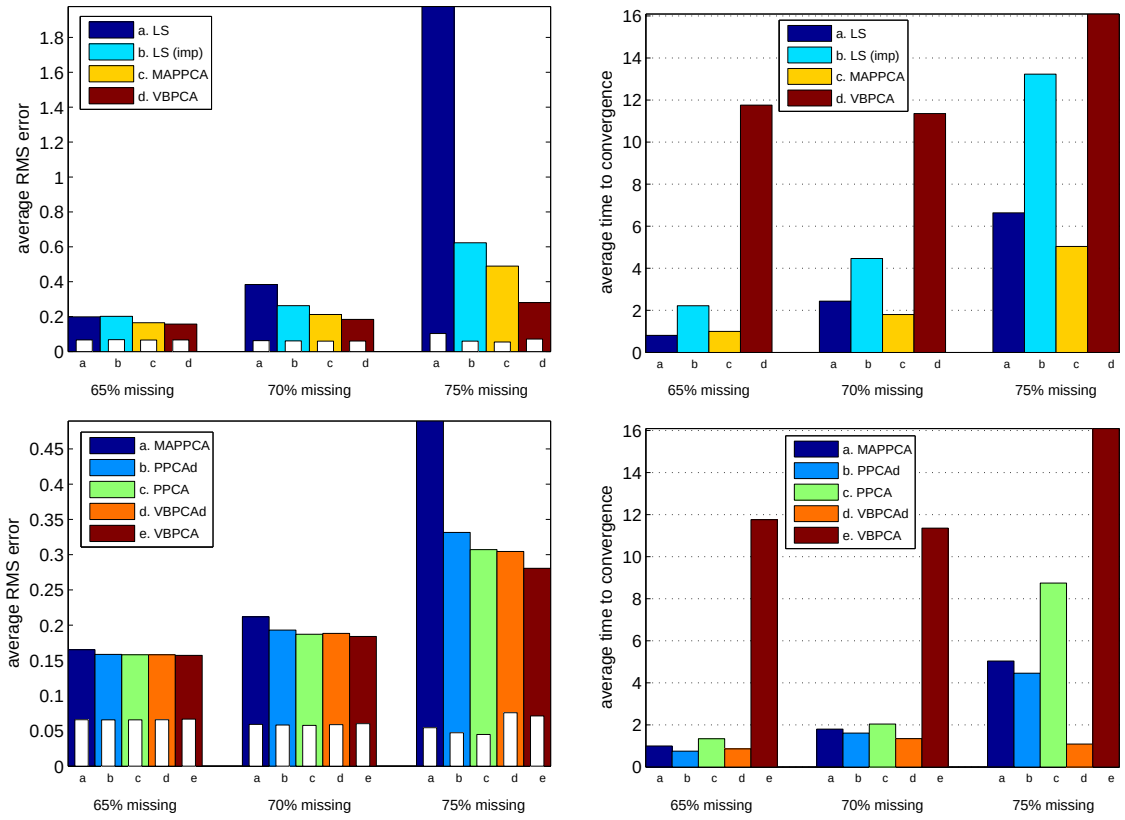


Figure 9: Left: Average test rms errors (colored bars) and training rms errors (white bars) obtained from 30 realizations of data set 100-10-g. Right: Average time to convergence, in seconds, estimated in the same way.

was that the solutions with the smaller cost function values generally provided better reconstruction accuracy. Similar results were obtained for VBPCAd (not shown here).

Thus, in order to find a *globally* good VB solution, one could, in practice, run the learning algorithm many times. The algorithm should be run until *full convergence*, otherwise the cost function values cannot be used for model comparison. Running VBPCA can be computationally heavy because it is the slowest algorithm among the considered ones. Therefore, VBPCAd is an attractive alternative for large-scale data sets.

7.4 Experiments with the Netflix Data

We have tested different PCA approaches in the task of collaborative filtering. The Netflix (2007) problem is a collaborative filtering task that has received a lot of attention recently. It consists of movie ratings given by $n = 480189$ customers to $d = 17770$ movies. There are $N = 100480507$ ratings from 1 to 5 given, and the task is to predict 2817131 other ratings among the same group of customers and movies. 1408395 of the ratings are reserved for validation (or probing). Note that 98.8% of the values are thus missing.

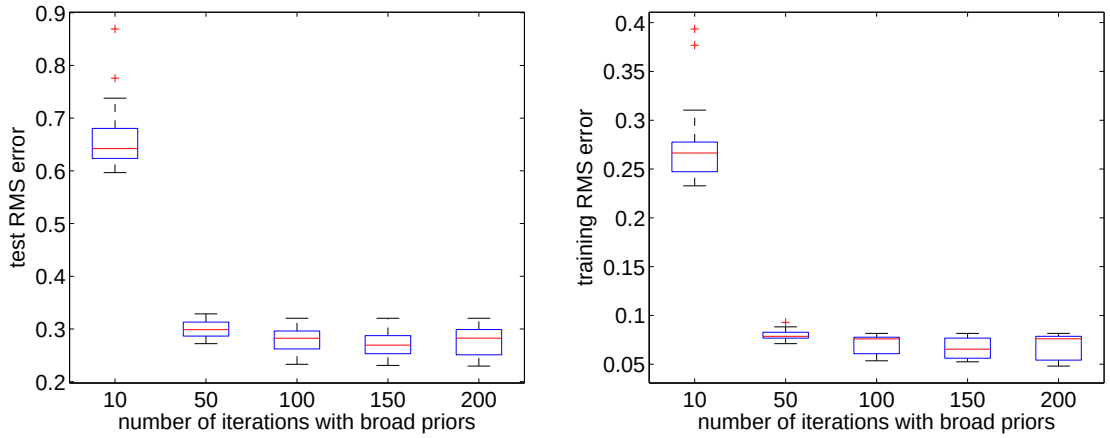


Figure 10: The underfitting effect for VBPCA when the priors are updated too early: Box plots of test (left) and training (right) RMS errors depending on the number of iterations with fixed broad priors for **W** (x-axes) at the beginning of learning. VBPCA was applied to 30 data sets of type 100-10-g with 75% of missing values.

