Principal Component Analysis for Large Scale Problems with Lots of Missing Values

Tapani Raiko, Alexander Ilin, and Juha Karhunen

Adaptive Informatics Research Center, Helsinki Univ. of Technology
P.O. Box 5400, FI-02015 TKK, Finland
{Tapani.Raiko, Alexander.Ilin, Juha.Karhunen}@tkk.fi
http://www.cis.hut.fi/projects/bayes/

Abstract. Principal component analysis (PCA) is a well-known classical data analysis technique. There are a number of algorithms for solving the problem, some scaling better than others to problems with high dimensionality. They also differ in their ability to handle missing values in the data. We study a case where the data are high-dimensional and a majority of the values are missing. In case of very sparse data, overfitting becomes a severe problem even in simple linear models such as PCA. We propose an algorithm based on speeding up a simple principal subspace rule, and extend it to use regularization and variational Bayesian (VB) learning. The experiments with Netflix data confirm that the proposed algorithm is much faster than any of the compared methods, and that VB-PCA method provides more accurate predictions for new data than traditional PCA or regularized PCA.

1 Introduction

Principal component analysis (PCA) [1–3] is a classic technique in data analysis. It can be used for compressing higher dimensional data sets to lower dimensional ones for data analysis, visualization, feature extraction, or data compression.

PCA can be derived from a number of starting points and optimization criteria [3, 4, 2]. 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 [5].

In this paper, we study PCA in the case that most of the data values are missing (or unknown). Common algorithms for solving PCA prove to be inadequate in this case, and we thus propose a new algorithm. The problem of overfitting is also studied and solutions given.

We make the typical assumption that values are missing at random, 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.

2 Principal Component Analysis

Assume that we have n d-dimensional data vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$, which form the $d \times n$ data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$. The matrix $\mathbf{X}$ is decomposed into $\mathbf{X} \approx \mathbf{AS}$, where $\mathbf{A}$ is a $d \times c$ matrix, $\mathbf{S}$ is a $c \times n$ matrix and $c \leq d \leq n$. Principal subspace methods [6, 4] find such $\mathbf{A}$ and $\mathbf{S}$ that the reconstruction error

$$C = \|\mathbf{X} - \mathbf{AS}\|_F^2 = \sum_{i=1}^{d} \sum_{j=1}^{n} (x_{ij} - \sum_{k=1}^{c} a_{ik} s_{kj})^2 , \tag{1}$$

is minimized. Typically the row-wise mean is removed from $\mathbf{X}$ as a preprocessing step. Without any further constraints, there exist infinitely many ways to perform such a decomposition. PCA constraints the solution by further requiring that the column vectors of $\mathbf{A}$ are of unit norm and mutually orthogonal and the row vectors of $\mathbf{S}$ are also mutually orthogonal [3, 4, 2, 5].

There are many ways to solve PCA [6, 4, 2]. We will concentrate on the subspace learning algorithm that can be easily adapted for the case of missing values and further extended.

Subspace Learning Algorithm works by applying gradient descent to the reconstruction error (1) directly yielding the update rules

$$\mathbf{A} \leftarrow \mathbf{A} + \gamma(\mathbf{X} - \mathbf{AS})\mathbf{S}^T , \qquad \mathbf{S} \leftarrow \mathbf{S} + \gamma \mathbf{A}^T(\mathbf{X} - \mathbf{AS}) . \tag{2}$$

Note that with $\mathbf{S} = \mathbf{A}^T\mathbf{X}$ the update rule for $\mathbf{A}$ is a batch version of Oja's subspace rule [7]. The algorithm finds a basis in the subspace of the largest principal components. If needed, the end result can be transformed into the PCA solution by proper orthogonalization of $\mathbf{A}$ and $\mathbf{S}$.

3 Principal Component Analysis with Missing Values

Let us consider the same problem when the data matrix has missing entries. We would like to find $\mathbf{A}$ and $\mathbf{S}$ such that $\mathbf{X} \approx \mathbf{AS}$ for the observed data samples. The rest of the product $\mathbf{AS}$ represents the reconstruction of missing values.

