BINARY PRINCIPAL COMPONENT ANALYSIS IN THE NETFLIX COLLABORATIVE FILTERING TASK

László Kozma, Alexander Ilin, Tapani Raiko

Helsinki University of Technology
Adaptive Informatics Research Center
P.O. Box 5400, FI-02015 TKK, Finland
firstname.lastname@tkk.fi

ABSTRACT

We propose an algorithm for binary principal component analysis (PCA) that scales well to very high dimensional and very sparse data. Binary PCA finds components from data assuming Bernoulli distributions for the observations. The probabilistic approach allows for straightforward treatment of missing values. An example application is collaborative filtering using the Netflix data. The results are comparable with those reported for single methods in the literature and through blending we are able to improve our previously obtained best result with PCA.

1. INTRODUCTION

Recommendation systems form an integral component of many modern e-commerce websites. Users often rate items on a numerical scale (e.g., from 1 to 5). If the system is capable of accurately predicting the rating a user would give to an item, it can use this knowledge to make recommendations to the user. The collected ratings can be represented in the form of a matrix in which each column contains ratings given by one user and each row contains ratings given to one item. Naturally most of the entries in the matrix are missing (as any given user rates just a subset of the items) and the goal is to predict these missing values. This is known as the *collaborative filtering* task.

Obtaining accurate predictions is the central problem in the research on recommendation systems. Although there are certain requirements that a practical recommendation system should fulfill (such as ease of interpretation, speed of response, ability to learn online), this work addresses only the collaborative filtering problem.

Collaborative filtering is an interesting benchmark problem for the machine learning community because a great variety of methods is applicable there. The competition recently organized by Netflix has stirred up the interest of many researchers by a one million dollar prize. A vast collection of approaches has been tried by the competing teams. Among the popular ones are nearest neighbor methods on either users, items or both, and methods using matrix decomposition [1, 2, 3, 4, 5]. It seems that no single method can provide prediction accuracy comparable with the current leaders in the competition. The most successful teams use a blend of a high number of different algorithms [3]. Thus, designing methods that may not be extremely good in the overall performance but which look at the data at a different angle may be beneficial.

The method presented in this work follows the matrix decomposition approach, which models the ratings using a linear combination of some features (factors) characterizing the items and users. In contrast to other algorithms reported in the Netflix competition, we first binarize the ratings and then perform matrix decomposition on binary data. The proposed model is closely related to the models presented in [6, 7, 8, 9]. Tipping [6] used a similar model for visualization of binary data, while Collins et al. [7] proposed a general method for the whole exponential family. Schein et al. [8] presented a practical algorithm for binary PCA and made extensive comparisons to show that it works better than traditional PCA in the case of binary data. Srebro and Jaakkola [9] extended the binary PCA model to the case where each element of the data has a weight, which includes the case with missing values (using a zero weight is the same as having a missing value in the data). The most important difference is that our algorithm is specially designed for high-dimensional data with lots of missing values and therefore we use point-estimates for model parameters and use extra regularization terms.

The paper is organized as follows. The proposed binary PCA algorithm is presented in Section 2. Its application to MovieLens and Netflix data is presented in Section 3, followed by the discussion.

2. METHOD

The approach deals with an unsupervised analysis of data in a matrix $\mathbf{Y}$ that contains three types of values: zeroes, ones and missing values. We make the assumption that the data are missing at random, that is, we do not try to model when the values are observed but only whether the values are zeroes or ones.

2.1. Binarization

As a preprocessing step, the ratings are encoded with binary values, according to the following scheme:

$$\begin{aligned} 1 &\rightarrow 0000 \\ 2 &\rightarrow 0001 \\ 3 &\rightarrow 0011 \\ 4 &\rightarrow 0111 \\ 5 &\rightarrow 1111 \end{aligned}$$

With this scheme, each element in the data tells whether a rating is greater or smaller than a particular threshold. This kind of binarization can be used also for continuous valued data.