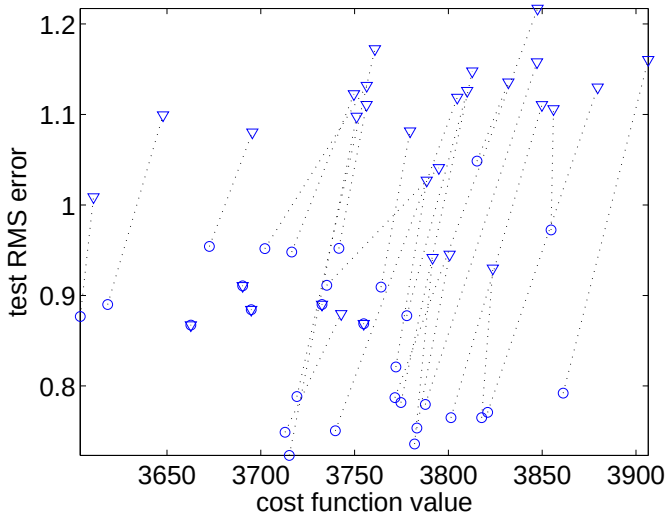


Figure 11: Test RMS error (y-axis) against cost function values (x-axis) obtained with VBPCA for 30 realizations of the data set 100-10-g with 80% of missing values. Two initializations were used for each realization: random (the corresponding local minima are shown with triangles) and with 100 iterations of PPCAd (the corresponding local minima are shown with circles). The dotted lines connect two solutions obtained for the same realization.

The PCA approach is one of the most popular techniques considered by the Netflix contestants (e.g., Funk, 2006; Bell et al., 2007; Salakhutdinov et al., 2007; Paterek, 2007; Lim and Teh, 2007). In our recent conference papers (Raiko et al., 2007a, 2008), we tried to estimate $c = 15$ principal components from the data using unregularized (LS) PCA, MAPPCA, and VBPCAd approaches. We reproduce the results for the LS approaches in Fig. 12. The results clearly show that overfitting is a serious problem for this sparse data set and the LS approach fails. It is also evident that the alternating optimization algorithm is very slow and it can hardly be used for such high-dimensional data. The imputation algorithm is also infeasible in this problem (Raiko et al., 2008). The proposed remedy is to use probabilistic approaches.

Fig. 13 presents the results obtained with MAPPCA and VBPCAd using a varying number of principal components. MAPPCA starts to overfit after about five hours of computation, that is, the RMS error on the test data (lower curves) starts to degrade even if the RMS error on the training data is still diminishing. This does not happen with VBPCAd as the validation error seems to decrease monotonically. The experiments also confirm that VBPCAd is computationally scalable to very large problems. We also tried to run VBPCA, but the straightforward Matlab implementation turned out to be too slow to produce meaningful results. The experiments were run on a dual cpu AMD Opteron SE 2220 using Matlab and the implementations did not use parallel computing.

Our best RMS error for the probing Netflix data was 0.9055 and it was achieved with VBPCAd with 50 components. The same model re-trained using both training and probing data provided an RMS error 0.8990 on the quiz set. The only pre/post-processing that we used was bounding the reconstructions $\hat{y}_{ij}$ such that $1 \leq \hat{y}_{ij} \leq 5$. This is a good result compared to conceptually similar models developed by other contestants. For example, Bell et al. (2007) reported an RMS error of 0.9135 with basic factor analysis with 40 components. Lim and Teh (2007) studied a model which basically implements VBPCA and they reported an RMS error of 0.9141 for a model with 30 components. Salakhutdinov and Mnih (2008b) reported the probing RMS error 0.9280 for an SVD method which essentially implements the LS approach. Their constrained probabilistic matrix factorization model provided a probing RMS error 0.9016 but it used extra informations and more sophisticated priors. The accuracy of our VBPCAd model is comparable to the accuracy of an MCMC approach applied to a similar generative model (Salakhutdinov and Mnih, 2008a). The RMS error achieved with a model with 60 components trained by MCMC was 0.8989 for the quiz set.

Besides reconstruction of missing values, probabilistic PCA methods provide extra information which can be useful in practice. For example, one can predict the uncertainty of the provided reconstructions, as discussed in Section 5. Fig. 14 shows good correlation between the probing RMS errors and the uncertainty of the reconstructions computed by VBPCAd. It is clear that the predicted uncertainty is underestimated, which is largely because of the approximations used in modeling the posterior distributions.

The estimated uncertainty can provide additional information when PCA is used as a preprocesing step for more sophisticated methods. For example, Bell and Koren (2007) used PCA results for constructing a user-oriented neighborhood model for collaborative filtering. Because each column of matrix $\mathbf{X}$ computed by PCA can be considered as a collection of some features associated with a particular user, the similarity between n users can be estimated based on the similarity between the columns of $\mathbf{X}$. The information about the uncertainty of the feature estimates could be used to build a robust similarity measure: Users who have rated few movies (and whose features

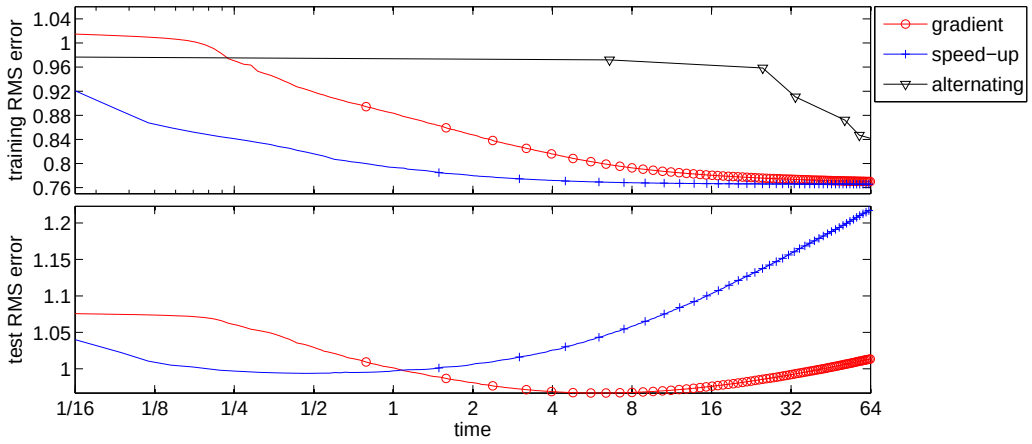


Figure 12: Learning curves for LS algorithms in the Netflix problem. The RMS error (y-axis) is plotted against the processor time in hours. The upper curves show the training error while the lower curves show the test error (for the probing data provided by Netflix). Note that the time scale is logarithmic. The probing error provided by the alternating algorithm is not shown because it is too large compared to the other two curves. The two gradient-based approaches were implemented using parallel computing (this implementation was about twice as fast as the implementation on a single CPU).