The subspace learning algorithm works in a straightforward manner also in the presence of missing values. We just take the sum over only those indices i and j for which the data entry x_{ij} (the ijth element of $\mathbf{X}$) is observed, in short $(i, j) \in O$. The cost function is

$$C = \sum_{(i,j) \in O} e_{ij}^2 , \qquad \text{with} \qquad e_{ij} = x_{ij} - \sum_{k=1}^{c} a_{ik} s_{kj} , \tag{3}$$

and its partial derivatives are

$$\frac{\partial C}{\partial a_{il}} = -2 \sum_{j|(i,j) \in O} e_{ij} s_{lj} , \qquad \frac{\partial C}{\partial s_{lj}} = -2 \sum_{i|(i,j) \in O} e_{ij} a_{il} . \tag{4}$$

We 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 ($d \gg 1$). Here we use only the diagonal part of the Hessian matrix, and include 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

$$a_{il} \leftarrow a_{il} - \gamma' \left(\frac{\partial^2 C}{\partial a_{il}^2} \right)^{-\alpha} \frac{\partial C}{\partial a_{il}} = a_{il} + \gamma \frac{\sum_{j|(i,j)\in O} e_{ij} s_{lj}}{\left(\sum_{j|(i,j)\in O} s_{lj}^2 \right)^{\alpha}} , \tag{5}$$

$$s_{lj} \leftarrow s_{lj} - \gamma' \left(\frac{\partial^2 C}{\partial s_{lj}^2} \right)^{-\alpha} \frac{\partial C}{\partial s_{lj}} = s_{lj} + \gamma \frac{\sum_{i|(i,j)\in O} e_{ij} a_{il}}{\left(\sum_{i|(i,j)\in O} a_{il}^2 \right)^{\alpha}} . \tag{6}$$

For comparison, we also consider two alternative PCA methods that can be adapted for missing values.

Imputation Algorithm Another option is to complete the data matrix by iteratively imputing the missing values (see, e.g., [8]). Initially, the missing values can be replaced by zeroes. With completed data, PCA can be solved by eigenvalue decomposition of the covariance matrix. Now, the missing values are replaced using the product $\mathbf{AS}$, PCA is applied again, and this process is iterated until convergence. This approach requires the use of the complete data matrix, and therefore it is computationally very expensive if a large part of the data matrix is missing.

EM Algorithm Grung and Manne [9] studied the EM-like algorithm for PCA in the case of missing values.[1] In the E-step, $\mathbf{A}$ is fixed and $\mathbf{S}$ is solved as a least squares problem. In the M-step, $\mathbf{S}$ is fixed and $\mathbf{A}$ is solved again as a least squares problem. Computations are a lot heavier than in the fully observed case, but still, experiments in [9] showed a faster convergence compared to the iterative imputation algorithm.

4 Overfitting in PCA

A trained PCA model can be used for reconstructing missing values by $\hat{x}_{ij} = \sum_{k=1}^{c} a_{ik} s_{kj}$. Although PCA performs a linear transformation of data, overfitting is a serious problem for large-scale datasets with lots of missing values. This happens when the cost value (3) is small for training data but the quality of prediction $\hat{x}_{ij}$ is poor for new data. This effect is illustrated using the following toy example.

[1] The procedure studied in [9] can be seen as the zero-noise limit of the EM algorithm for a probabilistic PCA model [8].

Assume that the observation space is $d = 2$-dimensional, and most of the data are only partly observed, that is either x_{1j} or x_{2j} is unknown for most js. These observations are represented by triangles placed on the two axes in Fig. 1. There are only two full observations (x_{1j}, x_{2j}) which are shown on the plot by circles. A solution which minimizes the cost function (3) to zero is defined by a line that passes through the two fully observed data points (see the left subfigure). The missing values are then reconstructed by points lying on the line.