Let us denote by $\mathbf{X}$ the sparsely populated matrix of ratings from 1 to 5 stars. $\mathbf{X}$ has dimensions $m \times p$, where m and p are the number of movies and the number of people, respectively. After binarization, we obtain binary matrices $\mathbf{X}_1$, $\mathbf{X}_2$, $\mathbf{X}_3$, $\mathbf{X}_4$ (one for each digit of the rating values), which we summarize in one matrix $\mathbf{Y}$ with dimensions $d \times p$, where $d = 4m$.

2.2. The model

We model the probability of each element y_{ij} of matrix $\mathbf{Y}$ to be 1 using the following formula:

$$P(y_{ij} = 1) = \sigma(\mathbf{a}_i^{\mathrm{T}} \mathbf{s}_j) \tag{1}$$

where $\mathbf{a}_i$ and $\mathbf{s}_j$ are column vectors with c elements and $\sigma(\cdot)$ is the logistic function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \,. \tag{2}$$

This can be understood as follows: The probability that the j-th person rates the i-th movie with the rating greater then l (assuming that y_{ij} is taken from the binary matrix $\mathbf{X}_l$) depends on the combination of c features of the movie (collected in vector $\mathbf{a}_i$) and c features of the person (collected in vector $\mathbf{s}_j$).

The features of the movies and people are not known and have to be estimated to fit the available ratings. We summarize the features in two matrices

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \dots & \mathbf{a}_d \end{bmatrix}^{\mathrm{T}} \tag{3}$$

$$\mathbf{S} = \begin{bmatrix} \mathbf{s}_1 & \dots & \mathbf{s}_p \end{bmatrix} \,. \tag{4}$$

with dimensionalities $d \times c$ and $c \times p$ respectively. In order to have the bias term b_i in the model $\sigma(b_i + \mathbf{a}_i^{\mathrm{T}} \mathbf{s}_j)$, we fixed the last row of $\mathbf{S}$ to all ones. Then, the elements in the last column of $\mathbf{A}$ play the role of the bias terms b_i.

We express the likelihood of $\mathbf{Y}$ as a product of Bernoulli distributions

$$L = \prod_{ij \in O} \sigma(\mathbf{a}_i^{\mathrm{T}} \mathbf{s}_j)^{y_{ij}} (1 - \sigma(\mathbf{a}_i^{\mathrm{T}} \mathbf{s}_j))^{(1 - y_{ij})} \,, \tag{5}$$

where O denotes the set of elements in $\mathbf{Y}$ corresponding to observed ratings. We use Gaussian priors for $\mathbf{A}$ and $\mathbf{S}$ to make the solution less prone to overfitting:

$$P(s_{kj}) = \mathcal{N}(0, 1) \tag{6}$$

$$P(a_{ik}) = \mathcal{N}(0, v_k) \tag{7}$$

$$i = 1, \dots, d\,; \quad k = 1, \dots, c\,; \quad j = 1, \dots, p\,.$$

Here we denote by $\mathcal{N}(\mu, v)$ the Gaussian probability density function with mean μ and variance v. The prior variance for the elements in $\mathbf{S}$ is set to unity to fix the scaling indeterminacy of the model. We use a separate hyperparameter v_k for each column of $\mathbf{A}$ so that, if the number of components c is chosen to be too large, unnecessary components can go to zero when the corresponding v_k is close to zero. In practice we use a separate v_k for different parts of $\mathbf{Y}$ corresponding to matrices $\mathbf{X}_l$, but we omit this detail here to simplify the notation.