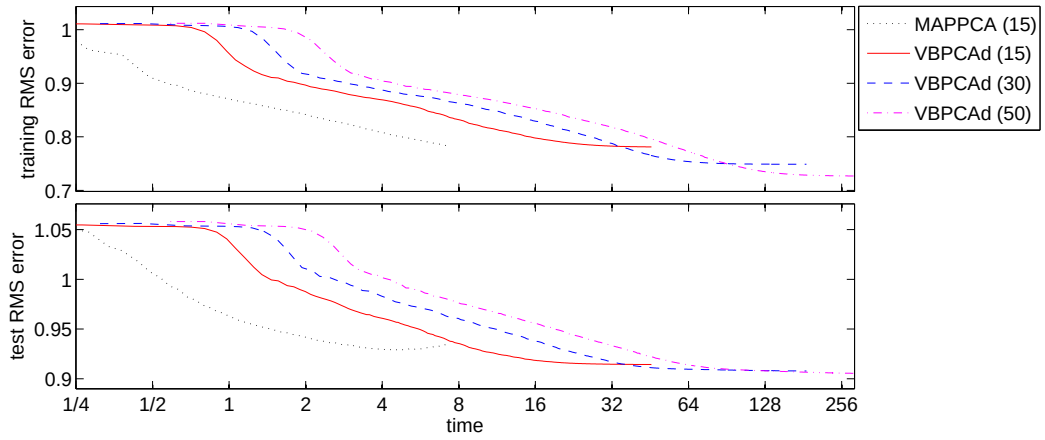


Figure 13: Learning curves for regularized methods in the Netflix problem. The RMS error (y-axis) is plotted against the processor time in hours. The upper curves show the training error while the lower curves show the test error (for the probing data provided by Netflix). Note that the time scale is logarithmic.

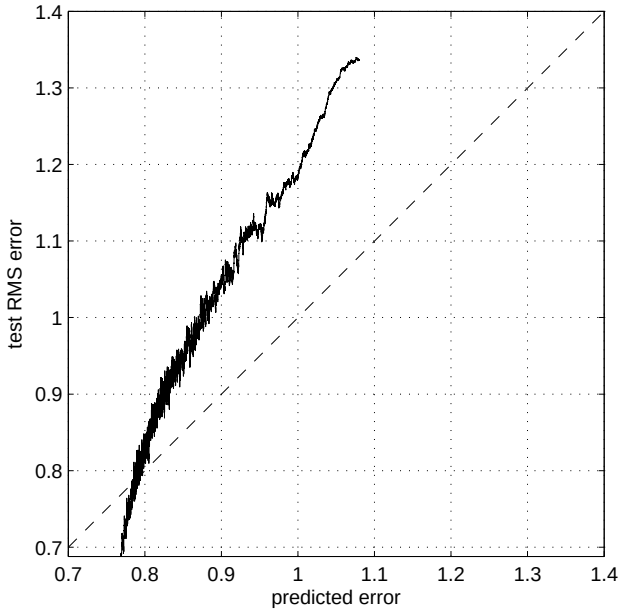


Figure 14: Uncertainty predicted by VBPCAd for the Netflix probing data. The RMS error for the probing data (y-axis) is plotted against the square root of the reconstruction variance estimated by VBPCAd with 50 components (x-axis). To show the dependency more clearly, the estimates of the probing RMS errors are obtained by averaging over a batch of ratings with similar predicted uncertainties.

are estimated with high uncertainty) should not affect the predictions of their neighbors as much as users who have rated many movies (and whose features are estimated more accurately).

Finally, we note that the accuracy achieved by the best teams in the Netflix competition is significantly better than the numbers reported here. However, that does not mean that PCA (and its implementations discussed in this paper) is of no use in collaborative filtering tasks. On the contrary, PCA or its close relatives were used by the leading teams (e.g., Bell et al., 2007) as an important element of the final blend of models. The key to success in that competition was in combining a vast collection of different models rather than in perfecting a single approach (e.g., The Ensemble, 2009).

8. Discussion

We have reviewed the problem of PCA learning in the presence of missing values and discussed various approaches to it. We demonstrated that the simplicity of PCA is lost when introducing missing values. Firstly, the estimation of the bias term and the covariance matrix of the data becomes difficult and thus the solution by eigen-decomposition cannot be used directly. Secondly, the convergence to a unique solution cannot be guaranteed even for the simplest PCA models.

Missing values also make the problem of overfitting more relevant to PCA, so there is a need for some form of regularization. In regularized solutions, reconstructions of data vectors are no longer

conditions	small-scale problems	large-scale problems
few missing values	imputation	gradient-based LS
$d \ll n$	PPCA	PPCAd
many missing values or $d \approx n$	VBPCA	VBPCAd

Table 6: Recommendations on the use of different PCA algorithms.

the projections of the data to the principal subspace. Regularization can be done elegantly using probabilistic models.

Inference and learning in the relevant probabilistic models are generally more complex compared to the complete data case. For example, the posterior covariance of principal components $\Sigma_{\mathbf{x}_j}$ is no longer same for each sample. Thus, a diagonal posterior covariance $\Sigma_{\mathbf{x}_j}$ is no longer sufficient for finding the optimal PCA solution. Regularized models typically have more local minima of the optimized cost function. In particular, there can be local minima corresponding to zero values of hyperparameters.