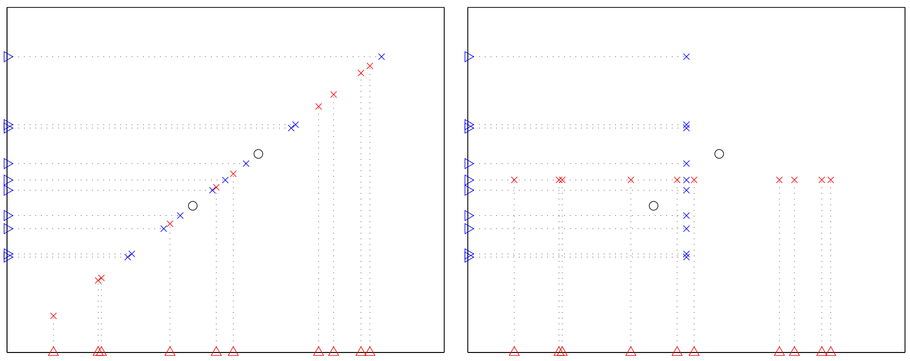
Fig. 1. An artificial example where all but two data points have one of the two components missing. On the left, the correlation between the components is determined by these two samples, giving a badly overfitted solution. On the right, the desired solution where the correlation is not trusted as much (the reconstruction is obtained using the VB algorithm explained in Section 4).

In this example, the solution is defined by only two points and the model is clearly overfitted: There is very little evidence in the data that there exists a significant correlation between the two dimensions. The overfitting problem is even more severe in high-dimensional problems because it is likely that there exist many such pairs of directions in which the evidence of correlation is represented by only a few samples. The right subfigure of Fig. 1 shows a regularized solution that is not overfitted. The correlation is not trusted and the missing values are reconstructed close to the row-wise means. Note that in regularized PCA, the reconstructions are no longer projections to the underlying subspace.

Another way to examine overfitting is to compare the number of model parameters to the number of observed values in data. A rule of thumb is that the latter should be at least tenfold to avoid overfitting. Consider the subproblem of finding the jth column vector of $\mathbf{S}$ given jth column vector of $\mathbf{X}$ while regarding $\mathbf{A}$ a constant. Here, c parameters are determined by the observed values of the jth column vector of $\mathbf{X}$. If the column has fewer than $10c$ observations, it is likely to suffer from overfitting, and if it has fewer than c observations, the subproblem is underdetermined.

Regularization A popular way to regularize ill-posed problems is penalizing the use of large parameter values by adding a proper penalty term into the cost function. This can be obtained using a probabilistic formulation with (independent) Gaussian priors and a Gaussian noise model:

$$p(x_{ij} \mid \mathbf{A}, \mathbf{S}) = \mathcal{N}\left(x_{ij}; \sum_{k=1}^{c} a_{ik} s_{kj}, v_x\right) \tag{7}$$

$$p(\mathbf{A}) = \prod_{i=1}^{d} \prod_{k=1}^{c} \mathcal{N}(a_{ik}; 0, 1), \quad p(\mathbf{S}) = \prod_{k=1}^{c} \prod_{j=1}^{n} \mathcal{N}(s_{kj}; 0, v_{sk}) \ . \tag{8}$$

The cost function (ignoring constants) is minus logarithm of the posterior of the unknown parameters:

$$C_{\mathrm{BR}} = \sum_{(i,j)\in O} \left(e_{ij}^2/v_x + \ln v_x\right) + \sum_{i=1}^{d} \sum_{k=1}^{c} a_{ik}^2 + \sum_{k=1}^{c} \sum_{j=1}^{n} \left(s_{kj}^2/v_{sk} + \ln v_{sk}\right) \ . \tag{9}$$

The cost function can be minimized using a gradient-based approach as described in Section 3. The corresponding update rules are similar to (5)–(6) except for the extra terms which come from the prior. Note that in case of joint optimization of C_{BR} w.r.t. a_{ik}, s_{kj}, v_{sk}, and v_x, the cost function (9) has a trivial minimum with $s_{kj} = 0$, $v_{sk} \to 0$. We try to avoid this minimum by using an orthogonalized solution provided by unregularized PCA for initialization. Note also that setting v_{sk} to small values for some components k is equivalent to removal of irrelevant components from the model. This allows for automatic determination of the proper dimensionality c instead of discrete model comparison (see, e.g., [10]).

