TKK Reports in Information and Computer Science

Espoo 2008 TKK-ICS-R6

PRACTICAL APPROACHES TO PRINCIPAL COMPONENT ANALYSIS IN THE PRESENCE OF MISSING VALUES

Alexander Ilin and Tapani Raiko

TEKNILLINEN KORKEAKOULU
TEKNISKA HÖGSKOLAN
HELSINKI UNIVERSITY OF TECHNOLOGY
TECHNISCHE UNIVERSITÄT HELSINKI
UNIVERSITE DE TECHNOLOGIE D'HELSINKI

TKK Reports in Information and Computer Science

Espoo 2008 TKK-ICS-R6

PRACTICAL APPROACHES TO PRINCIPAL COMPONENT ANALYSIS IN THE PRESENCE OF MISSING VALUES

Alexander Ilin and Tapani Raiko

Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Information and Computer Science

Teknillinen korkeakoulu
Informaatio- ja luonnontieteiden tiedekunta
Tietojenkäsittelytieteen laitos

Distribution:
Helsinki University of Technology
Faculty of Information and Natural Sciences
Department of Information and Computer Science
P.O.Box 5400
FI-02015 TKK
FINLAND
URL: http://ics.tkk.fi
Tel. +358 9 451 1
Fax +358 9 451 3369
E-mail: series@ics.tkk.fi

ISBN 978-951-22-9481-7 (Print)
ISBN 978-951-22-9482-4 (Online)
ISSN 1797-5034 (Print)
ISSN 1797-5042 (Online)
URL: http://lib.tkk.fi/Reports/2008/isbn9789512294824.pdf

TKK ICS
Espoo 2008

Abstract: Principal component analysis (PCA) is a classical data analysis technique that finds linear transformations of data that retain 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. Probabilistic formulation of PCA provides a good foundation for handling missing values, and we introduce 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

CONTENTS

1 INTRODUCTION

Principal component analysis (PCA) is a data analysis technique that can be traced back to [26]. It can be used to compress datasets 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 (see, e.g., [17, 4, 9, 12, 6]). 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 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-dimensional 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-dimensional vectors of principal components, and $\mathbf{m}$ is a d-dimensional bias vector. We assume that $c \leq d \leq n$. Principal subspace methods [6, 9] find $\mathbf{W}$, $\mathbf{X}$ 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. Without any further constraints, there exist infinitely many ways to perform such a decomposition. However, the subspace spanned by the column vectors of the matrix $\mathbf{W}$, called the *principal subspace*, is unique. PCA is a specific representation in the principal subspace. It is traditionally defined using the requirement that the columns 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, see [9, 17, 12] for details.

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.[1] We also formulate the PCA solution as the one which yields matrix $\mathbf{W}$ with mutually orthogonal columns and uncorrelated principal components which are scaled to unit variance, that is

$$\mathbf{W}^{\mathrm{T}}\mathbf{W} = \mathrm{diag}(\|\mathbf{W}_{:k}\|^2)\,, \qquad \|\mathbf{W}_{:k}\| \geq \|\mathbf{W}_{:l}\|\,, \quad k < l \tag{3}$$

$$\frac{1}{n}\sum_{j=1}^{n} \mathbf{x}_j \mathbf{x}_j^{\mathrm{T}} = \mathbf{I}\,, \tag{4}$$

where $\mathrm{diag}(v_k)$ denotes a diagonal matrix with diagonal elements v_k and $\mathbf{W}_{:k}$ is the k-th column of matrix $\mathbf{W}$.

[1]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 later.

In the matrix notation, the data vectors and principal components can be compiled into a $d \times n$ 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 $\mathbf{Y} \approx \mathbf{WX} + \mathbf{M}$ and the minimized cost function is the sum of the squared elements (or Frobenius norm) of matrix $\mathbf{Y} - \mathbf{WX} - \mathbf{M}$:

$$C = \|\mathbf{Y} - \mathbf{WX} - \mathbf{M}\|_F^2 . \tag{5}$$

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 [20], that is, the missingness does not depend on the unobserved data. 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 $\mathbf{Y} \approx \mathbf{WX} + \mathbf{M}$ for the observed values. The rest of the matrix $\mathbf{WX} + \mathbf{M}$ 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, for example, the optimal value of the bias term $\mathbf{m}$ in (1) is not generally equal to the row-wise mean of the data matrix, as in classical PCA. 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) Standard PCA approaches can easily lead to overfitting, thus regularization is often required. 4) The algorithms may require heavy computations, especially for large-scale problems. 5) The concept of the PCA basis in the principal subspace is not easily generalized in the presence of missing values. 6) 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.

PCA in the presence of missing values can be relevant for many datasets which appear in practical applications. Datasets may 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, there are applications in which available observations are 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 (or producing personal recommendations) by using other people's preferences (see, e.g., [13]) or historical data reconstruction in climatic records (see, e.g., [18]).

Historically, the missing value problem in PCA was first studied by Dear [8] 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 [35]. Christofferson [5] also used a one-component model for reconstructing missing values. Wiberg [34] first suggested to directly minimize the mean-square error of the observed part of the data. An algorithm in [7] already worked up to half of the values missing. Missing values problem using a multivariate normal distribution has been studied even earlier than using PCA, for instance by [1]. More historical references can be found in [17].

More recently, PCA with missing values was studied by Grung and Manne [11]. 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 [33] introduced the probabilistic formulation of PCA (PPCA). They mention missing data only shortly and do not give any formulas. Bishop [3] introduces variational Bayesian PCA (VBPCA) for choosing the number of components in PCA. Raiko and Valpola [29] reconstructed missing values with VBPCA to compare some nonlinear models to it. Oba [23] applied VBPCA for missing value estimation in gene expression profile data, and mentions 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 mean 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 EM interpretation.

Section 3 presents probabilistic models for PCA which provide a good foundation for handling missing values. First, we introduce formulas for the recently studied probabilistic PCA model in the case with missing values. We show how the notion of the PCA basis can be extended to probabilistic PCA and explain how to find it efficiently given the principal subspace. Later, 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.

Section 4 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 5, 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. Finally we conclude in Section 6.

The following notation is used throughout the article. 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 points. 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. The i-th row of $\mathbf{W}$ is denoted by $\mathbf{w}_i$, and the j-th column of $\mathbf{X}$ is $\mathbf{x}_j$. $\mathring{\mathbf{Y}}$ is the data matrix with zeros in the places of missing values.

2 LEAST SQUARES TECHNIQUES

2.1 Multiple local minima in the cost function

The minimum mean-square error compression of data is the formulation for PCA [35] 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 [34]:

$$C = \sum_{ij \in O} (y_{ij} - \hat{y}_{ij})^2 , \tag{6}$$

$$\hat{y}_{ij} = \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j + m_i = \sum_{k=1}^{c} w_{ik} x_{kj} + m_i . \tag{7}$$

In this section, we consider algorithms optimizing the cost function (6). We call it the least squares (LS) approach to PCA for 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. On the contrary, the cost function (6) can have multiple local minima. Let us demonstrate this using a simple example in which the 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. 1: The global minimum $\mathbf{W} = \pm[0.49 \quad 0.62 \quad 0.62]^{\mathrm{T}}$ 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 these

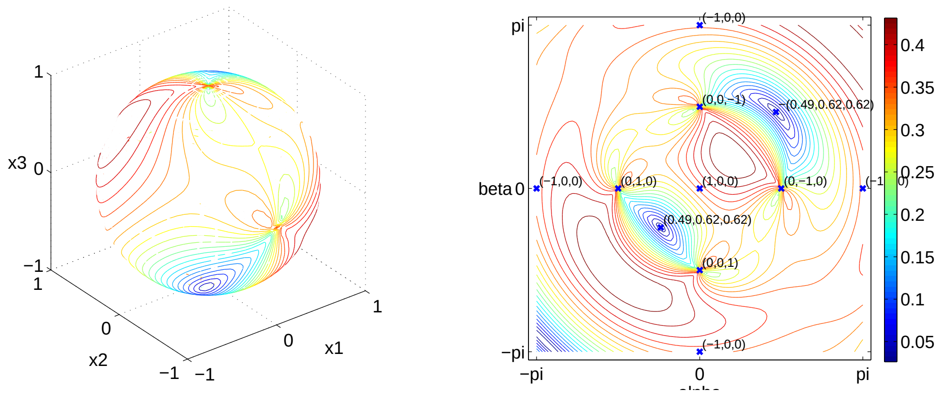

Figure 1: Example of local minima for the cost function (6). $d = 3$, $c = 1$, PCA solution can be defined by a unit length vector $\mathbf{W}$. Left: The cost function plotted on a surface of unit length vectors. Right: The same plot using the Euler vector representation: The matrix $\mathbf{W}$ is 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 $a_{13} = -a_{31} = \beta$.

two local solutions reconstructs perfectly three out of four observed values in $\mathbf{Y}$.

The cost function (6) can be minimized using any optimization procedure. Two possible approaches are presented below. After that, we present how the PCA basis can be found within the principal subspace. Finally, we discuss problems that arise with these methods when the number of missing values increases.

2.2 Alternating W–X algorithm

Complete data: It is possible to optimize the cost function (5), 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

$$\mathbf{X} \leftarrow (\mathbf{W}^{\mathrm{T}}\mathbf{W})^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{Y}\,, \qquad \mathbf{W} \leftarrow \mathbf{Y}\mathbf{X}^{\mathrm{T}}(\mathbf{X}\mathbf{X}^{\mathrm{T}})^{-1}. \tag{8}$$

This iteration is especially efficient when only a few principal components are needed, that is $c \ll d$ [31]. 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 [11] studied the alternating algorithm in the case of missing values. They assumed that the bias term is removed, for example, by subtracting the row-wise mean of $\mathbf{Y}$ from each row of $\mathbf{Y}$. In order to get the accurate least squares solution, we include the bias term into the estimation procedure, yielding the update rules

$$\mathbf{x}_j = (\mathbf{W}_j^{\mathrm{T}}\mathbf{W}_j)^{-1}\mathbf{W}_j^{\mathrm{T}}\,(\mathring{\mathbf{Y}}_{:j} - \mathbf{M}_j)\,, \quad j = 1, \ldots, n \tag{9}$$

$$m_i = \frac{1}{|O_i|}\sum_{j \in O_i}\left[y_{ij} - \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j\right] \tag{10}$$

$$\mathbf{w}_i^{\mathrm{T}} = \left(\mathring{\mathbf{Y}}_{i:} - m_i\right)^{\mathrm{T}}\mathbf{X}_i^{\mathrm{T}}(\mathbf{X}_i\mathbf{X}_i^{\mathrm{T}})^{-1}\,, \quad i = 1, \ldots, d\,. \tag{11}$$

where m_i is the i-th element of $\mathbf{m}$, $\mathbf{W}_j$ is matrix $\mathbf{W}$ in which an i-th row is replaced with zeros if y_{ij} is missing, vector $\mathbf{M}_j$ is formed from $\mathbf{m}$ similarly to $\mathbf{W}_j$, and $\mathbf{X}_i$ is matrix $\mathbf{X}$ in which a j-th column is replaced with zeros if y_{ij} is missing:

$$(\mathbf{W}_j)_{i:} = \begin{cases} \mathbf{w}_i\,, & ij \in O \\ 0\,, & ij \notin O \end{cases}$$

$$(\mathbf{M}_j)_{i:} = \begin{cases} m_i\,, & ij \in O \\ 0\,, & ij \notin O \end{cases}$$

$$(\mathbf{X}_i)_{:j} = \begin{cases} \mathbf{x}_j\,, & ij \in O \\ 0\,, & ij \notin O \end{cases}. \tag{12}$$

2.3 Oja-Karhunen (online learning) algorithm

Complete data: The basis of neural network implementation of PCA learning rules is a gradient descent optimization procedure. Such algorithms work on-line processing only one input vector $\mathbf{y}_j$ at once. Thus, 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 w.r.t. matrix $\mathbf{W}$ yields the update rule

$$\mathbf{W} \leftarrow \mathbf{W} + \gamma(\mathbf{Y} - \mathbf{W}\mathbf{X})\mathbf{X}^{\mathrm{T}}\,, \tag{13}$$

where $\gamma > 0$ is called the learning rate. Minimization w.r.t. matrix $\mathbf{X}$ can be performed using the least squares solution in (8). However, some neural approaches use learning rules which either explicitly orthogonalize $\mathbf{W}$ or which yield an orthogonal $\mathbf{W}$ at the convergence. Thus the update of $\mathbf{X}$ can be simplified to $\mathbf{X} = \mathbf{W}^{\mathrm{T}}\mathbf{Y}$, which together with (13) is the batch version of the Oja-Karhunen learning algorithm [24, 9]. 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 (7). Matrix $\mathbf{X}$ could then be updated using (9), as in the standard Oja-Karhunen approach. However, a gradient-based update can also be used for $\mathbf{X}$:

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

The bias term can be updated using (10). As we discuss later (see Section 4.2), gradient-based learning can be computationally more efficient than the alternating algorithm because there is no need to compute matrices $(\mathbf{W}_j^{\mathrm{T}}\mathbf{W}_j)^{-1}$ and $(\mathbf{X}_i\mathbf{X}_i^{\mathrm{T}})^{-1}$ in (9) and (11).