Another type of treatment for random variables is to use MCMC methods. Their benefits compared to VBPCA includes not having to assume posterior independence like in (23) and being able to vary the number of components (Hoff, 2008), but the downsides include worse interpretability, higher computational complexity, and the need to store samples in case the model is to be applied to new incoming data. Salakhutdinov and Mnih (2008a) used Gibbs sampling initialized with the MAPPCA method for the Netflix problem. The resulting accuracy was similar to ours with similar number of components (0.8989 with 60 components against our 0.8990 for VBPCAd with 50 components).

Large-scale PCA problems require fast converging learning algorithms which allow for efficient implementation. Additionally, sparse data sets require that the amount of computer memory would scale linearly with the number of observed values regardless of the data matrix dimensionalities. In this paper, efficient solutions are proposed based on fully factorial variational approximations and approximate Newton's iteration for the relevant optimization procedure. An alternative learning algorithm based on natural gradient was discussed in our conference paper (Raiko et al., 2008).

The choice of the right PCA algorithm depends on a particular data analysis problem. We gave some recommendations in Sections 7.1–7.3 and we summarize our recommendations in Table 6. The LS approaches can be applied to data sets with relatively few missing data: The *imputation* algorithm can be a good choice for smaller-scale problems and the gradient-based LS algorithms would be good for large data sets. When the number n of samples is relatively large and the data matrix is relatively densely populated, PPCA is a proper choice (or PPCAd for large-scale problems). VB algorithms can be most efficient when there are very few samples ($d \approx n$) or for very sparse data sets. Again, VBPCAd would be a better alternative for large-scale problems.

A Matlab toolbox which contains implementations of all the considered PCA techniques is available online at `http://www.cis.hut.fi/projects/bayes/`. Some of the implemented algorithms scale well to large-scale sparse data sets, which is achieved by low level implementation of core computations and by support of parallel computing. For example, the experiments with the Netflix data reported in Fig. 13 were obtained using the provided Matlab code.

Acknowledgments

The authors would like to thank Juha Karhunen, Erkki Oja, Alexey Kaplan, and Jaakko Luttinen for useful discussions. This work was supported by the Academy of Finland (projects 'Unsupervised machine learning in latent variable models', 'Novel machine learning techniques for studying climate variability and its impacts', 'Auditory approaches to automatic speech recognition' and 'Bayesian Machine Learning Algorithms for Discovering Relations with Latent Variable Models'), and by the IST Program of the European Community, under the PASCAL2 Network of Excellence. This publication only reflects the authors' views. We would like to thank Netflix (2007) for providing the data.

Appendix A. Notes on the Imputation Algorithm

The imputation algorithm can be seen as the application of the EM algorithm (Dempster et al., 1977) to the model

$$\mathbf{y}_j = \mathbf{W}\mathbf{x}_j + \mathbf{m} + \boldsymbol{\varepsilon}_j$$

with isotropic Gaussian noise $p(\boldsymbol{\varepsilon}_j) = \mathcal{N}(\boldsymbol{\varepsilon}_j; 0, v_y\mathbf{I})$ and model parameters $\mathbf{W}$, $\mathbf{x}_j$, $\mathbf{m}$, v_y. Note that $\mathbf{x}_j$ belong to the model parameters and missing values in $\mathbf{y}_j$ are treated as *hidden variables*. This is contrary to the probabilistic PCA model (see Section 3.1) where $\mathbf{x}_j$ are treated as hidden variables. Note also that with this interpretation, there is no well defined inference procedure for a new data case outside of the training set.

Following the view of the EM algorithm presented by Neal and Hinton (1999), we can write down the cost function

$$\begin{aligned} C_{\text{imp}}(\theta, v_y, q(Y_{\text{mis}})) &= \int q(Y_{\text{mis}}) \log \frac{q(Y_{\text{mis}})}{p(\mathbf{Y}|\theta, v_y)} dY_{\text{mis}} \\ &= \int q(Y_{\text{mis}}) \log \frac{q(Y_{\text{mis}})}{p(Y_{\text{mis}}|\theta, v_y)} dY_{\text{mis}} - \log p(Y_{\text{obs}}|\theta, v_y)\,, \end{aligned} \tag{40}$$

which should be minimized w.r.t. model parameters $\theta = \{\mathbf{W}, \mathbf{m}, \mathbf{x}_j, \forall j\}$, v_y and the pdf over the missing data $q(Y_{\text{imp}})$. We denoted the missing data by $Y_{\text{mis}} = \{y_{ij}\,, ij \notin O\}$ and the observed data by $Y_{\text{obs}} = \{y_{ij}\,, ij \in O\}$.

Minimization of (40) w.r.t. $q(Y_{\text{mis}})$ for fixed θ and v_y yields that $q(Y_{\text{mis}}) = p(Y_{\text{mis}}|\theta, v_y)$ because the first term in (40) is simply the Kullback-Leibler divergence between the two pdfs. It is straightforward that

$$p(Y_{\text{mis}}|\theta, v_y) = \prod_{ij \in O} \mathcal{N}(y_{ij}; \hat{y}_{ij}(\theta), v_y)\,,$$

where $\hat{y}_{ij}(\theta)$ are the reconstructed missing values using (5) with the current estimates θ. Thus, infilling missing data with the reconstructions $\hat{y}_{ij}(\theta)$ is the step of the imputation algorithm which corresponds to the E-step of the EM-algorithm.