Variational Bayesian Learning Variational Bayesian (VB) learning provides even stronger tools against overfitting. VB version of PCA [10] approximates the joint posterior of the unknown quantities using a simple multivariate distribution. Each model parameter is described *a posteriori* using independent Gaussian distributions: $q(a_{ik}) = \mathcal{N}(a_{ik}; \overline{a}_{ik}, \widetilde{a}_{ik})$ and $q(s_{kj}) = \mathcal{N}(s_{kj}; \overline{s}_{kj}, \widetilde{s}_{kj})$, where $\overline{a}_{ik}$ and $\overline{s}_{kj}$ denote the mean of the solution and $\widetilde{a}_{ik}$ and $\widetilde{s}_{kj}$ denote the variance of each parameter. The means $\overline{a}_{ik}$, $\overline{s}_{kj}$ can then be used as point estimates of the parameters while the variances $\widetilde{a}_{ik}$, $\widetilde{s}_{kj}$ define the reliability of the estimates (or credible regions). The direct extension of the method in [10] to missing values can be computationally very demanding. VB-PCA has been used to reconstruct missing values in [11, 12] with algorithms that complete the data matrix, which is also very inefficient in case a large part of data is missing. In this article, we implement VB learning using a gradient-based procedure similar to the subspace learning algorithm described in Section 3.

By applying the framework described in [12] to the model in Eqs. (7–8), the cost function becomes:

$$C_{\mathrm{KL}} = \mathrm{E}_q\left\{\ln \frac{q(\mathbf{A}, \mathbf{S})}{p(\mathbf{X}, \mathbf{A}, \mathbf{S})}\right\} = \sum_{(i,j)\in O} C_{xij} + \sum_{i=1}^{d} \sum_{k=1}^{c} C_{aik} + \sum_{k=1}^{c} \sum_{j=1}^{n} C_{skj} \ , \tag{10}$$

where individual terms are

$$C_{xij} = \frac{(x_{ij} - \sum_{k=1}^{c} \overline{a}_{ik}\overline{s}_{kj})^2 + \sum_{k=1}^{c} \left(\widetilde{a}_{ik}\overline{s}_{kj}^2 + \overline{a}_{ik}^2\widetilde{s}_{kj} + \widetilde{a}_{ik}\widetilde{s}_{kj}\right)}{2v_x} + \frac{\ln 2\pi v_x}{2},$$

$$C_{aik} = \frac{\overline{a}_{ik}^2 + \widetilde{a}_{ik}}{2} - \frac{1}{2}\ln \widetilde{a}_{ik} - \frac{1}{2}, \quad C_{skj} = \frac{\overline{s}_{kj}^2 + \widetilde{s}_{kj}}{2v_{sk}} - \frac{1}{2}\ln \frac{\widetilde{s}_{kj}}{v_{sk}} - \frac{1}{2}.$$

We update $\widetilde{a}$ and $\widetilde{s}$ to minimize the cost by setting the gradient of the cost function to zero:

$$\widetilde{a}_{ik} \leftarrow \left[1 + \sum_{j|(i,j)\in O} \frac{\overline{s}_{kj}^2 + \widetilde{s}_{kj}}{v_x}\right]^{-1}, \quad \widetilde{s}_{kj} \leftarrow \left[\frac{1}{v_{sk}} + \sum_{i|(i,j)\in O} \frac{\overline{a}_{ik}^2 + \widetilde{a}_{ik}}{v_x}\right]^{-1}. \tag{11}$$