2.4 Rotation in the principal subspace to the PCA basis

The PCA basis of the principal subspace allows some intuitive interpretation of the compressed data: A nice property of PCA is that it orders the principal components according to the amount of data variance they explain. However, algorithms which minimize the cost functions (2) or (6) generally converge to any basis in the principal subspace. Thus transformation of the final solution to the PCA basis might be needed to allow interpretation of the results.

An estimated principle subspace can be transformed into the PCA solution in the sense of (3)–(4) by ensuring that the column vectors of $\mathbf{W}$ are mutually orthogonal and that the row vectors of $\mathbf{X}$ are also mutually orthogonal, while keeping the product $\mathbf{WX}$ the same. The solution is to compute the eigen-decomposition

$$\frac{1}{n}\mathbf{X}\mathbf{X}^{\mathrm{T}} = \mathbf{U}_{\mathbf{X}}\mathbf{D}_{\mathbf{X}}\mathbf{U}_{\mathbf{X}}^{\mathrm{T}} \tag{14}$$

and the singular value decomposition

$$\mathbf{W}\mathbf{U}_{\mathbf{X}}\mathbf{D}_{\mathbf{X}}^{1/2} = \mathbf{U}_{\mathbf{W}}\mathbf{\Sigma}_{\mathbf{W}}\mathbf{V}_{\mathbf{W}}^{\mathrm{T}}.$$

The transformed solution is $\mathbf{W}_{\text{pca}} = \mathbf{U}_{\mathbf{W}}\mathbf{\Sigma}_{\mathbf{W}}$ and $\mathbf{X}_{\text{pca}} = \mathbf{V}_{\mathbf{W}}^{\mathrm{T}}\mathbf{D}_{\mathbf{X}}^{-1/2}\mathbf{U}_{\mathbf{X}}^{\mathrm{T}}\mathbf{X}$.

This transformation can be done for principal subspaces obtained both for fully and partially observed data. For fully observed data, we know that the $k < c$ first columns of $\mathbf{W}$ form the k-dimensional principal subspace. In the presence of missing values, the rotation guarantees (3)–(4), but still, the solution for $k < c$ dimensional principal subspace might be different.

2.5 Numerical problems and overfitting

The least squares PCA methods can work well for datasets with few missing values but they may not be applicable for sparse datasets because of the severe danger of 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 $\mathbf{W}_j^{\mathrm{T}}\mathbf{W}_j$ is rank deficient and equation (9) cannot be used. Moreover, even if $|O_j|$ is greater than c, matrix $\mathbf{W}_j^{\mathrm{T}}\mathbf{W}_j$ 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. 1.

The same problem can happen when the rows of $\mathbf{W}$ are estimated using, for example, (11): matrix $\mathbf{X}_i\mathbf{X}_i^{\mathrm{T}}$ can be rank deficient or badly conditioned. This is more probable when some rows of $\mathbf{Y}$ contain very few measurements. However, these problem can generally appear for any dataset and for any algorithm optimizing the cost function (6) regardless of the exact optimization procedure.

Overfitting problems can be avoided by using proper regularization. One way to prevent unbounded growth of model parameters is to add terms which penalize large parameter values into the cost function. This can be done elegantly using probabilistic models, which is discussed in Section 3.4.

2.6 Method using singular value decomposition

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

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

where $\mathbf{U}$ is a $d \times d$ orthogonal matrix, $\mathbf{V}$ is an $n \times n$ orthogonal matrix and $\mathbf{\Sigma}$ is a $d \times n$ pseudo-diagonal matrix (diagonal if $d = n$) with the singular values on the main diagonal [12]. The PCA solution is obtained by selecting the c largest singular values from $\mathbf{\Sigma}$, by forming $\mathbf{W}$ from the corresponding c columns of $\mathbf{U}$, and $\mathbf{X}$ from the corresponding c rows of $\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$ [17].

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 $\mathbf{D}$ contains the eigenvalues of $\mathbf{C}$, and the columns of the matrix $\mathbf{U}$ contain the unit-length eigenvectors of $\mathbf{C}$ in the same order [6, 9, 17, 12]. Again, the columns of $\mathbf{U}$ corresponding to the largest eigenvalues are taken as $\mathbf{W}$, and $\mathbf{X}$ 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 $\mathbf{V}$.

Incomplete data: The same approach cannot be directly used in the presence of missing values. Estimating the covariance matrix $\mathbf{C}$ becomes difficult: For example, if we estimate $\mathbf{C}$ for the data matrix below, leaving out terms with missing values from the average, we get for the estimate of the covariance matrix

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

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.

A simple alternative is an iterative procedure which alternates between imputing the missing values in $\mathbf{Y}$ and applying standard PCA to the completed data matrix (see, e.g., [17]). Initially, the missing values can be replaced, for example, by the row-wise means of $\mathbf{Y}$. The covariance matrix of the complete data can be estimated without the problems mentioned above. Next, $\mathbf{W}\mathbf{x}_j + \mathbf{m}$ can be used as a better estimate for the missing values and PCA can be applied to the updated data matrix. 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 (6) 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, it does provide an elegant remedy against overfitting. The cost function C_{ia} which is minimized by performing SVD can be represented as a sum of two terms: C_{obs} which is the true optimized function and a penalty term C_{mis} which keeps reconstructions of the missing values close to the imputed values (see Appendix A for details). Initialization of the reconstructions with the row-wise means of the data matrix introduces a bias towards the most "conservative" solutions.

Our experiments show that the imputation algorithm also has resistance against overlearning when the training error decreases but the generalization ability of the trained model gets worse. Badly overfitted solutions correspond to regions in the parameter space where a small decrease in the cost function (for training data) can cause a large increase of the reconstruction error (for test data). However, the term C_{mis} makes the algorithm shorten the steps and learning can practically stop once the algorithm enters regions in which the cost function changes very little.

3 PROBABILISTIC MODELS FOR PCA

3.1 Probabilistic PCA

Probabilistic formulation of PCA offers a number of benefits, including well-founded regularization, model comparison, interpretation of results, and extendability. *Probabilistic PCA* (PPCA) [33] explicitly includes the noise term in a generative model

$$\mathbf{y}_j = \mathbf{W}\mathbf{x}_j + \mathbf{m} + \boldsymbol{\epsilon}_j \, . \qquad (15)$$

Both the principal components $\mathbf{x}_j$ and the noise $\boldsymbol{\epsilon}_j$ are assumed Gaussian:

$$p(\mathbf{x}_j) = \mathcal{N}\left(\mathbf{x}_j; 0, \mathbf{I}\right) , \qquad p(\boldsymbol{\epsilon}_j) = \mathcal{N}\left(\boldsymbol{\epsilon}_j; 0, v\mathbf{I}\right) , \qquad (16)$$

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

Complete data: The E-step estimates the conditional distribution of the hidden variables given the data and the current values of the model parameters:

$$p(\mathbf{X}|\mathbf{Y},\mathbf{W},v) = \prod_{j=1}^{n} \mathcal{N}(\mathbf{x}_j;\overline{\mathbf{x}}_j,\boldsymbol{\Sigma}_{\mathbf{x}})$$

where the means $\overline{\mathbf{x}}_j$ can be summarized in a matrix $\overline{\mathbf{X}}$ (similarly to matrix $\mathbf{X}$) and the covariance $\boldsymbol{\Sigma}_{\mathbf{x}}$ is same for each $\mathbf{x}_j$:

$$\overline{\mathbf{X}} = \boldsymbol{\Psi}^{-1}\mathbf{W}^{\mathrm{T}}\mathbf{Y}\,, \qquad \boldsymbol{\Sigma}_{\mathbf{x}} = v\boldsymbol{\Psi}^{-1}\,, \qquad \boldsymbol{\Psi} = \mathbf{W}^{\mathrm{T}}\mathbf{W} + v\mathbf{I}\,. \tag{17}$$

The M-step re-estimates the model parameters as

$$\mathbf{W} = \mathbf{Y}\overline{\mathbf{X}}^{\mathrm{T}}(n\boldsymbol{\Sigma}_{\mathbf{x}} + \overline{\mathbf{X}}\overline{\mathbf{X}}^{\mathrm{T}})^{-1} \tag{18}$$

$$v = \frac{1}{nd}\sum_{i=1}^{d}\sum_{j=1}^{n}\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}}\overline{\mathbf{x}}_j\right)^2 + \frac{1}{d}\,\mathrm{tr}(\mathbf{W}\boldsymbol{\Sigma}_{\mathbf{x}}\mathbf{W}^{\mathrm{T}})\,.$$

Tipping and Bishop showed that $\mathbf{W}$ converges to the principal subspace [33]. In the zero-noise limit, that is for $v \to 0$, the iterations (17)–(18) reduce to the least squares projections in (8) [31]. Note also that the ML estimator for $\mathbf{m}$ is given by the row-wise mean of the data [33]. 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 yields the following modifications to the EM-algorithm. The E-step:

$$\overline{\mathbf{x}}_j = \boldsymbol{\Psi}_j^{-1}\mathbf{W}_j^{\mathrm{T}}(\mathring{\mathbf{Y}}_{:j} - \mathbf{M}_j)\,, \qquad \boldsymbol{\Sigma}_{\mathbf{x},j} = v\boldsymbol{\Psi}_j^{-1} \tag{19}$$

$$\boldsymbol{\Psi}_j = \mathbf{W}_j^{\mathrm{T}}\mathbf{W}_j + v\mathbf{I}\,, \qquad j = 1,\ldots,n\,. \tag{20}$$

The M-step:

$$m_i = \frac{1}{|O_i|}\sum_{j\in O_i}\left[y_{ij} - \mathbf{w}_i^{\mathrm{T}}\overline{\mathbf{x}}_j\right] \tag{21}$$

$$\mathbf{w}_i^{\mathrm{T}} = \left(\mathring{\mathbf{Y}}_{i:} - m_i\right)^{\mathrm{T}}\overline{\mathbf{X}}_i^{\mathrm{T}}\left(\overline{\mathbf{X}}_i\overline{\mathbf{X}}_i^{\mathrm{T}} + \sum_{j\in O_i}\boldsymbol{\Sigma}_{\mathbf{x},j}\right)^{-1}, \qquad i = 1,\ldots,d \tag{22}$$

$$v = \frac{1}{N}\sum_{ij\in O}\left(y_{ij} - \mathbf{w}_i^{\mathrm{T}}\overline{\mathbf{x}}_j - m_i\right)^2 + \frac{1}{N}\sum_{ij\in O}\mathbf{w}_i^{\mathrm{T}}\boldsymbol{\Sigma}_{\mathbf{x},j}\mathbf{w}_i\,,$$

where $\overline{\mathbf{X}}_i$ is defined similarly to (12).

There are several important 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 $\boldsymbol{\Sigma}_{\mathbf{x},j}$ 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 $\overline{\mathbf{X}}$. This makes the computations much more involved.

3.2 Rotation to the PCA basis for PPCA

PPCA is a principal subspace method because the ML estimates of the model parameters form an arbitrary basis in the principal subspace. The generalization of the classical PCA basis to the case of PPCA is not very straightforward. We give a possible definition of the PCA basis in the subspace found by PPCA.