Collecting the terms in (40) which depend only on θ and v_y gives

$$\begin{aligned} &-\int q(Y_{\text{mis}}) \log p(Y_{\text{obs}}, Y_{\text{mis}}|\theta, v_y) dY_{\text{mis}} \\ &= -\frac{dn}{2}\log 2\pi v_y - \frac{1}{2v_y}\sum_{ij\in O}\left(y_{ij}-\hat{y}_{ij}(\theta)\right)^2 - \frac{1}{2v_y}\sum_{ij\notin O}\left[\left(\bar{y}_{ij}-\hat{y}_{ij}(\theta)\right)^2 + v_y\right] \\ &= -\frac{dn}{2}\log 2\pi v_y - \frac{1}{2v_y}\sum_{ij}\left(y_{ij}-\hat{y}_{ij}(\theta)\right)^2 - \frac{dn-N}{2}. \end{aligned} \tag{41}$$

It is easy to see that minimization of (41) w.r.t. θ is equivalent to minimizing the LS cost (2) for the infilled data matrix. This is done in the imputation algorithm by SVD of the complete data matrix. Thus, the M-step of the EM-algorithm corresponds to performing SVD in the imputation algorithm. Minimization of (41) w.r.t. v_y does not affect the steps of the imputation algorithm.

The imputation algorithm implicitly minimizes the cost function (4), thus it belongs to the class of LS PCA algorithms. It can be shown (Neal and Hinton, 1999) that the minimum of (40) coincides with the minimum of minus log-likelihood

$$-\log p(Y_{\text{obs}}|\theta, v_y) = \frac{N}{2}\log 2\pi v_y + \frac{1}{2v_y}\sum_{ij\in O}\left(y_{ij}-\hat{y}_{ij}(\theta)\right)^2. \tag{42}$$

Replacing v_y in (42) with its maximum likelihood estimate yields that the function being minimized is

$$\frac{N}{2}\left[\log\left(\frac{2\pi}{N}\sum_{ij\in O}\left(y_{ij}-\hat{y}_{ij}(\theta)\right)^2\right)+1\right],$$

from which it follows that the imputation algorithm implicitly optimizes the sum-squared error.

Appendix B. Conditions Fulfilled at the Convergence of Probabilistic PCA

The variational view of the EM algorithm allows for an interpretation of the PPCA learning algorithm in which the cost function

$$C = \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{X})} d\mathbf{X} - \int q(\mathbf{X}) \log p(\mathbf{Y}|\mathbf{W}, \mathbf{X}, \mathbf{m}, v_y) d\mathbf{X} \tag{43}$$

is minimized w.r.t. to the model parameters $\mathbf{W}$, $\mathbf{m}$, v_y and the pdf $q(\mathbf{X})$. The E-step fixes the model parameters and minimizes the cost function w.r.t. the distribution $q(\mathbf{X})$ while the M-step minimizes C w.r.t. $\mathbf{W}$, $\mathbf{m}$ and v_y assuming that $q(\mathbf{X})$ is fixed.

Let us consider simple transformations of $\{\mathbf{W}, \mathbf{m}, q(\mathbf{X})\}$ which do not change the second term in (43). For example, subtraction of a constant $\mathbf{b}$ from the columns of $\mathbf{X}$ can be compensated by changing the bias term $\mathbf{m}$ correspondingly:

$$q(\mathbf{X}) \leftarrow \prod_{j=1}^{n} \mathcal{N}\left(\mathbf{x}_j; \bar{\mathbf{x}}_j - \mu, \Sigma_{\mathbf{x}_j}\right), \tag{44}$$

$$\mathbf{m} \leftarrow \mathbf{m} + \mathbf{W}\mu. \tag{45}$$

because $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{m} = \mathbf{W}(\mathbf{x} - \mu) + (\mathbf{W}\mu + \mathbf{m})$. Similarly, rotation of the columns of $\mathbf{X}$ can be compensated by changing $\mathbf{W}$:

$$q(\mathbf{X}) \leftarrow \prod_{j=1}^{n} \mathcal{N}\left(\mathbf{x}_j; \mathbf{A}\overline{\mathbf{x}}_j, \mathbf{A}\Sigma_{\mathbf{x}_j}\mathbf{A}^{\mathrm{T}}\right), \tag{46}$$

$$\mathbf{W} \leftarrow \mathbf{W}\mathbf{A}^{-1} \tag{47}$$

because $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{m} = (\mathbf{W}\mathbf{A}^{-1})(\mathbf{A}\mathbf{x}) + \mathbf{m}$.

These transformations generally affect the first term in (43) which is the Kullback-Leibler divergence between $q(\mathbf{X})$ and $p(\mathbf{X})$:

$$D = \sum_{j=1}^{n} \int q(\mathbf{x}_j) \log \frac{q(\mathbf{x}_j)}{p(\mathbf{x}_j)} d\mathbf{x}_j = \frac{1}{2} \sum_{j=1}^{n} \left[\mathrm{tr}(\Sigma_{\mathbf{x}_j}) + \overline{\mathbf{x}}_j^{\mathrm{T}} \overline{\mathbf{x}}_j - \log\det \Sigma_{\mathbf{x}_j} \right]. \tag{48}$$

and at the convergence this term cannot be made smaller by transformations of the form (44)–(45) or (46)–(47).

Transformation (44)–(45) changes (48) to

$$D = \frac{1}{2} \sum_{j=1}^{n} \left[\mathrm{tr}(\Sigma_{\mathbf{x}_j}) + (\overline{\mathbf{x}}_j - \mu)^{\mathrm{T}} (\overline{\mathbf{x}}_j - \mu) - \log\det \Sigma_{\mathbf{x}_j} \right].$$

Now taking the derivative w.r.t. μ and equating it to zero gives the optimal $\mu = \frac{1}{n}\sum_{j=1}^{n} \overline{\mathbf{x}}_j$. At the convergence, the optimal μ should be zero and therefore (34) should hold.