2.3. Learning

We use simple maximum a posteriori estimation for the model parameters $\mathbf{A}$, $\mathbf{S}$, v_k to make the method scale well to high-dimensional problems. The logarithm of the posterior of the unknown parameters is:

$$\begin{aligned} F = &\sum_{ij \in O} y_{ij} \log \sigma(\mathbf{a}_i^{\mathrm{T}} \mathbf{s}_j) \\ &+ \sum_{ij \in O} (1 - y_{ij}) \log(1 - \sigma(\mathbf{a}_i^{\mathrm{T}} \mathbf{s}_j)) \\ &- \sum_{i=1}^{d} \sum_{k=1}^{c} \left[\frac{1}{2} \log 2\pi v_k + \frac{1}{2 v_k} a_{ik}^2 \right] \\ &- \sum_{k=1}^{c} \sum_{j=1}^{p} \left[\frac{1}{2} \log 2\pi + \frac{1}{2} s_{kj}^2 \right] \end{aligned} \tag{8}$$

We perform gradient-based maximization of F w.r.t. parameters $\mathbf{A}$ and $\mathbf{S}$. The required derivatives are

$$\frac{\partial F}{\partial \mathbf{a}_i} = \sum_{j | ij \in O} \mathbf{s}_j^T (y_{ij} - \sigma(\mathbf{a}_i^{\mathrm{T}} \mathbf{s}_j)) - \mathrm{diag}(1/v_k) \mathbf{a}_i \tag{9}$$

$$\frac{\partial F}{\partial \mathbf{s}_j} = \sum_{i | ij \in O} \mathbf{a}_i^T (y_{ij} - \sigma(\mathbf{a}_i^{\mathrm{T}} \mathbf{s}_j)) - \mathbf{s}_j \,, \tag{10}$$

where $\mathrm{diag}(1/v_k)$ is a diagonal matrix with $1/v_k$ on the main diagonal. Here we rely on the property of the sigmoid function $\sigma' = \sigma(1-\sigma)$. We use the gradient-ascent update rules

$$\mathbf{a}_i \leftarrow \mathbf{a}_i + \alpha \frac{\partial F}{\partial \mathbf{a}_i} \quad (11)$$

$$\mathbf{s}_k \leftarrow \mathbf{s}_k + \alpha \sqrt{\frac{p}{d}} \frac{\partial F}{\partial \mathbf{s}_j} \quad (12)$$

where the scaling $\sqrt{p/d}$ accounts for the fact that the number of rows and columns in the data can significantly differ. Such scaling is a crude but effective approximation of the Newton's method. We use a simple strategy of selecting the learning rate: α is halved if the update leads to decrease of F and α is increased by 20% on the next iteration if the update is successful.

The variance parameters v_k are point-estimated using a simple update rule which maximizes F:

$$v_k = \frac{1}{d} \sum_{i=1}^{d} a_{ik}^2 \quad (13)$$

2.4. Prediction

Once $\mathbf{A}$ and $\mathbf{S}$ have been estimated, one can compute the probabilities for each element of matrices $\mathbf{X}_l$ using (1). We can then transform the probabilities to the actual ratings by first calculating the probability of each particular rating value:

$$P_{x=1} = (1-x_1)(1-x_2)(1-x_3)(1-x_4) \quad (14)$$

$$P_{x=2} = (1-x_1)(1-x_2)(1-x_3)x_4 \quad (15)$$

$$P_{x=3} = (1-x_1)(1-x_2)x_3x_4 \quad (16)$$

$$P_{x=4} = (1-x_1)x_2x_3x_4 \quad (17)$$

$$P_{x=5} = x_1x_2x_3x_4 \quad (18)$$

where x is one rating in matrix $\mathbf{X}$, $P_{x=r}$ denotes the probability that $x = r$ and x_l ($l = 1, \ldots, 4$) are the probabilities computed for the corresponding elements in matrices $\mathbf{X}_l$. Then we estimate the rating as expectation

$$\hat{x} = \frac{1P_{x=1} + 2P_{x=2} + 3P_{x=3} + 4P_{x=4} + 5P_{x=5}}{P_{x=1} + P_{x=2} + P_{x=3} + P_{x=4} + P_{x=5}} . \quad (19)$$