It can be shown (see Appendix B) that the following holds for a PPCA solution at the convergence:

$$\frac{1}{n}\sum_{j=1}^{n}\overline{\mathbf{x}}_j = 0\,. \tag{23}$$

$$\boldsymbol{\Sigma}_* = \frac{1}{n}\Big(\overline{\mathbf{X}}\overline{\mathbf{X}}^{\mathrm{T}} + \sum_{j=1}^{n}\boldsymbol{\Sigma}_{\mathbf{x},j}\Big) = \mathbf{I}\,. \tag{24}$$

Any orthogonal rotation of this subspace, that is transformation $\mathbf{Qx}$ with an orthogonal matrix $\mathbf{Q}$, does not affect (23)–(24). Notice that in the noiseless limit (when $v \to 0$) $\boldsymbol{\Sigma}_*$ is the sample covariance matrix of the principal components and it is equal to the identity matrix, as required by (4). Therefore, a (practically) unique PPCA solution can be defined as the *orthogonal* basis for which (23)–(24) are fulfilled together with the PCA requirement in (3). Then, the columns $\mathbf{W}_{:k}$ normalized to unit norm and the squared norms $\|\mathbf{W}_{:k}\|^2$ is the analogue of the eigenvectors and eigenvalues of the data covariance matrix in classical PCA.

A transformation of the PPCA solution can easily be performed such that (23)–(24) and (3) hold. The means $\overline{\mathbf{x}}_j$ are simply centralized with a proper update of the bias term $\mathbf{m}$ (see Appendix B). Then, a linear transformation of the subspace can be performed similarly to as explained in Section 2.4, with the exception that $\boldsymbol{\Sigma}_*$ is used in (14). Such a transformation can be performed once at the end of learning to enable interpretation of the results. However, our experiments show faster convergence when this transformation is also performed during learning between iterations of the EM algorithm.

3.3 Overfitting examples

PPCA provides some remedy against overfitting compared to the simplest least squares algorithms. First, a nonzero noise level v regularizes badly conditioned matrices $\mathbf{W}_j^{\mathrm{T}}\mathbf{W}_j$ in (19)–(20). Second, even if the noise level is small, badly conditioned matrices $\boldsymbol{\Psi}_j$ result in large values both in the means $\overline{\mathbf{x}}_j$ and in the covariance matrices $\boldsymbol{\Sigma}_{\mathbf{x},j}$. This diminishes the effect of large values in $\overline{\mathbf{x}}_j$ when $\mathbf{W}$ is re-estimated using (22).

However, PPCA can overfit, for example, if the estimated number of principal components is unreasonably large. Let us consider an example when the observation space is two-dimensional ($d = 2$) and each data vector is only partly observed, that is either y_{1j} or y_{2j} is known for all js. These observations are represented by triangles placed on the two axes in Fig. 2. A solution which minimizes the PPCA cost function to zero is defined by a line, as shown in the upper sub-plot. The missing values are then reconstructed by points lying on the line. In fact, there are infinitely many solutions for $(\mathbf{W}, v)$ which are equally good in terms of the cost function. This is illustrated in the lower sub-plots in Fig. 2. The solution provided by the learning algorithm depends on the initial conditions. Different solutions provide very distinct reconstructions but the reconstruction is naïve for all of them. The situation would not change significantly if there were a few fully observed data vectors: the found solution would be defined by those few samples and it would not generalize well for new data.

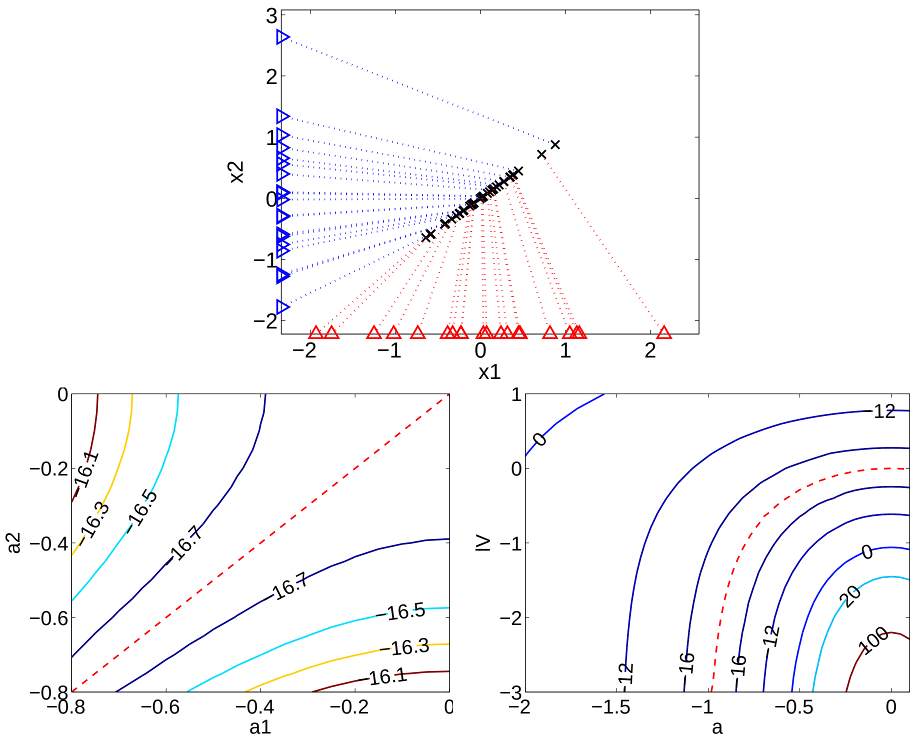

Figure 2: A simple example of overfitting with PPCA: $d = 2$ and each column of $\mathbf{Y}$ contains only one observed value. Top: The measurements are marked with blue and red triangles at the two axes. The corresponding reconstructions are shown with crosses. Bottom left: The cost as a function of the two elements of $\mathbf{W}$. Parameter v has been optimized to minimize the cost. Bottom right: The cost as a function of v and $\mathbf{W}$ in which the two elements are equal. In the two plots at the bottom, the red dashed line represents parameter values which minimize the cost function.

Note that in the presence of missing values, the number of components, which can be estimated to provide good generalization, may be smaller than the "true" number of principal components underlying the data. This type of overfitting could be avoided using tools which can estimate the right number of components needed for reliable modeling, for example, cross-validation or Bayesian techniques.

The second example shows that PPCA may overfit when observed values are distributed unevenly in the data matrix. Let us assume that three-dimensional ($d = 3$) 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 (see Fig. 3a). Then, the reconstruction provided by a trained PPCA model can be very inaccurate for the third row.

It relies on the two available observations (see Fig. 3b), which causes serious overfitting. Such situations can be relevant to some real-world applications. For example, historical climatic records typically contain few measurements in the polar regions.

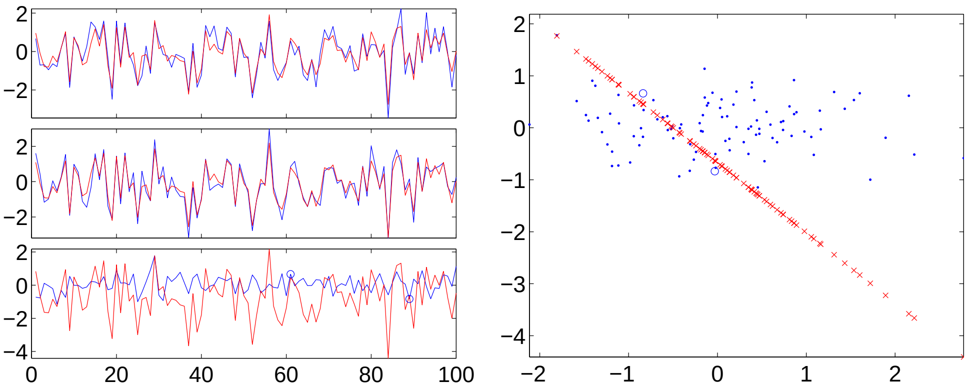

Figure 3: An example when PPCA overfits for rows of $\mathbf{Y}$ which contain very few measurements. The three-dimensional ($d = 3$) data are described well by one principal component ($c = 1$). Left: The three rows of $\mathbf{Y}$ are shown with blue lines. The first two rows are fully observed, the only two observations in the third row are marked with circles. The PPCA reconstruction is shown with red lines. Right: The blue dots is the scatter plot of the third row of $\mathbf{Y}$ (x-axe) against the principal component estimated by PPCA (y-axe). The dots corresponding to the two observations are marked with circles. The red crosses is the reconstruction of the third row (x-axe) against the estimated principal component (y-axe). The estimated correlation are due to the two fully observed data vectors.

3.4 Bayesian regularization in PCA

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 (15)–(16) can be complemented with a Gaussian prior over the elements of matrix $\mathbf{W}$:[2]

$$p(\mathbf{m}) = \mathcal{N}\left(\mathbf{m}; 0, w_m \mathbf{I}\right) , \qquad p(\mathbf{W}) = \prod_{k=1}^{c} \mathcal{N}\left(\mathbf{W}_{:k}; 0, w_k \mathbf{I}\right) . \tag{25}$$

The model (25) uses shared prior for all elements in the same column of $\mathbf{W}$, parameterized with w_k. This is done to allow automatic selection of the right number of components needed for PCA. The hyperparameters w_m, 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

[2]Here, we use a zero mean prior for $\mathbf{m}$ for the sake of simplicity. Including a mean hyperparameter μ, i.e. $p(\mathbf{m}) = \prod_i \mathcal{N}\left(m_i; \mu, w_m\right)$, can be useful in practice.

w_k should tend to zero. This is called *automatic relevance determination* in the machine learning literature [4]. Such prior also introduces a bias towards the PCA basis within the principal subspace.

3.5 Maximum a posteriori estimation

MAPwx approach: point estimates for W and X

The simplest way to identify the model is to find maximum a posteriori (MAP) estimates of the model parameters. The log-posterior (assuming uniform prior for hyperparameters v, w_k, w_m) is proportional to:

$$-\frac{1}{v}\sum_{ij\in O}(y_{ij}-\mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j-m_i)^2 - N\log 2\pi v - \frac{1}{w_m}\|\mathbf{m}\|^2 - d\log 2\pi w_m \\ -\sum_{k=1}^{c}\left[\frac{1}{w_k}\|\mathbf{W}_{:k}\|^2 + d\log 2\pi w_k + \|\mathbf{X}_{k:}\|^2 + n\log 2\pi\right]. \quad (26)$$

We refer to the method which finds the maximum of (26) of w.r.t. all the unknown quantities $\mathbf{W}$, $\mathbf{X}$, v, w_k, w_m as the *MAPwx approach*.

Maximization of (26) can be done by any suitable optimization procedure (see Appendix C). However, there are some difficulties that should be avoided. First, using an improper prior for hyperparameters leads to a situation where the posterior pdf is infinitely large when $w_k \to 0$, $\|\mathbf{W}_{:k}\| \to 0$. This problem can be overcome, for example, by adding a broad prior for variance hyperparameters into the model. This results in extra terms in (26) penalizing small values of w_k, w_m and v.[3]

The MAPwx approach can suffer from both underfitting and overfitting problems. Underfitting happens when the algorithm gets stuck in the vicinity of a solution with $w_k \approx 0$ for some k. This can be interpreted as the automatic relevance determination deciding that some of the principal components are useless, even before they have been properly fitted to the data. Such solutions may correspond to regions with little changes in the cost functions (plateaus). On the other hand, overfitting can happen since discarding posterior uncertainty in the estimates of $\mathbf{W}$ and $\mathbf{X}$ results in underestimated noise variance v and thus to the problems discussed in Section 2.5.