Similarly, transformation (46)–(47) gives

$$\begin{aligned} D &= \frac{1}{2} \sum_{j=1}^{n} \left[\mathrm{tr}(\mathbf{A}\Sigma_{\mathbf{x}_j}\mathbf{A}^{\mathrm{T}}) + \overline{\mathbf{x}}_j^{\mathrm{T}} \mathbf{A}^{\mathrm{T}} \mathbf{A} \overline{\mathbf{x}}_j - \log\det(\mathbf{A}\Sigma_{\mathbf{x}_j}\mathbf{A}^{\mathrm{T}}) \right] \\ &= \frac{n}{2} \mathrm{tr}(\mathbf{A}\Sigma_* \mathbf{A}^{\mathrm{T}}) - \frac{1}{2} \sum_{j=1}^{n} \log\det(\mathbf{A}\Sigma_{\mathbf{x}_j}\mathbf{A}^{\mathrm{T}}), \end{aligned}$$

where $\Sigma_* = \frac{1}{n}\sum_{j=1}^{n} \left[\overline{\mathbf{x}}_j \overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j} \right]$. Taking the derivative w.r.t. $\mathbf{A}$ gives

$$\mathbf{A}\Sigma_* - (\mathbf{A}^{\mathrm{T}})^{-1} = 0$$

and therefore the optimal $\mathbf{A}$ satisfies $\mathbf{A}\Sigma_*\mathbf{A}^{\mathrm{T}} = \mathbf{I}$. Transformation (46)–(47) yields that $\Sigma_* \leftarrow \mathbf{A}\Sigma_*\mathbf{A}^{\mathrm{T}}$ and therefore at the convergence $\Sigma_* = \mathbf{I}$, which is condition (35).

Appendix C. Variational Bayesian PCA (VBPCA)

The following form of the posterior approximation is used:

$$q(\mathbf{W}, \mathbf{X}, \mathbf{m}) = \prod_{i=1}^{d} \mathcal{N}(m_i; \overline{m}_i, \widetilde{m}_i) \prod_{i=1}^{d} \mathcal{N}(\mathbf{w}_i; \overline{\mathbf{w}}_i, \Sigma_{\mathbf{w}_i}) \prod_{j=1}^{n} \mathcal{N}\left(\mathbf{x}_j; \overline{\mathbf{x}}_j, \Sigma_{\mathbf{x}_j}\right).$$

The update of the principal components:

$$\Sigma_{\mathbf{x}_j} = v_y\Big(v_y\mathbf{I} + \sum_{i\in O_j}\left[\overline{\mathbf{w}}_i\overline{\mathbf{w}}_i^{\mathrm{T}} + \Sigma_{\mathbf{w}_i}\right]\Big)^{-1},$$

$$\overline{\mathbf{x}}_j = \frac{1}{v_y}\Sigma_{\mathbf{x}_j}\sum_{i\in O_j}\overline{\mathbf{w}}_i(y_{ij}-\overline{m}_i), \qquad j = 1,\ldots,n.$$

The update of the bias term and matrix $\mathbf{W}$:

$$\overline{m}_i = \frac{v_m}{|O_i|(v_m + v_y/|O_i|)}\sum_{j\in O_i}[y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}}\overline{\mathbf{x}}_j], \tag{49}$$

$$\widetilde{m}_i = \frac{v_y v_m}{|O_i|(v_m + v_y/|O_i|)}, \tag{50}$$

$$\Sigma_{\mathbf{w}_i} = v_y\Big(v_y\,\mathrm{diag}(v_{w,k}^{-1}) + \sum_{j\in O_i}\left[\overline{\mathbf{x}}_j\overline{\mathbf{x}}_j^{\mathrm{T}} + \Sigma_{\mathbf{x}_j}\right]\Big)^{-1},$$

$$\overline{\mathbf{w}}_i = \frac{1}{v_y}\Sigma_{\mathbf{w}_i}\sum_{j\in O_i}\overline{\mathbf{x}}_j(y_{ij}-\overline{m}_i), \qquad i = 1,\ldots,d$$

and the variance parameters:

$$v_y = \frac{1}{N}\sum_{ij\in O}\left[(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}}\overline{\mathbf{x}}_j - \overline{m}_i)^2 + \widetilde{m}_i + \overline{\mathbf{w}}_i^{\mathrm{T}}\Sigma_{\mathbf{x}_j}\overline{\mathbf{w}}_i + \overline{\mathbf{x}}_j^{\mathrm{T}}\Sigma_{\mathbf{w}_i}\overline{\mathbf{x}}_j + \mathrm{tr}(\Sigma_{\mathbf{x}_j}\Sigma_{\mathbf{w}_i})\right],$$

$$v_{w,k} = \frac{1}{d}\sum_{i=1}^{d}\left(\overline{w}_{ik}^2 + \widetilde{w}_{ik}\right), \qquad v_m = \frac{1}{d}\sum_{i=1}^{d}\left(\overline{m}_i^2 + \widetilde{m}_i\right), \tag{51}$$

where $\widetilde{w}_{ik}$ is the k-th element on the diagonal of $\Sigma_{\mathbf{w}_i}$.

Appendix D. VBPCA with Fully Factorial Approximation (VBPCAd)

VBPCAd uses the fully factorial posterior approximation

$$q(\mathbf{W},\mathbf{X},\mathbf{m}) = \prod_{i=1}^{d}\mathcal{N}(m_i;\overline{m}_i,\widetilde{m}_i)\prod_{i=1}^{d}\prod_{k=1}^{c}\mathcal{N}(w_{ik};\overline{w}_{ik},\widetilde{w}_{ik})\prod_{k=1}^{c}\prod_{j=1}^{n}\mathcal{N}\left(x_{kj};\overline{x}_{kj},\widetilde{x}_{kj}\right).$$