The gradients for learning $\overline{a}$ and $\overline{s}$ are somewhat similar to (4):

$$\frac{\partial C_{\mathrm{KL}}}{\partial \overline{a}_{il}} = \overline{a}_{il} + \sum_{j|(i,j)\in O} \frac{-\left(x_{ij} - \sum_{k=1}^{c} \overline{a}_{ik}\overline{s}_{kj}\right)\overline{s}_{lj} + \overline{a}_{il}\widetilde{s}_{lj}}{v_x}, \tag{12}$$

$$\frac{\partial C_{\mathrm{KL}}}{\partial \overline{s}_{lj}} = \frac{\overline{s}_{lj}}{v_{sk}} + \sum_{i|(i,j)\in O} \frac{-\left(x_{ij} - \sum_{k=1}^{c} \overline{a}_{ik}\overline{s}_{kj}\right)\overline{a}_{il} + \widetilde{a}_{il}\overline{s}_{lj}}{v_x}. \tag{13}$$

We can use the speed-up described in Section 3 by computing the second derivatives. These derivatives happen to coincide with the inverse of the updated variances given in (11): $\partial^2 C_{\mathrm{KL}}/\partial\overline{a}_{il}^2 = \widetilde{a}_{il}^{-1}$ and $\partial^2 C_{\mathrm{KL}}/\partial\overline{s}_{lj}^2 = \widetilde{s}_{lj}^{-1}$. The v_x and v_s parameters are updated to minimize the cost C_{KL} assuming all the other parameters fixed. The complete algorithm works by alternating four update steps: $\{\widetilde{a}\}$, $\{\widetilde{s}\}$, $\{\overline{a}, \overline{s}\}$, and $\{v_x, v_s\}$.

5 Experiments

Collaborative filtering is the task of predicting preferences (or producing personal recommendations) by using other people's preferences. The Netflix problem [13] is such a task. 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 the 98.8% of the values are thus missing. We tried to find $c = 15$ principal components from the data using a number of methods.[2] The mean rating was subtracted for each movie and robust estimation of the mean was used for the movies with few ratings.

[2] The PCA approach has been considered by other Netflix contestants as well (see, e.g., [14, 15]).

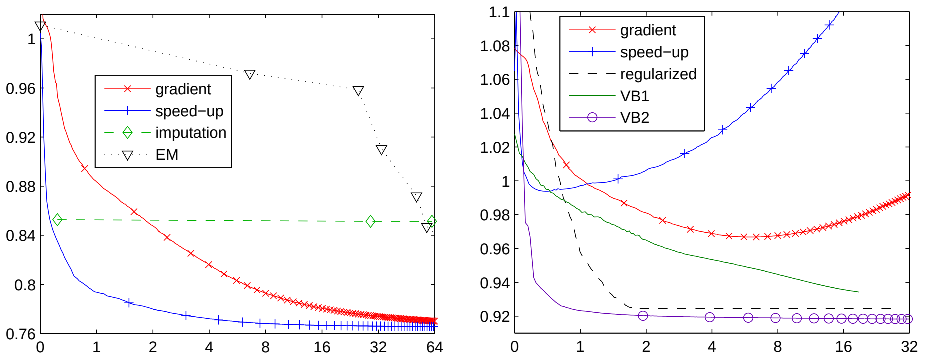

Fig. 2. *Left:* Learning curves for unregularized PCA (Section 3) applied to the Netflix data: Root mean square error on the training data is plotted against computation time in hours. Runs are given for two values of the speed-up parameter α and marks are plotted after every 50 iterations. For comparison, the training errors for the imputation algorithm and the EM algorithm are shown. The time scale is linear below 1 and logarithmic above 1. *Right:* The root mean square error on the validation data from the Netflix problem during runs of several algorithms: basic PCA (Section 3) with two values of α, regularized PCA (Section 4) and VB (Section 4). VB1 has v_{sk} fixed to large values while VB2 updates all the parameters. The curves clearly reveal overlearning for unregularized PCA.

Computational Performance In the first set of experiments we compared the computational performance of different algorithms for PCA with missing values. The root mean square (rms) error is measured on the training data, that is, the observed values in the training set. All experiments were run on a dual cpu AMD Opteron SE 2220.