MAPw approach: Point estimates for W

An approach which is more resistant against overfitting is a direct extension of PPCA to incorporate priors for $\mathbf{W}$ and $\mathbf{m}$. The MAP estimates for $\mathbf{W}$, v, w_k, w_m (thus $\mathbf{X}$ integrated out from the posterior (26)) can be found using the EM algorithm. The resulting learning rules resemble the EM algorithm

[3]In this case, a hyperparameter w, which is a prior variance of a set of zero-mean variables z_i, $i = 1, \ldots, m$, can be updated as $w = \frac{2\beta+\sum_{i=1}^{m}\langle z_i^2\rangle}{2\alpha+m}$, where $\langle z_i^2\rangle$ denotes the expectation of z_i^2 in the posterior, $\alpha = 10^{-3}$ and $\beta = 10^{-3}$ are small constants. This essentially limits the estimate of w from below such that $w \geq \frac{\beta}{\alpha+m/2}$. The ML estimate $w = \frac{1}{m}\sum_{i=1}^{m}\langle z_i^2\rangle$ is achieved when $\alpha, \beta \to 0$. In the rest of the article, we always present the ML update rules for variance hyperparameters to keep the formulas simple. However, properly modified update rules are used in corresponding practical implementations.

for PPCA with the following modifications of the M-step:

$$m_i = \frac{w_m}{|O_i|(w_m + v/|O_i|)} \sum_{j \in O_i} \left[y_{ij} - \mathbf{w}_i^{\mathrm{T}} \overline{\mathbf{x}}_{:j} \right]$$

$$\mathbf{w}_i^{\mathrm{T}} = \left(\mathring{\mathbf{Y}}_{i:} - m_i\right)^{\mathrm{T}} \overline{\mathbf{X}}_i^{\mathrm{T}} \Big(\overline{\mathbf{X}}_i \overline{\mathbf{X}}_i^{\mathrm{T}} + \sum_{j \in O_i} \mathbf{\Sigma}_{\mathbf{x},j} + v \, \mathrm{diag}(w_k^{-1}) \Big)^{-1},$$

$$w_k = \sum_{i=1}^{d} w_{ik}^2 \,, \qquad w_m = \sum_{i=1}^{d} m_i^2 \,,$$

for each $i = 1, \dots, d$. This simple modification of PPCA is often enough to solve the basic overfitting problems considered in Section 3.3.

3.6 Variational Bayesian PCA (VBPCA)

The variational Bayesian (VB) approach to PCA was first introduced in [3] for the case of fully observed data. Let us present it by considering the problem of ML estimation of the hyperparameters $\boldsymbol{\xi} = (v, w_k, w_m)$ in the model defined in (15)–(16), (25). In this case, the model parameters $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{X}, \mathbf{m})$ are treated as *hidden* variables. Application of the EM algorithm to this problem requires computation of the posterior $p(\boldsymbol{\theta}|\mathbf{Y}, \boldsymbol{\xi})$ on the E step, which is generally intractable.

The variational view of the EM algorithm [21] justifies replacing the actual posterior $p(\boldsymbol{\theta}|\mathbf{Y}, \boldsymbol{\xi})$ with a simpler pdf $q(\boldsymbol{\theta})$, which is selected to have a tractable form. Then, the E-step is modified to update the approximation $q(\boldsymbol{\theta})$ so as to minimize the cost function

$$C(q) = \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p(\mathbf{Y}, \boldsymbol{\theta}|\boldsymbol{\xi})} d\boldsymbol{\theta} \,. \tag{27}$$

Thus, the cost function is a function of *variational parameters* $\boldsymbol{\theta}_q$, the parameters of the approximating pdf q. On the M-step, the approximation $q(\boldsymbol{\theta}\,;\ \boldsymbol{\theta}_q)$ can be used as it was the actual posterior $p(\boldsymbol{\theta}|\mathbf{Y}, \boldsymbol{\xi})$.

The complexity of the cost function depends on the form of the approximating pdf $q(\boldsymbol{\theta})$. A computationally convenient form for the PCA model is

$$q(\boldsymbol{\theta}) = \prod_{i=1}^{d} q(m_i) \prod_{i=1}^{d} q(\mathbf{w}_i) \prod_{j=1}^{n} q(\mathbf{x}_j) \,. \tag{28}$$

The cost function can be minimized alternately w.r.t. one factor $q(\boldsymbol{\theta}_i)$ in (28) while keeping the other ones fixed. The E-step is thus divided further into three separate steps that update $\mathbf{W}$, $\mathbf{X}$, and $\mathbf{m}$. Using conjugate priors for model parameters allows computation of the *optimal* pdfs $q(\boldsymbol{\theta}_i)$ on each step. For example, the optimal $q(m_i)$, $q(\mathbf{w}_i)$ and $q(\mathbf{x}_j)$ are Gaussian. Then the update of any of them boils down to re-estimation of the corresponding mean and covariance matrix.

The update rules for VBPCA are given in Appendix D. Note that the update rules resemble the EM algorithm for the PPCA model: The PPCA learning rules can be obtained by setting $\mathbf{\Sigma}_{\mathbf{w}_i} = \mathbf{0}$, $\widetilde{m}_i = 0$, $w_k \to \infty$, and $w_m \to \infty$. 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 while the covariance matrices $\mathbf{\Sigma}_{\mathbf{w}_i}$, $\mathbf{\Sigma}_{\mathbf{x}_{:j}}$ and variances $\widetilde{m}_i$ provide analogue of confidence intervals, that is they estimate the uncertainty about the corresponding unknown quantities.

Note that in the fully Bayesian formulation, the hyperparameters are assigned priors too and all the model parameters are treated equally. The actual cost function is

$$C_{\mathrm{vb}} = \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p(\mathbf{Y}, \boldsymbol{\theta})} d\boldsymbol{\theta} \,,$$

where $\boldsymbol{\theta}$ includes $\mathbf{W}$, $\mathbf{X}$, $\mathbf{m}$ and v, w_k, w_m. We omitted the effect of the prior for v, w_k, w_m in Appendix D to simplify the equations (see a footnote in Section 3.5). A useful property of the VB cost function is that it provides a lower bound of the model evidence in the vicinity of a found solution $\mathcal{M}$:

$$\log p(\mathbf{Y}|\mathcal{M}) \geq C_{\mathrm{vb}} \,. \tag{29}$$

There are important advantages of VBPCA compared to the previously discussed techniques:

1. VBPCA is more resistant against overfitting compared to the MAPwx and MAPw approaches. The reason for that is that VB estimation is sensitive to posterior probability mass rather than posterior probability density. Typically, overfitted solutions correspond to high peaks in the posterior density while robust solutions contain much of the posterior probability mass in their neighborhood.

2. The method provides estimates of the confidence intervals for all unknown quantities. This can be useful to detect unreliable results similar to the one presented in Fig. 3. The pdf of a missing value y_{ij}, $ij \notin O$ can be approximated as a Gaussian with mean $\overline{y}_{ij}$ and variance $\widetilde{y}_{ij}$ (see Appendix D). The latter can be used as a measure of uncertainty of the missing value reconstruction.

3. The value of the cost function can be used for comparison between different VBPCA solutions (local minima).[4] A smaller cost yields a greater lower bound of the solution evidence. This is especially useful in the presence of missing values when the cost function is likely to have multiple local minima even for the simplest models (see Section 2.1).

4 PCA LEARNING FOR LARGE-SCALE PROBLEMS

4.1 Computational complexity

Obtaining a reasonably good solution in appropriate time is a very important issue for high-dimensional problems in which learning may take several days.

[4]Valid model comparison requires proper prior and taking into account posterior uncertainty for all model parameters. However, we discard the corresponding terms of the cost function for hyperparameters because their influence is negligible in practice.

The algorithms presented in the previous sections scale differently to problems with large dimensions and sparse datasets. The computational complexity of different optimization schemes as well as the models in which they are applicable are listed in Table 1. For example, the alternating optimization algorithm for PPCA requires computation and inversion of matrices $\mathbf{\Phi}_i$ and $\mathbf{\Psi}_j$, which are generally unique for each row of $\mathbf{W}$ and each column of $\mathbf{X}$, respectively. The corresponding computational complexity $O(Nc^2+nc^3)$ per iteration[5] is quite a bit heavier than with complete data, whose complexity is $O(ndc)$ [31] per iteration.

Optimization method	Imputation with SVD	Alternating $\mathbf{W}$–$\mathbf{X}$	Gradient (Oja-Karhunen)
Complexity	$O(nd^2)$	$O(Nc^2 + nc^3)$	$O(Nc + nc)$
LS	✓	✓	✓
PPCA		✓	
MAPwx		✓	✓
MAPw		✓	
VBPCA		✓	
PPCAd			✓
VBPCAd			✓

Table 1: Summary of applicability of optimization algorithms to different models, and the 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.

Another important issue for sparse datasets is efficient memory usage. It is preferable that the amount of used memory scales linearly with the number of observed values regardless of the data matrix dimensionalities. Only the imputation algorithm requires memory for the complete data matrix, and even that requirement can be diminished at a cost of greater time complexity.

4.2 Gradient-based learning

The gradient-based optimization scheme, extending the Oja-Karhunen approach to regularized PCA models, can be very efficient for large-scale problems and sparse datasets. 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 $\mathbf{W}$ and $\mathbf{X}$ on the changes in the cost function. This results in slow convergence without proper speed-up procedures (see, e.g., [14]).

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 diago-

[5]The computational complexity of the alternating $\mathbf{W}$–$\mathbf{X}$ scheme can be reduced if the matrices required inversion (e.g., $\mathbf{\Phi}_i$, $\mathbf{\Psi}_j$ for PPCA) are computed once for all rows (columns) of $\mathbf{Y}$ that have observed values at exactly the same columns (rows respectively).

nal 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 (6) 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 [28, 27].

The idea of gradient-based learning can be extended directly to regularized PCA as well. For example, the gradients required for gradient-based implementation of the MAPwx approach are given in Appendix C.

A potential problem of gradient-based learning is that cost functions of regularized PCA models can have regions with almost constant values (plateaus corresponding to one of the principal component set to zero). This can yield very slow convergence once the algorithm enters such area.

4.3 Factorial variational approximations

The problem of overfitting can be especially important for high-dimensional sparse datasets. The situations discussed in Sections 2.5 and 3.3 are more probable when data points are sparsely distributed in high dimensions. There are many more pairs of data dimensions where an unexisting correlation can be falsely invented. Thus, methods penalizing large values of model parameters and taking into account their posterior uncertainty are especially relevant here.

The PPCA, MAPw and VBPCA (Sections 3.1, 3.5 and 3.6 respectively) approaches, which take into account the posterior uncertainty at least in $\mathbf{X}$, are hardly applicable for large-scale problems. First, the computational burden of one iteration of the alternating $\mathbf{W}$–$\mathbf{X}$ scheme is very large (see Table 1). Application of the gradient-based optimization is cumbersome because one needs to update $c \times c$ 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.

The MAPwx approach scales well for large dimensionalities and therefore can be an attractive approach for large-scale datasets. However, the MAPwx does not quite solve the overfitting problem, so we would like to find a good compromise.

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

$$q(\boldsymbol{\theta}) = \prod_{i=1}^{d} q(m_i) \prod_{i=1}^{d} \prod_{k=1}^{c} q(w_{ik}) \prod_{c=1}^{k} \prod_{j=1}^{n} q(x_{kj}) \,. \tag{30}$$

instead of (28). Such posterior approximation was used, for example, in [30] for PCA in the presence of missing values. The implementation proposed there was based on the imputation algorithm and thus not easily scalable for high-dimensional sparse problems.

The fully factorial approximation (30) 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, w_k, w_m, which can be point-estimated.

The corresponding cost function (see Appendix E) can be minimized in different ways. For example, one can alternate between minimization w.r.t. *all* $\{\overline{w}_{ik}, \widetilde{w}_{ik}, \forall i, k\}$ and *all* $\{\overline{x}_{kj}, \widetilde{x}_{kj}, \forall k, j\}$ keeping the others fixed. This would be computationally heavy as it would require computation of matrices $\boldsymbol{\Phi}_i^{-1}$ and $\boldsymbol{\Psi}_j^{-1}$, similarly to VBPCA from Appendix D. Another possibility is to minimize the cost function w.r.t. each pair of parameters $\overline{w}_{ik}$, $\widetilde{w}_{ik}$ ($\overline{x}_{kj}$, $\widetilde{x}_{kj}$ respectively) keeping the rest of the parameters fixed. This approach is likely to result in slow convergence.