2.5. Comparison to PCA

What are the differences between binary PCA and application of traditional PCA to binary data? Traditional PCA corresponds to using the squared error $(y_{ij} - \mathbf{a}_i^{\mathrm{T}}\mathbf{s}_j)^2$ as the learning criterion. Fig. 1 shows the loss function as a function of $\mathbf{a}_i^{\mathrm{T}}\mathbf{s}_j$. There are two notable differences. Firstly, the loss of binary PCA grows only linearly (instead of quadratically), so it is more robust against outliers (few data samples with bad reconstructions). Secondly, the loss of binary

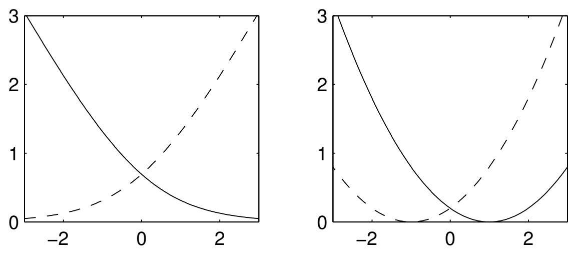

Fig. 1. Loss function $-\log P(y_{ij})$ as a function of $\mathbf{a}_i^{\mathrm{T}}\mathbf{s}_j$ in the case where the observation is 1 (solid line) or not (dashed line). *Left:* Binary PCA. *Right:* Traditional PCA (where binarization -1,1 is used for symmetry).

PCA is monotonic. Traditional PCA can generate predictions outside of the valid range. When the prediction $\mathbf{a}_i^{\mathrm{T}}\mathbf{s}_j$ is greater than 1, there is an incentive in the learning criterion to change the prediction to the negative direction. This is clearly undesired, and the monotonicity of binary PCA avoids the effect. Such an effect can be observed in Fig. 3 where traditional PCA predicts ratings over 5 and below 1.

How about differences between the proposed approach and traditional PCA for the ratings 1 to 5 without binarization? The biggest difference is that the data matrix is four times as large with the proposed approach. This increases the number of parameters as the number of rows in the matrix $\mathbf{A}$ is fourfold. The increase brings some expressive power. Consider for instance that some people (or movies) tend to have mostly extreme ratings 1 and 5. The binary model could find and make use of this effect, unlike the traditional PCA model. On the other hand, the increased number of parameters might cause overfitting and increases computational complexity.

3. EXPERIMENTS

We implemented our method and tested it on two widely used data sets for collaborative filtering: the MovieLens 100K and the Netflix data set. Both data sets contain integer ratings with values 1 to 5. Other available information, such as the titles of the movies, time of rating, data about users were not used, although they might help the prediction task.

The algorithm was implemented in Matlab, using functions for sparse matrices, with the most computationally expensive components implemented in C++. The reported running times were measured on a dual cpu AMD Opteron SE 2220 machine.

3.1. MovieLens data

The MovieLens 100K data set from the GroupLens Research Group at the University of Minnesota contains 100000 ratings with values 1-to-5 given by 943 users on 1682 movies.

# components c	Train rms	Test rms
binary PCA, movies$\times$users		
10	0.8956	0.9248
binary PCA, users$\times$movies		
10	0.8449	0.9028
20	0.8413	0.9053
30	0.8577	0.9146
PCA, users$\times$movies		
10	0.7674	0.8905
20	0.7706	0.8892
30	0.7696	0.8880

Table 1. Performance obtained for MovieLens data

About 6.3% of all possible ratings are given. We randomly split up the available ratings into a training set (95%) and a test set (5%). We removed the 8 movies which do not have any ratings in the training set as a result of the split.