The VB cost function is then the sum of the following terms:

$$C_{\mathrm{vb}} = \sum_{ij\in O}C_{yij} + \sum_{i=1}^{d}C_{mi} + \sum_{i=1}^{d}\sum_{k=1}^{c}C_{wik} + \sum_{k=1}^{c}\sum_{j=1}^{n}C_{xkj},$$

where the individual terms are

$$C_{yij} = \frac{1}{2v_y}\left[(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}}\overline{\mathbf{x}}_j - \overline{m}_i)^2 + \widetilde{m}_i + \sum_{k=1}^{c}\left(\widetilde{w}_{ik}\overline{x}_{kj}^2 + \overline{w}_{ik}^2\widetilde{x}_{kj} + \widetilde{w}_{ik}\widetilde{x}_{kj}\right)\right] + \frac{1}{2}\log 2\pi v_y,$$

$$C_{mi} = \left\langle\log\frac{q(m_i)}{p(m_i)}\right\rangle = \frac{\overline{m}_i^2 + \widetilde{m}_i}{2v_m} - \frac{1}{2}\log\frac{\widetilde{m}_i}{v_m} - \frac{1}{2},$$

$$C_{wik} = \left\langle\log\frac{q(w_{ik})}{p(w_{ik})}\right\rangle = \frac{\overline{w}_{ik}^2 + \widetilde{w}_{ik}}{2v_{w,k}} - \frac{1}{2}\log\frac{\widetilde{w}_{ik}}{v_{w,k}} - \frac{1}{2},$$

$$C_{xkj} = \left\langle\log\frac{q(x_{kj})}{p(x_{kj})}\right\rangle = \frac{1}{2}(\overline{x}_{kj}^2 + \widetilde{x}_{kj}) - \frac{1}{2}\log\widetilde{x}_{kj} - \frac{1}{2}.$$

Variational parameters $\overline{m}_i$, $\widetilde{m}_i$, $v_{w,k}$ and v_m are updated to minimize the cost function using the update rules in (49)–(51). Parameters $\widetilde{w}$, $\widetilde{x}$ can be updated to minimize the cost function:

$$\widetilde{w}_{ik} = v_y \left[\frac{v_y}{v_{w,k}} + \sum_{j \in O_i} \left(\overline{x}_{kj}^2 + \widetilde{x}_{kj} \right) \right]^{-1}, \tag{52}$$

$$\widetilde{x}_{kj} = v_y \left[v_y + \sum_{i \in O_j} \left(\overline{w}_{ik}^2 + \widetilde{w}_{ik} \right) \right]^{-1}. \tag{53}$$

The update rule for v_y is

$$v_y = \frac{1}{N} \sum_{ij \in O} \left[(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}} \overline{\mathbf{x}}_j - \overline{m}_i)^2 + \widetilde{m}_i + \sum_{k=1}^{c} \left(\widetilde{w}_{ik} \overline{x}_{kj}^2 + \overline{w}_{ik}^2 \widetilde{x}_{kj} + \widetilde{w}_{ik} \widetilde{x}_{kj} \right) \right].$$

Minimizing the cost function w.r.t. each parameter $\overline{w}_{ij}$ keeping the others fixed would lead to a slow algorithm (e.g., Honkela et al., 2003). Instead, derivatives required for gradient-based optimization are:

$$\frac{\partial C_{\mathrm{vb}}}{\partial \overline{w}_{ik}} = \frac{\overline{w}_{ik}}{v_{w,k}} + \frac{1}{v_y} \sum_{j \in O_i} \left[-\left(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}} \overline{\mathbf{x}}_j - \overline{m}_i \right) \overline{x}_{kj} + \overline{w}_{ik} \widetilde{x}_{kj} \right],$$

$$\frac{\partial C_{\mathrm{vb}}}{\partial \overline{x}_{kj}} = \overline{x}_{kj} + \frac{1}{v_y} \sum_{i \in O_j} \left[-\left(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}} \overline{\mathbf{x}}_j - \overline{m}_i \right) \overline{w}_{ik} + \widetilde{w}_{ik} \overline{x}_{kj} \right].$$

The second-order derivatives which can be used to speed up learning, as explained in Section 6.1, coincide with the inverse of the updated variances given in (52)–(53): $\partial^2 C_{\mathrm{vb}} / \partial \overline{w}_{ik}^2 = \widetilde{w}_{ik}^{-1}$ and $\partial^2 C_{\mathrm{vb}} / \partial \overline{x}_{kj}^2 = \widetilde{x}_{kj}^{-1}$.

Appendix E. MAPPCA

The cost function C_{MAP} is given in (39). The alternating optimization procedure can be implemented as follows:

$$\mathbf{x}_j = \left(v_y \mathbf{I} + \sum_{i \in O_j} \mathbf{w}_i \mathbf{w}_i^{\mathrm{T}} \right)^{-1} \sum_{i \in O_j} \mathbf{w}_i (y_{ij} - m_i), \qquad j = 1, \ldots, n,$$

$$m_i = \frac{v_m}{|O_i|(v_m + v_y/|O_i|)} \sum_{j \in O_i} \left[y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j \right],$$

$$\mathbf{w}_i = \left(v_y \, \mathrm{diag}(v_{w,k}^{-1}) + \sum_{j \in O_i} \mathbf{x}_j \mathbf{x}_j^{\mathrm{T}} \right)^{-1} \sum_{j \in O_i} \mathbf{x}_j (y_{ij} - m_i), \qquad i = 1, \ldots, d,$$

$$v_y = \frac{1}{N} \sum_{ij \in O} (y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i)^2,$$

$$v_{w,k} = \frac{1}{d} \sum_{i=1}^{d} w_{ik}^2, \qquad v_m = \frac{1}{d} \sum_{i=1}^{d} m_i^2.$$

Gradient-based learning discussed in Section 6.1 requires the following derivatives:

$$\frac{\partial C_{\mathrm{MAP}}}{\partial w_{ik}} = \frac{w_{ik}}{v_{w,k}} + \frac{1}{v_y} \sum_{j \in O_i} \left[-\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i\right) x_{kj} \right],$$

$$\frac{\partial C_{\mathrm{MAP}}}{\partial x_{kj}} = x_{kj} + \frac{1}{v_y} \sum_{i \in O_j} \left[-\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i\right) w_{ik} \right],$$

$$\frac{\partial^2 C_{\mathrm{MAP}}}{\partial w_{ik}^2} = \frac{1}{v_{w,k}} + \frac{1}{v_y} \sum_{j \in O_i} x_{kj}^2,$$

$$\frac{\partial^2 C_{\mathrm{MAP}}}{\partial x_{kj}^2} = 1 + \frac{1}{v_y} \sum_{i \in O_j} w_{ik}^2.$$

References

T. W. Anderson. Maximum likelihood estimates for a multivariate normal distribution when some observations are missing. *Journal of the American Statistical Association*, 52:200–203, 1957.

R. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2007)*, 2007.