Minimization of the VB cost function in the fully factorial case (30) can be done efficiently using the gradient-based optimization scheme, as explained in Section 4.2. 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 $\{v, w_k, w_m, \forall k\}$. The required derivatives are reported in Appendix E. 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 $p(\mathbf{X}|\mathbf{Y}, \mathbf{W}, \mathbf{m}, v)$ can be approximated to be fully factorial on the E step. The approximation can be fitted to the true posterior by minimizing the cost function

$$C(\mathbf{W}, \mathbf{m}, v, q(\mathbf{X})) = \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{Y}, \mathbf{X}|\mathbf{W}, \mathbf{m}, v)} d\mathbf{X} \,. \tag{31}$$

Using the fully factorial $q(\mathbf{X})$ in (31) is motivated by the view of the EM algorithm presented in [21]. The resulting update rules are same as in VBPCAd (see Appendix E) with the exception that $w_k \to \infty$, $w_m \to \infty$, $\widetilde{w}_{ik} = 0$, and $\widetilde{m}_i = 0$. We refer to this approach as *PPCAd*.[6]

There is an interesting connection between the speed-up proposed in Section 4.2 and the fully diagonal posterior approximation. The second order

[6]The fully factorial approximation does not restrict the generality of the PPCA model in the case of complete data. Using 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 (see, e.g., [16]). Equation (17) tells that the posterior covariance $\boldsymbol{\Sigma}_{\mathbf{x},j}$ is the same for each j in the case of fully observed data. It is diagonal if and only if the columns of $\mathbf{W}$ are mutually orthogonal. This is requirement (3) of PCA. Requirement (4) is fulfilled because of the assumed prior model for $\mathbf{X}$ in PPCA. Thus the fully factorial approximation forces the PPCA model towards the PCA basis. The fully factorial approximation in the case with missing values generally introduces a bias in favor of solutions where the posterior covariances on the average are closer to being diagonal.

derivatives needed for the speed-up coincide with the inverse of the posterior variance of each parameter, and thus their computation is practically free.

5 EXPERIMENTS

5.1 Artificial data

The purpose of this section is to compare the performance of the different PCA algorithms discussed in the previous sections. We consider two characteristics: speed and accuracy in the task of missing value reconstruction. We also show advantages and drawbacks of the discussed approaches. The compared techniques include the least squares approaches as well as PPCA and Bayesian techniques. Table 2 summarizes different variants of the regularized (probabilistic) PCA models. The approaches differ on the posterior model for $\mathbf{X}$ and on the use of prior for $\mathbf{W}$.

Table 2: Variants of probabilistic approaches to PCA with missing values

	prior for $\mathbf{W}$		posterior for $\mathbf{W}$			posterior for $\mathbf{X}$		
	yes	no	point	diag	full	point	diag	full
PPCA		✓	✓					✓
PPCAd		✓	✓				✓	
VBPCA	✓				✓			✓
VBPCAd	✓			✓			✓	
MAPw	✓		✓					✓
MAPwx	✓		✓			✓		

In the experiments, the gradient-based (gr) optimization procedure with the speed-up proposed in Section 4.2 was used for MAPwx, least squares (LS), PPCAd and VBPCAd approaches. The control parameter was set $\alpha = 2/3$. The optimization which alternates (alt) between minimization w.r.t. $\mathbf{W}$ and $\mathbf{X}$ was used for the least squares, PPCA, MAPw, and VBPCA models. The imputation (imp) algorithm was also considered. The different algorithms are marked as (gr), (alt), and (imp) on the figures when necessary.

We generated data matrices $\mathbf{Y}$ according to the noisy PCA model (15) with $\mathbf{m} = 0$ and fixed dimensionalities d, c and n. The mixing matrix $\mathbf{W}$ was generated by taking a uniformly random orthogonal matrix and scaling its columns by $1, 2, \ldots, c$. Noise variance was set to $v = 0.01$.

Observed values were selected randomly in $\mathbf{Y}$, the rest of the data matrix was used for validation (to compute the test error). The ratio of observed values to the total number of elements in a data matrix

$$\beta = \frac{N}{dn} \tag{32}$$

was varied. We generated 30 data matrices for each β and ran different algorithms on all datasets in two experiments: 1) a typical situation when the number of samples n is much greater than the dimension d of the data vectors; 2) a difficult case when n is close to d.

The methods were compared using three quantities: the average training rms error, the average test rms error and the average time to convergence. The average values were computed across different datasets. 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.

The imputation algorithm was initialized with zeros for unobserved values while the other algorithms were initialized randomly. To study the effect of multiple local minima, we also initialized VB techniques using 100 iterations of the PPCAd algorithm (marked on the figures as "VBPCA (i)" and "VBPCAd (i)"). The number of estimated principal components was set to the number of components used to generate the data.

Experiment 1: Typical simple case

The number of samples n was set much larger than the dimensionality d of data vectors: $d = 10$, $c = 2$, $n = 1000$. The ratio of observed values β was varied in the interval $[0.2\ \ 0.9]$. The results of the experiments are presented in Fig. 4. Here we present some of our observations based on the obtained results:

- All the methods provide the same error with very little overfitting when the amount of missing values is relatively small. Therefore, the best strategy here is to use the fastest method, for example, the least squares approach.

- The *imputation algorithm* performs very well in the considered experiments: It converges relatively fast even for sparse data and it does not have serious problems with overfitting. Therefore, it can be used very efficiently for datasets with reasonable dimensionalities d and n. 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 datasets.

- Gradient-based and alternating LS algorithms work well when the amount of missing data is small. The quality of their performance deteriorates even for a decently small ratio of missing values: The algorithms easily get stuck at local minima with infinitely growing parameters $\mathbf{X}$ (similarly to the example from Section 2.1). Once arrived at a local minima, learning proceeds very slowly: prediction of only some observations is improved by using large parameter values. In such cases, the cost function was slowly but surely decreasing, but the probing error was growing to very large values (see Fig. 4). The algorithms might have been stopped before convergence.

- Approaches using variational approximations (VBPCA, VBPCAd, PPCAd) may suffer from the underfitting problem: The models may be overpenalized by setting some of the useful principal components to zero. The algorithms seem to find such local minima more often for sparse data (see, e.g., the larger test error for VBPCAd for $\beta = 0.2$). Special care has to be taken to avoid such solutions, for example, using flat prior for $\mathbf{W}$ at the beginning of VB learning typically helps.

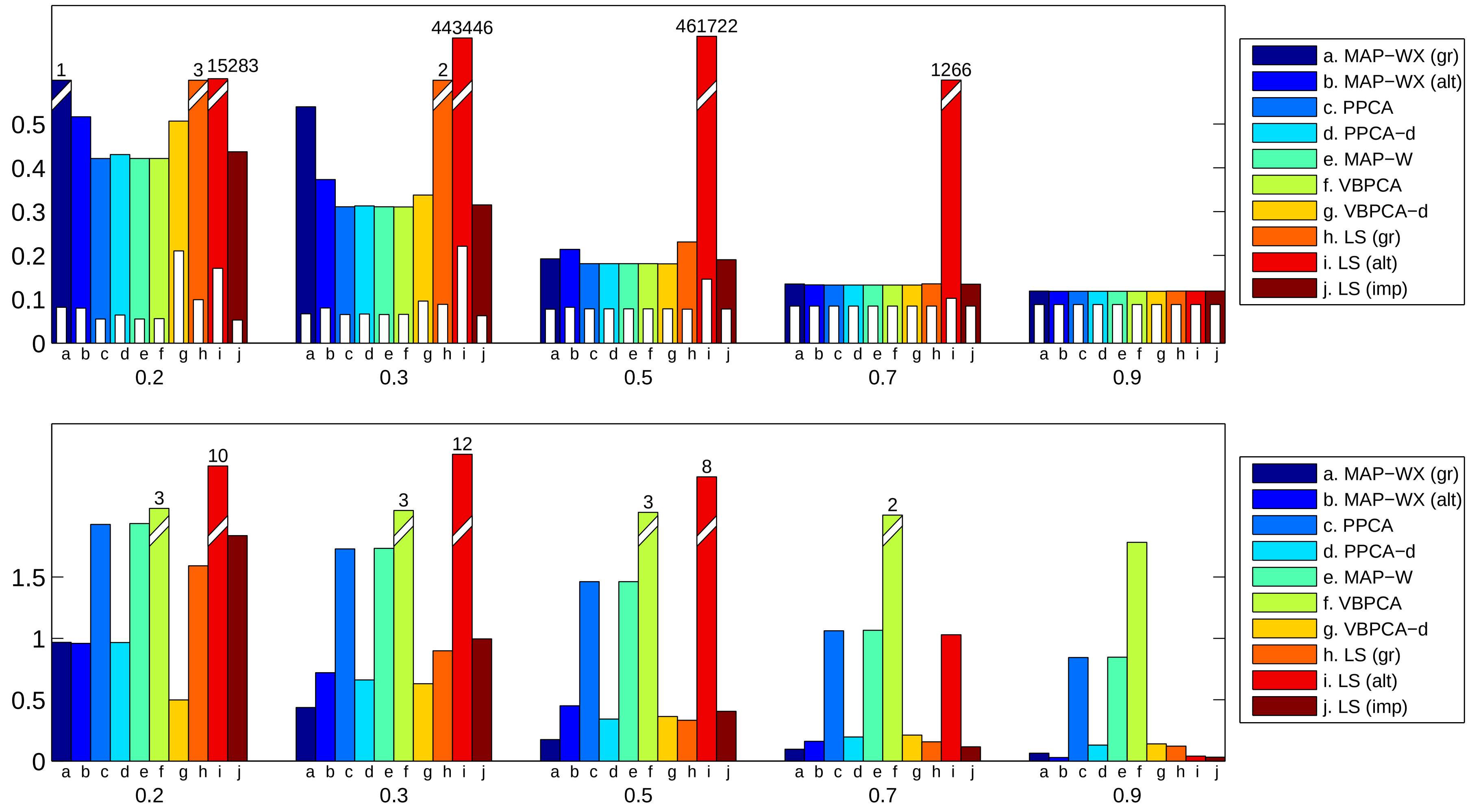

Figure 4: Experiment 1. The data are generated according to the noisy PCA model with $d = 10$, $c = 2$, $n = 1000$. All the algorithms use the correct number of components and they were initialized (except for the LS imputation) randomly. Top: Test rms error (colored bars) against the training rms error (thin white bars). The ratio β of observed values is shown under each group of bars. Bottom: The estimated time until convergence. The bars with too large values are shrunk and their corresponding values are shown on top of the bars.

We have used this in the presented experiments and most "bad" local were avoided. If this special care is taken, the accuracy of these three methods is similar to PPCA, MAPw and the imputation algorithm.

- The fastest algorithms with relatively little overfitting for sparse data are based on fully factorial variational approximations (PPCAd and VBPCAd). Their speed and generalization properties are very similar when $d \ll n$, if undesirable local minima (solutions with zero components) are avoided.

- PPCA and MAPw approaches have less problems with multiple local minima compared to the approaches using variational approximations. However, their convergence is quite slow compared to approaches based on point estimates for $\mathbf{X}$ (MAPwx) or fully factorial approximations (PPCAd, VBPCAd). The difference between PPCA and MAPw is very small in terms of the rms error. In general, MAPw can be more resistant to overfitting happening in separate rows of the data matrix (see the second example in Section 3.3). Both algorithms can hardly be used for very high-dimensional problems because of the heavy computations of the posterior covariance matrices.

- MAPwx approaches are fast and accurate when there is little missing data. The two variants of the algorithm optimize the same cost function but can get stuck at (different) local minima. This explains the difference of the test error for the two approaches for $\beta \leq 0.5$.

Experiment 2: Difficult case
The data dimensionalities in the second experiment were $d = 100$, $c = 10$, $n = 100$. The ratio of observed values β was varied in the interval [0.05 0.35]. The results of the experiments are presented in Fig. 5. Here we present some of our observations based on the obtained results:

- All the methods excepts for VB approaches severely overfit for very sparse data. For example, for $\beta = 0.05$ the training error is zero while the test error is large. The results provided for $\beta = 0.05$ by VBPCA and VBPCAd are more reliable: for example, VBPCA sets all the principal components to zero, yielding large (but similar) training and test errors.

- The MAPwx approach is not efficient in the case of sparse data because of the overfitting problem (see, e.g., results for $\beta = 0.15, 0.2$). Although large values of both parameters $\mathbf{W}$ and $\mathbf{X}$ are penalized in MAPwx models, the corresponding algorithms get easily stuck in local minima, overfitting to only part of the data.

- The imputation approach does not perform well either: It converges quite slowly and it also generalizes poorly to new data.

- Using more complex posterior models only for parameter $\mathbf{X}$ (in PPCA and PPCAd) slightly reduces the overfitting effect. A bit better generalization is achieved when a prior for $\mathbf{W}$ is also used (in MAPw).