First we ran our method on both the movies$\times$users and the transposed matrix (see Table 1). The table shows the root mean square reconstruction error for the training set and for the test set. We want to present how the choice of the number of components affects the performance of the method. We also test whether the use of the original movies$\times$users matrix or its transpose gives better results. The main difference between these is the way the bias values are obtained (bias of each person or bias of each movie). Having the users in the rows gives somewhat better results as reported for other methods as well. Simple averaging of the results from the original and transposed data with 10 components yielded a test error of 0.9014, which is slightly better than either of the two. However, for a proper blending of the results an additional validation set would be needed. We compare the results with those obtained earlier using the PCA with missing values method [4].

We tried the algorithm on MovieLens data using different number c of latent components. We noticed that using too many components did not lead to serious overfitting problems and the models with different c effectively used similar numbers of components. This is the positive effect of regularization. Using more than 10 components, which was the optimal number for MovieLens, resulted in slower convergence but the test error did not start to grow. The convergence time was about 6 minutes for the 30-component case.

Fig. 2 shows that the reconstruction error on binarized data is a good indicator of the reconstruction error on 1-to-5 ratings. There, we display the results of five runs with different initializations for the model with 10 components. Although the model converged to five different local minima, the achieved performance is similar between the runs.

Finally, we compared the proposed binarization scheme (see Section 2.1) with a more standard one when a single binary digit is set to 1 for each rating value (i.e. $1 \to 10000$, $2 \to 01000$ and so on). The final error was significantly larger: 1.0779 on the training set and 1.1062 on the test set.

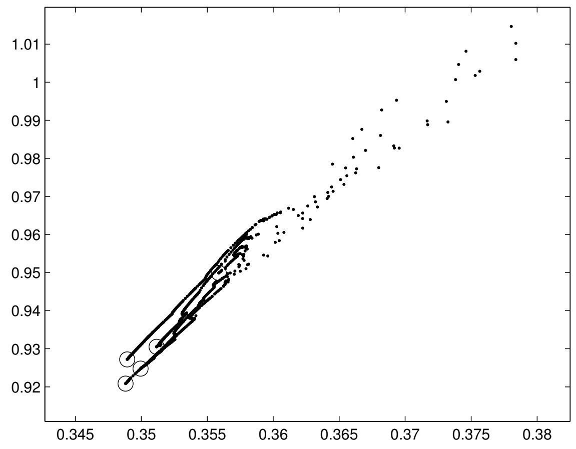

Fig. 2. Test rms error on binary data (x-axis) against the test rms error on 1-to-5 ratings for the MovieLens data. The scatter plots show intermediate results during learning for five runs with different initializations. The circles represent the solutions at the convergence.

In Fig. 3, we compare the predictions of the ratings given by our method with the predictions obtained with our earlier PCA model [4]. Even though the difference in total prediction error is less than 2%, there seems to be significant difference in the predictions. Therefore it is hoped that the method captures different phenomena in the data than PCA.

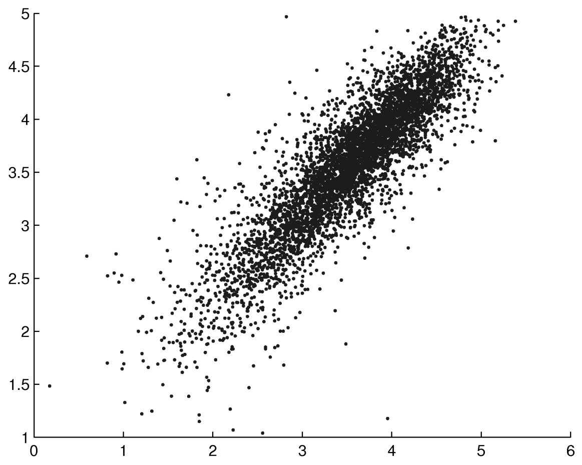

Fig. 3. Predictions on the test set from the MovieLens data using PCA (x-axis) and the binary PCA model (y-axis).

Method	Probe rms	Quiz subset
VBPCA, 50 components	0.9055	0.9070
BinPCA, 20 components	0.9381	0.9392
Blend	0.9046	0.9062

Table 2. Performance obtained for Netflix data

3.2. Netflix data

The Netflix data set contains ratings given by 480189 users on 17770 movies. Only about 1.2% of the ratings are available, which are split up by the organizers as training and probe set.