The comparison methods, the imputation algoritm and the EM algorithm, were very slow, except for the first iteration of the imputation algorithm due to the complete data matrix being sparse. Fig. 2 (left) shows the learning curves. The closer a curve is to the origin, the faster the algorithm minimizes the cost function.

We also tested the subspace learning algorithm described in Section 3 with and without the proposed speed-up, starting from the same random starting point. The learning rate γ was adapted such that if an update decreased the cost function, γ was multiplied by 1.1. Each time an update would increase the cost, the update was canceled and γ was divided by 2. The best α seemed to be around 0.6 to 0.7, the curve shown in Fig. 2 is for $\alpha = 5/8$. It gave a more than tenfold speed-up compared to the gradient descent algorithm even if one iteration took on average 97 seconds against the gradient descent's 57 seconds.

Overfitting We compared PCA (Section 3), regularized PCA (Section 4) and VB-PCA (Section 4) by computing the root mean square reconstruction error for the validation set, that is, ratings that were not used for training. We tested

VB-PCA by firstly fixing v_{sk} to large values (this run is marked as VB1 in Fig. 2) and secondly by adapting them (marked as VB2) to isolate the effects of the two types of regularization. We initialized regularized PCA and VB1 using unregularized subspace learning algorithm with $\alpha = 0.625$ transformed into the PCA solution. VB2 was initialized using VB1. The parameter α was set to 2/3.

Fig. 2 (right) shows the results. The performance of unregularized PCA starts to degrade after a while of learning, especially with large values of α. This effect, known as overlearning, did not appear with VB. Regularization helped a lot and the best results were obtained using VB2: The final validation rms error was 0.9180 and the training rms error was 0.7826 which is naturally a bit larger than the unregularized 0.7657.

Acknowledgments This work was supported in part by the Academy of Finland under its Centers for Excellence in Research Program, 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 [13] for providing the data.

References

1. Pearson, K.: On lines and planes of closest fit to systems of points in space. Philosophical Magazine **2**(6) (1901) 559–572
2. Jolliffe, I.: Principal Component Analysis. Springer-Verlag (1986)
3. Bishop, C.: Pattern Recognition and Machine Learning. Springer-Verlag (2006)
4. Diamantaras, K., Kung, S.: Principal Component Neural Networks - Theory and Application. Wiley (1996)
5. Haykin, S.: Modern Filters. Macmillan (1989)
6. Cichocki, A., Amari, S.: Adaptive Blind Signal and Image Processing - Learning Algorithms and Applications. Wiley (2002)
7. Oja, E.: Neural networks, principal components, and subspaces. International Journal of Neural Systems **1**(1) (1989) 61–68
8. Tipping, M., Bishop, C.: Probabilistic principal component analysis. Journal of the Royal Statistical Society: Series B (Statistical Methodology) **61**(3) (1999) 611–622
9. Grung, B., Manne, R.: Missing values in principal components analysis. Chemometrics and Intelligent Laboratory Systems **42**(1) (August 1998) 125–139
10. Bishop, C.: Variational principal components. In: Proc. 9th Int. Conf. on Artificial Neural Networks (ICANN99). (1999) 509–514
11. Oba, S., Sato, M., Takemasa, I., Monden, M., Matsubara, K., Ishii, S.: A Bayesian missing value estimation method for gene expression profile data. Bioinformatics **19**(16) (2003) 2088–2096
12. Raiko, T., Valpola, H., Harva, M., Karhunen, J.: Building blocks for variational Bayesian learning of latent variable models. Journal of Machine Learning Research **8**(Jan) (2007) 155–201
13. Netflix: Netflix prize webpage (2007) `http://www.netflixprize.com/`.
14. Funk, S.: Netflix update: Try this at home. Available at `http://sifter.org/~simon/journal/20061211.html` (December 2006)
15. Salakhutdinov, R., Mnih, A., Hinton, G.: Restricted Boltzmann machines for collaborative filtering. In: Proc. Int. Conf. on Machine Learning. (2007) To appear.