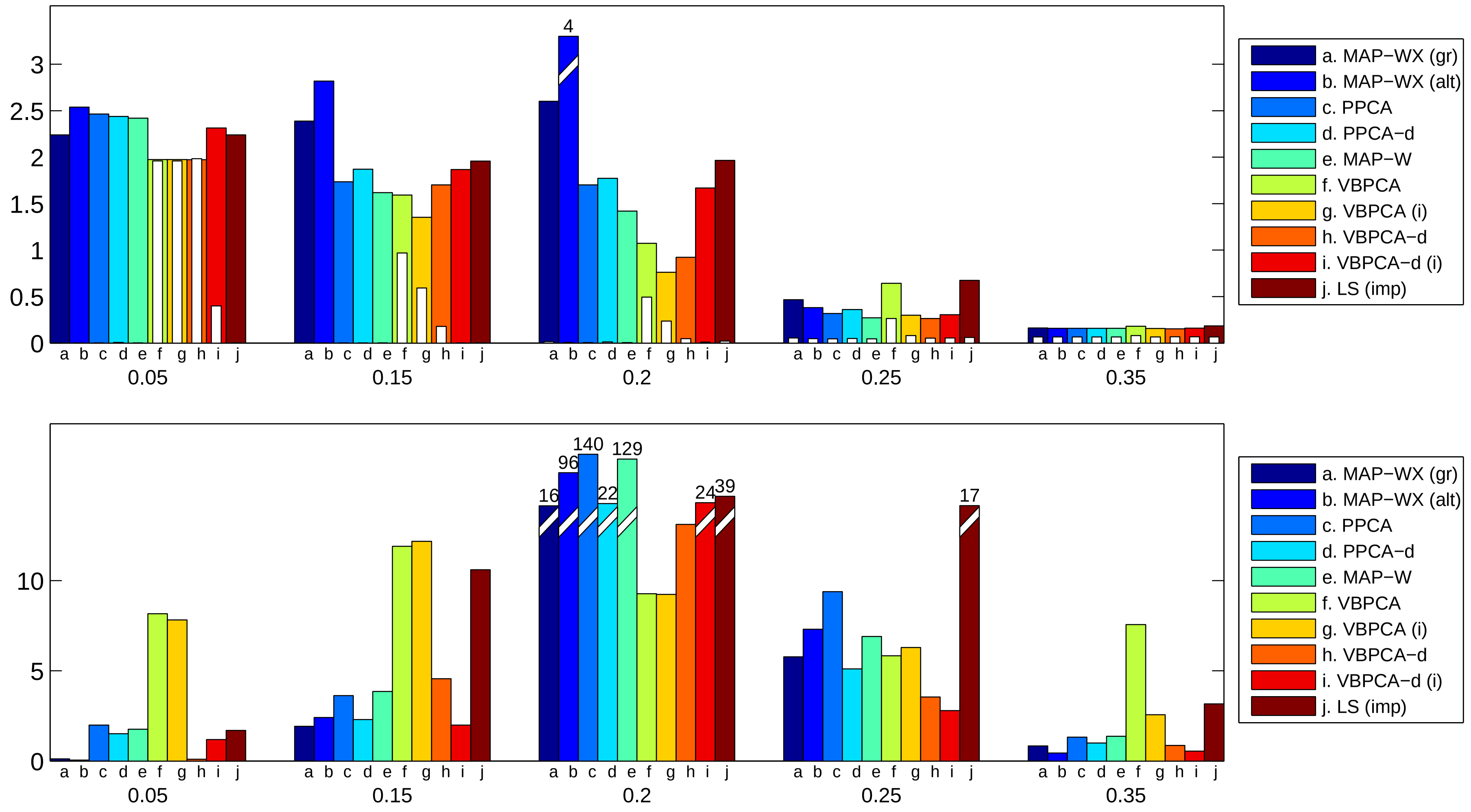

Figure 5: Experiment 2. The data are generated according to the noisy PCA model with $d = 100$, $c = 10$, $n = 100$. All the algorithms use the correct number of components and they were initialized (except for the LS imputation) randomly. Top: Test rms error (colored bars) against the training rms error (thin white bars). The ratio β of observed values is shown under each group of bars. Bottom: The estimated time until convergence. The bars with too large values are shrunk and their corresponding values are shown on top of the bars.

- For structured sparse data when $d \approx n$, the best performance is achieved by using the VB techniques. The advantage of VBPCA is largest for $\beta = 0.2$: with an initialization that decreases chances of underfitting, VBPCA converges fast and provides the best test error.

Note that most of the algorithms converge fast for very sparse datasets ($\beta = 0.05$) or relatively dense data ($\beta = 0.35$). These are the cases when overfitting is hard to avoid and when the learning problem is relatively simple, respectively. There is an increase of the time to convergence in the "transition phase" between the two extreme cases. The longest convergence time is observed for $\beta = 0.2$ when, interestingly, the number of estimated parameters is approximately equal to the number of data points.[7]

One of the reasons why VB approaches outperform the alternative methods is that they can cut off from the model principal components which cannot be estimated reliably. For example, none of the principal components was used in the trained VBPCA models for very sparse data ($\beta = 0.05$) while all 10 components were used for the datasets with $\beta = 0.35$. This result suggests that the optimal number of principal components which can be estimated for reliable reconstruction of missing data can be smaller than the "true" number of components underlying the data.

The downside of the automatic determination of the dimensionality of the principal subspace by VB methods is the existence of multiple local minima in the cost function: Any component set to zero is a potentially attractive solution for the method. In the presented experiments, we used two different initializations and the trained VBPCA models were different for most datasets. The good news is that the goodness of different local minima can be compared using the VB cost function values, which is the lower bound of the data likelihood, as shown in (29). Fig. 6 presents the test rms errors against the cost function values obtained for VBPCA and VBPCAd for the datasets with $\beta = 0.2$. The results show that a solution with a smaller cost function value generally provides a smaller test rms error both for VBPCA and VBPCAd. The correlations between the cost function values and the test rms errors are more evident for VBPCA than for VBPCAd because VBPCA gives a tighter lower bound for the data likelihood due to of a better approximation. Note also from Fig. 6 that the actual number of principal components used strongly affects the reconstruction quality.

The practical problem of finding the *global* minimum of the VB cost function is the need to run the learning algorithm many times. It is also advisable to avoid "bad" local solutions, for example, by proper initialization. The required computations for VBPCA can be quite heavy, especially for large-scale datasets. In such cases, VBPCAd, as a "light" version of VBPCA, might be more efficient.

5.2 Experiments with Netflix data

We have tested different PCA approaches in the presence of missing values in the task of collaborative filtering. The Netflix problem [22] is a collaborative

[7]The moment of convergence was estimated as the time instant after which the changes of the training rms error were negligibly small, although the final stages of learning had very little effect on the test error.

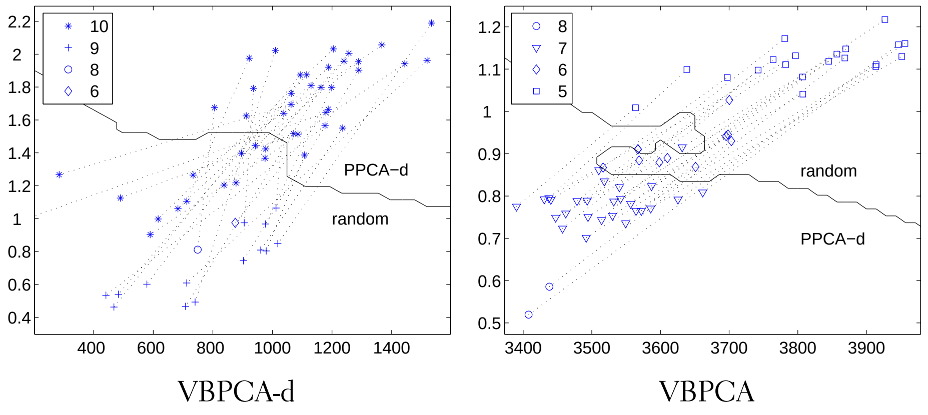

Figure 6: Experiment with $d = 100$, $c = 10$, $n = 100$ and the ratio of observed values $\beta = 0.2$. Test rms error (y-axes) against cost function values (x-axes) obtained with VBPCAd (left) and VBPCA (right). The solid line separates solutions obtained for two different initializations: random and with 100 iterations of PPCAd. The dotted lines connect two local minima for the same dataset. The plots show the values after convergence. Different markers represent the number of components effectively used in the corresponding model.

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.

The PCA approach is one of the most popular techniques considered by Netflix contestants (see, e.g., [10, 32, 25, 19]). In our recent conference papers [27, 28], we tried to estimate $c = 15$ principal components from the data using unregularized (LS) PCA, MAPwx, and VBPCAd approaches. The main conclusion was that when data are very sparse, overfitting becomes a serious problem and the imputation algorithm becomes very slow. The proposed remedy was to use, for example, VBPCAd instead.

Fig. 7 presents results with MAPwx and VBPCAd using a varying number of principal components. MAPwx starts to overfit after about 5 hours of computation, that is, the root mean square (rms) error on the test data (upper 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 after the early fluctuation. 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. All experiments were run on a dual cpu AMD Opteron SE 2220 using Matlab.

Our best rms error for probing Netflix data was 0.9055 achieved with VBPCAd with 50 components. This is a good result compared to other implementations of similar PCA models reported by other contestants (see, e.g., [2, 19]). The results suggest that the PCA model is too simple to solve the Netflix problem: The model captures global similarities while local correlations should also be taken into account to achieve better performance [2].

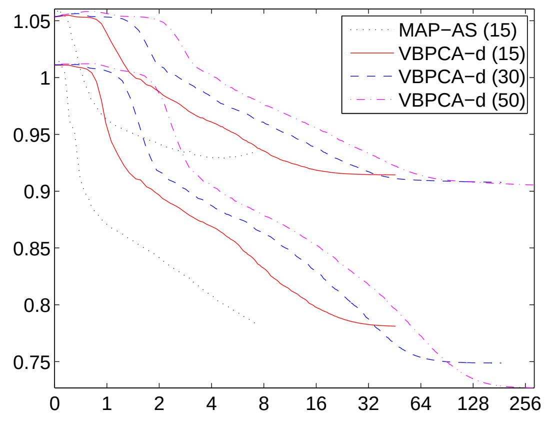

Figure 7: Experiments with the Netflix data. The rms error (y-axis) is plotted against the processor time in hours. The lower curves show the training error while the upper curves show the test error (for the probing data provided by Netflix). The time scale is linear from 0 to 1 and logarithmic above 1.

6 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 bias term can no longer be removed as a preprocessing step, since the optimal bias depends on the other parameters. Secondly, the estimation of the covariance matrix of the data becomes difficult and thus the solution by eigen-decomposition cannot be used directly. Thirdly, 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 the projections of the data to the principal subspace. Regularization can be done elegantly using probabilistic models.

The relevant probabilistic models are generally more complex compared to the complete data case. For example, the posterior covariance of principal components $\mathbf{\Sigma}_{\mathbf{x},j}$ is no longer same for each sample. Thus, a diagonal posterior covariance $\mathbf{\Sigma}_{\mathbf{x},j}$ is no longer sufficient for finding the optimal PCA solution. Regularized models typically have more local minima. In particular, there are local minima of the cost function when some of the hyperparameters go to zero.

In the paper, we have considered only some possible approaches to overcome the overfitting problem. Other possible alternatives include, for example, the evidence framework (see, e.g., [4]), Tikhonov regularization (which is closely related to MAPwx) and early stopping. For early stopping, the data are divided into training and validation sets. Both the training and validation errors are estimated as the error of reconstruction of missing values from the principal components. Learning is stopped when the validation error starts growing due to overlearning. We have tried early stopping in the experiments with artificial data (similar to the ones presented in Section 5.1) but the performance provided by the probabilistic methods considered in this article was generally better.

Large-scale PCA problems require fast converging learning algorithms which allow for efficient implementation. Additionally, sparse datasets require that the amount of computer memory would scale linearly with the number of observed values regardless of the data matrix dimensionalities. In the 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 [28].

The PCA approaches considered in this article can be used in different tasks. We summarized our recommendations based on the experimental results from Section 5 in Fig. 8.