We ran the model with different number of components using a 100-hour running time limit (see Fig. 4) and the overall best result was obtained with just 20 components. The obtained reconstruction rms errors when the model was trained using the Netflix training set only are shown in Fig. 2. Although the performance level obtained with the binary PCA model (0.9392) was not as good as the one of the PCA model (0.9070), blending the two approaches allowed to improve the overall performance to 0.9062. Blending was done by doing linear regression from the predictions of the two models to the real ratings on the probing data. The obtained coefficients were 0.8590 for PCA and 0.1369 for binary PCA. Note that the numbers presented here can still be improved if also the probing data is used for learning.

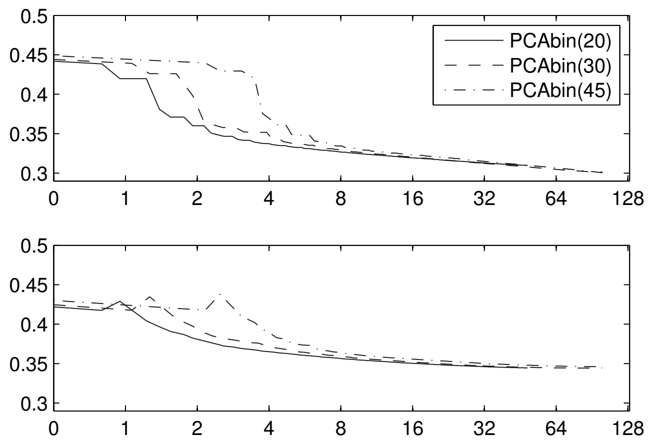

Fig. 4. Running binary PCA on the Netflix data. The x-axis shows running time in hours (linear between 0 and 1, log-scale after 1). Training rms is shown above and probe rms is shown below (both on binary digits). The test error is not increasing at any point which indicates that there is no overlearning.

In Fig. 5, we show predictions obtained with the model for part of probing data in comparison with the true ratings. One can see that the predicted matrices $\mathbf{X}_l$ are in good agreement with the binarization scheme. It is notable that the method very rarely predicts the rating 1. It is the least common rating already in the data, but a conservative predictor avoids it even further.

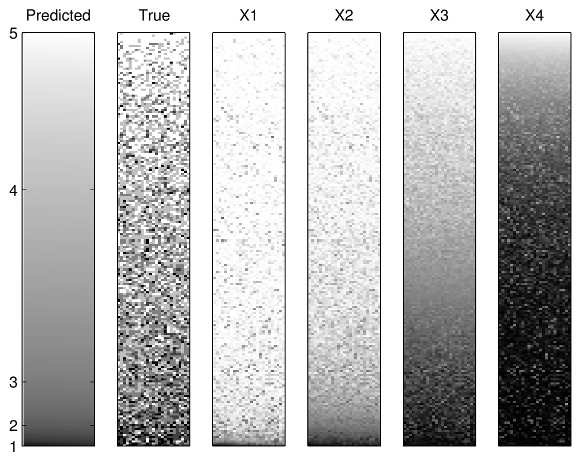

Fig. 5. A random sample of 5000 ratings in the Netflix probing data is sorted (from top to bottom) according to the binary PCA prediction. The images show the predicted rating, the true rating, and the predicted binary probabilities for the four digits (from left to right).

3.3. Computational complexity

Each iteration of the proposed algorithm takes $O(Nc)$ operations, where $N = |O|$ is the number of observed values, and assuming $d < p < N$. The costly operations are the summations in Equations (8)–(10). There is only one algorithm previously proposed for binary PCA with missing values. Each iteration of it takes $O(pdc^2)$ operations [9]. For Netflix data and $c = 20$ features, this makes about a thousandfold difference in our advantage.