R. Bell, Y. Koren, and C. Volinsky. The BellKor solution to the Netflix Prize. Available at `http://www.netflixprize.com/`, 2007.

C. M. Bishop. Variational principal components. In *Proceedings of the 9th International Conference on Artificial Neural Networks (ICANN99)*, pages 509–514, 1999.

C. M. Bishop. *Pattern Recognition and Machince Learning*. Springer, Cambridge, 2006.

W. J. Boscardin and X. Zhang. Modeling the covariance and correlation matrix of repeated measures. In A. Gelman and X.-L. Meng, editors, *Applied Bayesian Modeling and Causal Inference from an Incomplete-Data Perspective*. John Wiley & Sons, New York, 2004.

A. Christoffersson. *The One Component Model with Incomplete Data*. PhD thesis, Uppsala University, 1970.

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

C. L. de Ligny, G. H. E. Nieuwdorp, W. K. Brederode, W. E. Hammers, and J. C. van Houwelingen. An application of factor analysis with missing data. *Technometrics*, 23(1):91–95, 1981.

R. E. Dear. A principal components missing data method for multiple regression models. Technical Report SP-86, Santa Monica: Systems Development Corporation, 1959.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39(1):1–38, 1977.

K. Diamantaras and S. Kung. *Principal Component Neural Networks - Theory and Application*. Wiley, 1996.

S. Funk. Netflix update: Try this at home. Available at `http://sifter.org/~simon/journal/20061211.html`, December 2006.

Z. Ghahramani and M. I. Jordan. Learning from incomplete data. Technical report CBCL-108, Massachusetts Institute of Technology, 1994.

B. Grung and R. Manne. Missing values in principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 42(1):125–139, 1998.

S. Haykin. *Modern Filters*. Macmillan, 1989.

G. E. Hinton and D. van Camp. Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory*, pages 5–13, Santa Cruz, CA, USA, 1993.

P. Hoff. Model averaging and dimension selection for the singular value decomposition. *Journal of the American Statistical Association*, 102(478):674–685, 2008.

T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

A. Honkela, H. Valpola, and J. Karhunen. Accelerating cyclic update algorithms for parameter estimation by pattern searches. *Neural Processing Letters*, 17(2):191–203, 2003.

A. Ilin and H. Valpola. On the effect of the form of the posterior approximation in variational learning of ICA models. *Neural Processing Letters*, 22(2):183–204, 2005.

I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 2nd edition, 2002.

A. Kaplan, Y. Kushnir, M. Cane, and M. Blumenthal. Reduced space optimal analysis for historical datasets: 136 years of Atlantic sea surface temperatures. *Journal of Geophysical Research*, 102: 27835–27860, 1997.

Y. J. Lim and Y. W. Teh. Variational Bayesian approach to movie rating prediction. In *Proceedings of of KDD Cup and Workshop 2007, held during the Thirteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, California, 2007.

R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. J. Wiley & Sons, 1987.

J. Luttinen and A. Ilin. Transformations in variational Bayesian factor analysis to speed up learning. *Neurocomputing*, 73(7–9):1093–1102, 2010.

T. Minka. Automatic choice of dimensionality for PCA. In V. Tresp T. Leen, T. Dietterich, editor, *Advances in Neural Information Processing Systems 13*, pages 598–604. MIT Press, Cambridge, MA, USA, 2001.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. The MIT Press, Cambridge, MA, USA, 1999.

Netflix. Netflix prize webpage, 2007. http://www.netflixprize.com/.

S. Oba, M. Sato, I. Takemasa, M. Monden, K. Matsubara, and S. Ishii. A Bayesian missing value estimation method for gene expression profile data. *Bioinformatics*, 19(16):2088–2096, 2003.

E. Oja. *Subspace Methods of Pattern Recognition*. Research Studies Press and J. Wiley, 1983.

A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, 2007.

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559–572, 1901.

T. Raiko and H. Valpola. Missing values in nonlinear factor analysis. In *Proceedings of the 8th International Conference on Neural Information Processing (ICONIP'01)*, pages 822–827, Shanghai, 2001.

T. Raiko, A. Ilin, and J. Karhunen. Principal component analysis for large scale problems with lots of missing values. In *Proceedings of the 18th European Conference on Machine Learning (ECML 2007)*, pages 691–698, Warsaw, Poland, 2007a.

T. Raiko, H. Valpola, M. Harva, and J. Karhunen. Building blocks for variational Bayesian learning of latent variable models. *Journal of Machine Learning Research*, 8:155–201, 2007b.

T. Raiko, A. Ilin, and J. Karhunen. Principal component analysis for sparse high-dimensional data. In *Proceedings of the 14th International Conference on Neural Information Processing (ICONIP 2007)*, pages 566–575, Kitakyushu, Japan, 2008.

S. Roweis. EM algorithms for PCA and SPCA. In *Advances in Neural Information Processing Systems 10*, pages 626–632, Cambridge, MA, 1998. MIT Press.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning (ICML-2008)*, Helsinki, Finland, 2008a.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20*, 2008b.

R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning (ICML-2007)*, 2007.

N. Srebro and T. Jaakkola. Weighted low-rank approximations. In *Proceedings of the 20th International Conference on Machine Learning (ICML-2003)*, Washington DC, 2003.

The Ensemble. Website of the runner-up in the Netflix competition, 2009. http://www.the-ensemble.com/.

M. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

C. S. Wallace. Classification by minimum-message-length inference. In S. G. Aki, F. Fiala, and W. W. Koczkodaj, editors, *Advances in Computing and Information – ICCI '90*, volume 468 of *Lecture Notes in Computer Science*, pages 72–81. Springer, Berlin, 1990.

T. Wiberg. Computation of principal components when data are missing. In Johannes Gordesch and Peter Naeve, editors, *COMPSTAT 1976: Proceedings in Computational Statistics, 2nd symposium held in Berlin (West)*, pages 229–236. Wien: Physica-Verlag, 1976.

G. Young. Maximum likelihood estimation and factor analysis. *Psychometrika*, 6(1):49–53, 1941.