$\beta \approx 1$:

small d, n, c	large d, n, c
imputation	LS gradient

average β:

	small c	large c
$d \ll n$	PPCA	PPCAd
$d \approx n$	VBPCA	VBPCAd

small β:

	small c	large c
$d \ll n$	MAPw	VBPCAd
$d \approx n$	VBPCA	VBPCAd

Figure 8: Recommendations on the use of PCA methods for different degrees of sparsity β defined as in (32).

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 datasets, 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. 7 have been 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 (Alexander Ilin's postdoctoral project), and the IST Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views. We would like to thank Netflix [22] for providing the data.

REFERENCES

[1] 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.

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

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

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

[5] A. Christoffersson. *The one component model with incomplete data*. PhD thesis, Uppsala University, 1970.

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

[7] 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, Feb. 1981.

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

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

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

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

[12] S. Haykin. *Modern Filters*. Macmillan, 1989.

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

[14] 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.

[15] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. J. Wiley, 2001.

[16] 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.

[17] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[18] 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.

[19] 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, August 2007.

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

[21] 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.

[22] Netflix. Netflix prize webpage, 2007. `http://www.netflixprize.com/`.

[23] 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.

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

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

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

[27] 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. Springer-Verlag, September 2007.

[28] 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)*, Kitakyushu, Japan, November 2007. To appear.

[29] T. Raiko and H. Valpola. Missing values in nonlinear factor analysis. In *Proc. of the 8th Int. Conf. on Neural Information Processing (ICONIP'01)*, pages 822–827, Shanghai, 2001.

[30] 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(Jan):155–201, January 2007.

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

[32] R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the International Conference on Machine Learning*, 2007.

[33] 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.

[34] 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.

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

A NOTES ON THE IMPUTATION ALGORITHM

The cost function minimized by the imputation algorithm

The imputation algorithm is shown to belong to the class of least squares PCA algorithms, that is that it minimizes the cost function (6). Performing SVD on the complete data matrix is equivalent to finding a low-rank approximation of $\mathbf{Y}$ which minimizes the cost function (2). Let us rewrite (2) as:

$$C_{\text{ia}} = C_{\text{obs}} + C_{\text{mis}}\,, \tag{33}$$

$$C_{\text{obs}} = \sum_{ij \in O} \left(y_{ij} - \hat{y}_{ij}(\boldsymbol{\theta})\right)^2, \qquad C_{\text{mis}} = \sum_{ij \notin O} \left(\overline{y}_{ij} - \hat{y}_{ij}(\boldsymbol{\theta})\right)^2, \tag{34}$$

where $\boldsymbol{\theta} = (\mathbf{W}, \mathbf{X}, \mathbf{m})$ is the model parameters, $\hat{y}_{ij}(\boldsymbol{\theta})$ are the reconstructions obtained with $\boldsymbol{\theta}$ as in (7), and $\overline{y} = \hat{y}_{ij}(\boldsymbol{\theta}_{\text{prev}})$ are the imputed missing values computed using $\boldsymbol{\theta}$ from the previous iteration (denoted by $\boldsymbol{\theta}_{\text{prev}}$).

The imputation algorithm minimizes (33) as a function $C_{\text{ia}}(Y_{\text{imp}}, \boldsymbol{\theta})$ alternately w.r.t. the imputed values $Y_{\text{imp}} = \{\overline{y}_{ij}\,,\, ij \notin O\}$ and the model parameters $\boldsymbol{\theta}$. Let Y^*_{imp} be the minimizer of $C_{\text{ia}}(Y_{\text{imp}}, \boldsymbol{\theta})$ w.r.t. Y_{imp} for a fixed $\boldsymbol{\theta}$:

$$Y^*_{\text{imp}} = Y_{\text{imp}}(\boldsymbol{\theta}) = \arg\min_{Y_{\text{imp}}} C_{\text{ia}}(Y_{\text{imp}}, \boldsymbol{\theta})\,. \tag{35}$$

This naturally implies that $C_{\text{mis}}(Y^*_{\text{imp}}, \boldsymbol{\theta}) = 0$ and therefore

$$C_{\text{obs}}(\boldsymbol{\theta}) = C_{\text{ia}}(Y^*_{\text{imp}}, \boldsymbol{\theta})\,. \tag{36}$$

Taking into account (36), we get

$$\frac{dC_{\text{obs}}}{d\boldsymbol{\theta}} = \left.\frac{dC_{\text{ia}}}{d\boldsymbol{\theta}}\right|_{Y^*_{\text{imp}}} = \left.\frac{\partial C_{\text{ia}}}{\partial \boldsymbol{\theta}}\right|_{Y^*_{\text{imp}}} + \left.\frac{\partial C_{\text{ia}}}{\partial Y_{\text{imp}}}\right|_{Y^*_{\text{imp}}} \frac{\partial Y_{\text{imp}}}{\partial \boldsymbol{\theta}} = \left.\frac{\partial C_{\text{ia}}}{\partial \boldsymbol{\theta}}\right|_{Y^*_{\text{imp}}}, \tag{37}$$

where $\partial C_{\text{ia}}/\partial Y_{\text{imp}} = 0$ because of (35).

The cost (33) is minimized w.r.t. $\boldsymbol{\theta}$ only when $C_{\text{ia}}(\boldsymbol{\theta}) = C_{\text{ia}}(Y^*_{\text{imp}}, \boldsymbol{\theta}) = C_{\text{obs}}(\boldsymbol{\theta})$, which together with (37) suggests that the minima of $C_{\text{ia}}(Y_{\text{imp}}, \boldsymbol{\theta})$ coincide with the minima of $C_{\text{obs}}(\boldsymbol{\theta})$ which is the cost function (6).

EM interpretation of the imputation algorithm

The imputation algorithm can be seen as the application of the EM steps to the model

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

with isotropic Gaussian noise $p(\boldsymbol{\epsilon}_j) = \mathcal{N}(\boldsymbol{\epsilon}_j; 0, v\mathbf{I})$ and the model parameters $\mathbf{W}$, $\mathbf{x}_j$, $\mathbf{m}$, v. Note that $\mathbf{x}_j$ belong to the model parameters and the missing values are treated as *hidden* variables. This is contrary to the probabilistic PCA model in [33, 31] (see Section 3.1) where $\mathbf{x}_j$ are treated as hidden variables.

Let us consider the t-th iteration of the imputation algorithm denoting the set of missing values by $Y_{\text{mv}} = \{y_{ij} \mid ij \notin O\}$ and the parameter values found at the t-th iteration as $\boldsymbol{\theta}^t = (\mathbf{W}, \mathbf{X}, \mathbf{m}, v)^t$. The algorithm alternates between two steps:

1. E-step. The missing values Y_{mv} are updated as $\overline{y}_{ij} = \sum_{k=1}^{c} w^t_{ik} x^t_{kj} + m^t_i$, which is equivalent to computing the mean of the posterior distribution $p(Y_{\text{mv}}|\boldsymbol{\theta}^t)$.

2. M-step. The expected complete-data log-likelihood:

$$\int \log p(\mathbf{Y}|\boldsymbol{\theta}) p(Y_{\text{mv}}|\boldsymbol{\theta}^t)\, dY_{\text{mv}} = \langle \log p(\mathbf{Y}|\boldsymbol{\theta}) \rangle_{\boldsymbol{\theta}^t} =$$
$$= -\frac{dn}{2} \log 2\pi v - \frac{1}{v} \sum_{ij \in O} \left(y_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i \right)^2$$
$$-\frac{1}{v} \sum_{ij \notin O} \left[\left(\overline{y}_{ij} - \mathbf{w}_i^{\mathrm{T}} \mathbf{x}_j - m_i \right)^2 + v^t \right],$$

   is minimized w.r.t. $\mathbf{W}$, $\mathbf{X}$, $\mathbf{m}$ by performing PCA on the completed matrix $\mathbf{Y}$.

B ROTATION OF THE PCA BASIS FOR PROBABILISTIC PCA

The variational view of the EM algorithm [21] allows for an interpretation of the PPCA learning algorithm in which the cost function

$$C(\mathbf{W}, \mathbf{m}, v, q(\mathbf{X})) = \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{Y}, \mathbf{X}|\mathbf{W}, \mathbf{m}, v)} d\mathbf{X} \tag{38}$$

is minimized w.r.t. to the model parameters $\mathbf{W}$, $\mathbf{m}$, v 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 assuming that $q(\mathbf{X})$ is fixed.

The cost function (38) can be rewritten as:

$$C = \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{X})} d\mathbf{X} - \int q(\mathbf{X}) \log p(\mathbf{Y}|\mathbf{W}, \mathbf{X}, \mathbf{m}, v) d\mathbf{X} . \tag{39}$$

Now we note that one can always transform linearly the columns of $\mathbf{X}$ and compensate it by a proper transformation of parameters $\mathbf{W}$ and $\mathbf{m}$ without changing the second term in (39). For example, subtraction of a constant $\mathbf{m}_x$ from $\mathbf{x}$ can be compensated by changing the bias term $\mathbf{m}$ correspondingly:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{m} = \mathbf{W}(\mathbf{x} - \mathbf{m}_x) + \mathbf{W}\mathbf{m}_x + \mathbf{m} = \mathbf{W}\mathbf{x}_1 + \mathbf{m}_1 . \tag{40}$$

Similarly, rotation of the columns of $\mathbf{X}$ can be compensated by changing $\mathbf{W}$:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{m} = (\mathbf{W}\mathbf{A}^{-1})(\mathbf{A}\mathbf{x}) + \mathbf{m} = \mathbf{W}_2\mathbf{x}_2 + \mathbf{m} . \tag{41}$$

Such transformations generally affect the first term in (39) and at the convergence this term should be minimized w.r.t. any linear transformations. Note that the first term in (39) is the Kullback-Leibler divergence between $q(\mathbf{X})$ and $p(\mathbf{X})$:

$$\begin{aligned} D(q(\mathbf{X})||p(\mathbf{X})) &= \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{X})} d\mathbf{X} = \sum_j \int q(\mathbf{x}_j) \log \frac{q(\mathbf{x}_j)}{p(\mathbf{x}_j)} d\mathbf{x}_j \\ &= \frac{1}{2}\Big[- \log \det \prod_j \mathbf{\Sigma}_{\mathbf{x},j} + \sum_j \operatorname{tr}(\mathbf{\Sigma}_{\mathbf{x},j}) + \sum_j \overline{\mathbf{x}}_j^{\mathrm{T}} \overline{\mathbf{x}}_j \Big] . \end{aligned} \tag{42}$$

Therefore we seek the transformation that keeps the explanation of the observed variables equally good but makes the latent variables as close to the prior model as possible.

The transformation (40) changes (42) as

$$\begin{aligned} D(q_{\mathbf{m}}||p) = \frac{1}{2}\Big[- \log \det \prod_j \mathbf{\Sigma}_{\mathbf{x},j} &+ \sum_j \operatorname{tr}(\mathbf{\Sigma}_{\mathbf{x},j}) \\ &+ \sum_j (\overline{\mathbf{x}}_j - \mathbf{m}_x)^{\mathrm{T}} (\overline{\mathbf{x}}_j - \mathbf{m}_x) \Big] . \end{aligned} \tag{43}$$

The optimal $\mathbf{m}_x$ can be found by taking the derivative of (43) and equating it to zero, which yields

$$\mathbf{m}_x = \frac{1}{n} \sum_{j=1}^{n} \overline{\mathbf{x}}_j .$$

Similarly, the transformation (41) yields

$$D(q_{\mathbf{W}}||p) = \frac{1}{2}\Big[- \log \det \prod_j \mathbf{A}\mathbf{\Sigma}_{\mathbf{x},j}\mathbf{A}^{\mathrm{T}} + n \operatorname{tr}(\mathbf{A}\mathbf{\Sigma}_* \mathbf{A}^{\mathrm{T}}) \Big] , \tag{44}$$

where

$$\mathbf{\Sigma}_* = \frac{1}{n}\Big(\overline{\mathbf{X}}\overline{\mathbf{X}}^{\mathrm{T}} + \sum_j \mathbf{\Sigma}_{\mathbf{x},j} \Big) . \tag{45}$$

Taking the derivative of (44) w.r.t. $\mathbf{A}$ yields

$$-n(\mathbf{A}^{\mathrm{T}})^{-1} + \mathbf{A}\mathbf{\Sigma}_* = 0$$

and therefore the optimal $\mathbf{A}$ satisfies

$$\mathbf{A}\mathbf{\Sigma}_*\mathbf{A}^{\mathrm{T}} = \mathbf{I}\,. \tag{46}$$

Hence, the optimal linear transformation of $\mathbf{X}$ is done by centralization of the posterior means $\overline{\mathbf{x}}_j$ and diagonalization of the matrix in (45). This transformation resembles normalization and so called whitening (see, e.g., [15]) of the posterior distributions of the latent variables. At the convergence, no linear transformation can decrease the value of the cost function (38), which implies that $\frac{1}{n}\sum_{j=1}^{n}\overline{\mathbf{x}}_j = 0$ and $\mathbf{\Sigma}_* = \mathbf{I}$.