4. DISCUSSION

We presented a method suitable for large-scale collaborative filtering tasks. It is based on binarizing the ratings and modeling them with binary PCA. The method seems to be robust against overlearning even in problems with lots of missing values. The results appear to be stable with respect to restarts from random initializations.

The prediction performance obtained with the proposed approach is slightly worse than our previous results obtained with PCA [4]. However, the model seems to capture different aspects of the data (see Fig. 3) and blending the proposed approach with other models can improve the overall results. We chose the used optimization algorithm for simplicity. It would be a straightforward task to make the learning algorithm more efficient by using better ways to find the

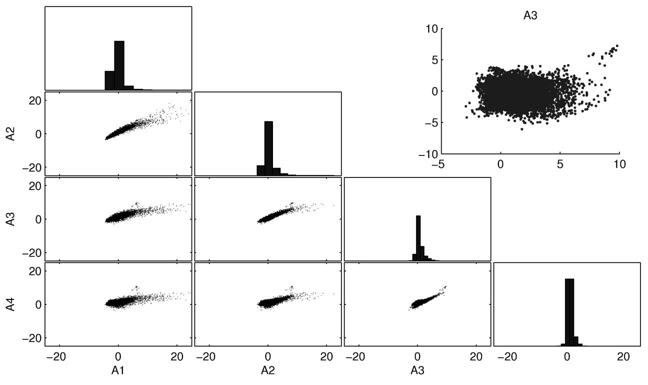

Fig. 6. Scatter plot of the matrix $\mathbf{A}$ learned from the Netflix data. *Bottom left:* The values from the four sub-parts (one for each l) of the most prominent component of the $\mathbf{A}$ matrix, plotted against each other. There are some correlations clearly visible. *Top right:* Two most prominent components of $\mathbf{A}$ plotted against each other, showing only the third sub-part ($l = 3$). There is some non-Gaussian structure visible.

learning rate and by using for instance conjugate gradient methods.

One way to extend the proposed model is to use a more sophisticated prior for $\mathbf{A}$ and $\mathbf{S}$. The most prominent effect that we have ignored in the modeling, is the correlations between the elements in the $\mathbf{A}$ matrix corresponding to different digits. These correlations are clearly visible in Fig. 6. However, the prior model (7) that we use for $\mathbf{A}$ assumes that all elements are independent. Furthermore, as one can see from Fig. 6, there is some non-Gaussian structure that could be modeled with a more complex prior.

Taking into account the posterior uncertainty, for example by using variational Bayesian learning or sampling, can help improve the accuracy of the model (see, e.g. [4]). While this would increase the complexity (both mathematically and computationally), it could still be feasible even for the Netflix problem. This is left as future work.

5. ACKNOWLEDGEMENTS

LK was funded by TKK MIDE programme, project UI-ART. AI was funded by the Academy of Finland through a postdoctoral research project. TR was funded by the Academy of Finland project 'Unsupervised machine learning in latent variable models' and the PASCAL2 Network of Excellence. This publication only reflects the authors' views. We would like to thank Netflix [10] for providing the data.

6. REFERENCES

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

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

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

[4] 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)*, Warsaw, Poland, September 2007.

[5] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *Neural Information Processing Systems 20*, 2008.

[6] M. E. Tipping, "Probabilistic visualization of high-dimensional binary data," in *Advances in Neural Information Processing Systems 11*, 1999, pp. 592–598.

[7] M. Collins, S. Dasgupta, and R. Schapire, "A generalization of principal components analysis to the exponential family," in *Advances in Neural Information Processing Systems 14 (NIPS)*, Cambridge, MA, 2002, MIT Press.

[8] A. Schein, L. Saul, and L. Ungar, "A generalized linear model for principal component analysis of binary data," in *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics*, Key West, FL, January 2003.

[9] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *20th International Conference on Machine Learning (ICML)*, August 2003.

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