The PPCA solution has a rotational ambiguity: Using any linear transformation (41) with an *orthogonal* matrix $\mathbf{A}$ does not affect (46) and therefore the corresponding solution is equally good w.r.t. the cost function values. We can use this fact to define the PCA rotation within the principal subspace (see Section 3.2).

C MAPWX APPROACH TO PCA

The linear observation model $\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{m} + \boldsymbol{\epsilon}$ is accompanied with the following Gaussian priors:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}; 0, \mathbf{I}) \qquad p(\boldsymbol{\epsilon}) = \mathcal{N}(\boldsymbol{\epsilon}; 0, v\mathbf{I})$$

$$p(\mathbf{m}) = \mathcal{N}(\mathbf{m}; 0, w_m\mathbf{I}) \qquad p(\mathbf{W}) = \prod_{k=1}^{c}\mathcal{N}(\mathbf{W}_{:k}; 0, w_k\mathbf{I})$$

The cost function is

$$C_{\mathrm{MAP}} = \sum_{ij\in O}\left(\frac{1}{v}(y_{ij} - \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j - m_i)^2 + \log 2\pi v\right) + \frac{1}{w_m}\|\mathbf{m}\|^2 + d\log 2\pi w_m$$
$$+ \sum_{k=1}^{c}\left(\frac{1}{w_k}\|\mathbf{W}_{:k}\|^2 + d\log 2\pi w_k + \|\mathbf{X}_{k:}\|^2 + n\log 2\pi\right)\,.$$

The alternating optimization procedure can be implemented as follows:

$$\mathbf{x}_j = (\mathbf{W}_j^{\mathrm{T}}\mathbf{W}_j + v\mathbf{I})^{-1}\mathbf{W}_j^{\mathrm{T}}(\mathring{\mathbf{Y}}_{:j} - \mathbf{M}_j)\,, \qquad j = 1, \ldots, n\,,$$

$$m_i = \frac{w_m}{|O_i|(w_m + v/|O_i|)}\sum_{j\in O_i}\left[y_{ij} - \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_{:j}\right]\,, \qquad i = 1, \ldots, d\,,$$

$$\mathbf{w}_i^{\mathrm{T}} = (\mathring{\mathbf{Y}}_{i:} - m_i)^{\mathrm{T}}\mathbf{X}_i^{\mathrm{T}}(\mathbf{X}_i\mathbf{X}_i^{\mathrm{T}} + v\,\mathrm{diag}(w_k^{-1}))^{-1}\,, \qquad i = 1, \ldots, d\,,$$

$$v = \frac{1}{N}\sum_{ij\in O}(y_{ij} - \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j - m_i)^2\,, \qquad w_k = \frac{1}{d}\sum_{i=1}^{d} w_{ik}^2\,, \qquad w_m = \frac{1}{d}\sum_{i=1}^{d} m_i^2\,.$$

Gradient-based learning discussed in Section 4.2 requires:

$$\frac{\partial C_{\mathrm{MAP}}}{\partial w_{ik}} = \frac{w_{ik}}{w_k} + \frac{1}{v}\sum_{j\in O_i} -\Big(y_{ij} - \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j - m_i\Big)x_{kj}\,,$$

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

$$\frac{\partial^2 C_{\mathrm{MAP}}}{\partial w_{ik}^2} = \frac{1}{w_k} + \frac{1}{v}\sum_{j\in O_i} x_{kj}^2\,, \qquad \frac{\partial^2 C_{\mathrm{MAP}}}{\partial x_{kj}^2} = 1 + \frac{1}{v}\sum_{i\in O_j} w_{ik}^2\,.$$

D VARIATIONAL BAYESIAN PCA (VBPCA)

The cost function of VBPCA

$$\begin{aligned} C_{\mathrm{vb}} &= \int q(\boldsymbol{\theta}) \log \frac{q(\boldsymbol{\theta})}{p(\mathbf{Y},\boldsymbol{\theta}|v,w_k,w_m)} d\boldsymbol{\theta} \\ &= \left\langle \frac{1}{2w_m}\mathbf{m}^{\mathrm{T}}\mathbf{m} + \frac{d}{2}\log 2\pi w_m \right\rangle \\ &+ \sum_{i=1}^{d}\left\langle \frac{1}{2}\mathbf{w}_i^{\mathrm{T}}\operatorname{diag}(w_k^{-1})\mathbf{w}_i + \frac{1}{2}\sum_{k=1}^{c}\log 2\pi w_k \right\rangle + \sum_{j=1}^{n}\left\langle \frac{1}{2}\mathbf{x}_j^{\mathrm{T}}\mathbf{x}_j + \frac{c}{2}\log 2\pi \right\rangle \\ &+ \sum_{ij\in O}\left\langle \frac{1}{2v}(y_{ij} - m_i - \mathbf{w}_i^{\mathrm{T}}\mathbf{x}_j)^2 + \frac{1}{2}\log 2\pi v \right\rangle + \langle \log q(\boldsymbol{\theta})\rangle \end{aligned} \qquad (47)$$

is minimized w.r.t. the approximating pdf $q(\boldsymbol{\theta})$ and model hyperparameters v, w_k, w_m. Here we use the notation $\boldsymbol{\theta} = (\mathbf{W},\mathbf{X},\mathbf{m})$ and $\langle f(\boldsymbol{\theta})\rangle = \int f(\boldsymbol{\theta})q(\boldsymbol{\theta})d\boldsymbol{\theta}$.

Using the following form of q

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

allows for the following optimization procedure. The update of the principal components:

$$\overline{\mathbf{x}}_j = \boldsymbol{\Psi}_j^{-1}\overline{\mathbf{W}}_j^{\mathrm{T}}(\mathring{\mathbf{Y}}_{:j} - \overline{\mathbf{M}}_j)\,, \qquad \boldsymbol{\Sigma}_{\mathbf{x},j} = v\boldsymbol{\Psi}_j^{-1}$$

$$\boldsymbol{\Psi}_j = \overline{\mathbf{W}}_j^{\mathrm{T}}\overline{\mathbf{W}}_j + \sum_{i\in O_j}\boldsymbol{\Sigma}_{\mathbf{w},i} + v\mathbf{I}\,, \qquad j = 1,\ldots,n\,.$$

The update of the bias term:

$$\overline{m}_i = \frac{w_m}{|O_i|(w_m + v/|O_i|)}\sum_{j\in O_i}[y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}}\overline{\mathbf{x}}_j] \qquad (48)$$

$$\widetilde{m}_i = \frac{v w_m}{|O_i|(w_m + v/|O_i|)}\,, \qquad i = 1,\ldots,d\,. \qquad (49)$$

The update of matrix $\mathbf{W}$:

$$\overline{\mathbf{w}}_i^{\mathrm{T}} = (\mathring{\mathbf{Y}}_{i:} - \overline{m}_i)^{\mathrm{T}}\overline{\mathbf{X}}_i^{\mathrm{T}}\boldsymbol{\Phi}_i^{-1}\,, \qquad \boldsymbol{\Sigma}_{\mathbf{w},i} = v\boldsymbol{\Phi}_i^{-1}$$

$$\boldsymbol{\Phi}_i = \overline{\mathbf{X}}_i\overline{\mathbf{X}}_i^{\mathrm{T}} + \sum_{j\in O_i}\boldsymbol{\Sigma}_{\mathbf{x},j} + v\operatorname{diag}(w_k^{-1})\,, \qquad i = 1,\ldots,d$$

and the variance parameters:

$$v = \frac{1}{N} \sum_{ij \in O} (y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}} \overline{\mathbf{x}}_j - \overline{m}_i)^2 + \frac{1}{N} \sum_{ij \in O} [\widetilde{m}_i + \overline{\mathbf{w}}_i^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{x},j} \overline{\mathbf{w}}_i + \overline{\mathbf{x}}_j^{\mathrm{T}} \boldsymbol{\Sigma}_{\mathbf{w},i} \overline{\mathbf{x}}_j + \mathrm{tr}(\boldsymbol{\Sigma}_{\mathbf{x},j} \boldsymbol{\Sigma}_{\mathbf{w},i})] ,$$

$$w_k = \frac{1}{d} \sum_{i=1}^{d} \overline{w}_{ik}^2 + \widetilde{w}_{ik} , \qquad w_m = \frac{1}{d} \sum_{i=1}^{d} \overline{m}_i^2 + \widetilde{m}_i , \tag{50}$$

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

In analogy to PPCA (see Appendix B), one can transform a VBPCA solution to fit the prior model better. The requirements (23)–(24) are then supplemented by

$$\boldsymbol{\Sigma}_{\mathbf{W}} = \overline{\mathbf{W}}^{\mathrm{T}} \overline{\mathbf{W}} + \sum_{i=1}^{d} \boldsymbol{\Sigma}_{\mathbf{w},i} = \mathrm{diag}(\lambda_i) , \qquad \lambda_i \geq \lambda_k , \quad i < k .$$

The pdf of a missing value y_{ij}, $ij \notin O$ can be approximated as a Gaussian with the mean and variance parameters defined as

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

E VB PCA WITH FULLY FACTORIAL APPROXIMATION (VBPCAD)

The VBPCAd approach uses the fully factorial posterior approximation

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

Then, the cost function (47) splits into the sum of simple terms:

$$C_{\mathrm{vb}} = \sum_{ij \in O} C_{xij} + \sum_{i=1}^{d} C_{bi} + \sum_{i=1}^{d} \sum_{k=1}^{c} C_{aik} + \sum_{k=1}^{c} \sum_{j=1}^{n} C_{skj} ,$$

where the individual terms are

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

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

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

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

Variational parameters $\overline{m}_i$, $\widetilde{m}_i$, w_k and w_m are updated to minimize the cost function using the update rules in (48)–(50). Parameters $\widetilde{w}$, $\widetilde{x}$ can be updated to minimize the cost function:

$$\widetilde{w}_{ik} = v \left[\frac{v}{w_k} + \sum_{j \in O_i} \overline{x}_{kj}^2 + \widetilde{x}_{kj} \right]^{-1}, \tag{51}$$

$$\widetilde{x}_{kj} = v \left[v + \sum_{i \in O_j} \overline{w}_{ik}^2 + \widetilde{w}_{ik} \right]^{-1}. \tag{52}$$

The update rule for v is

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

Derivatives required for gradient-based optimization:

$$\frac{\partial C_{\mathrm{vb}}}{\partial \overline{w}_{ik}} = \frac{\overline{w}_{ik}}{w_k} + \frac{1}{v} \sum_{j \in O_i} - \Big(y_{ij} - \overline{\mathbf{w}}_i^{\mathrm{T}} \overline{\mathbf{x}}_j - \overline{m}_i \Big) \overline{x}_{kj} + \overline{w}_{ik} \widetilde{x}_{kj} \,,$$

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

The second-order derivatives which can be used to speed up learning, as explained in Section 4.2, coincide with the inverse of the updated variances given in (51)–(52): $\partial^2 C_{\mathrm{vb}} / \partial \overline{w}_{ik}^2 = \widetilde{w}_{ik}^{-1}$ and $\partial^2 C_{\mathrm{vb}} / \partial \overline{x}_{kj}^2 = \widetilde{x}_{kj}^{-1}$.

TKK REPORTS IN INFORMATION AND COMPUTER SCIENCE

ISBN 978-951-22-9481-7 (Print)
ISBN 978-951-22-9482-4 (Online)
ISSN 1797-5034 (Print)
ISSN 1797-5042 (Online)