Sampling PCA, enhancing recovered missing values in large scale matrices.

Luis Gabriel De Alba Rivera
80555S

May 21, 2009

1 Introduction

Human preferences (the *quality* tags we put on things) are language terms that can be easily translated into a numerical domain. We could assign low values to odd things and high values to enjoyable things, i.e.; rate things according to our experience. These ratings serve us to easily (and grossly) classify and order our preferences from the ones we like the most to the ones we dislike the most. Of course we are limited: we can not rate what we do not know, however; it may be of our interest to know the possible ratings of these unknowns.

In this project we will be working with large and sparse matrices of movies ratings. The objective will be to recover a subset of the missing values as accurately as possible. Recovering these missing values equal to predicting movies ratings and, therefore; predicting movies preferences for different users. The idea of correctly recovering movies ratings for different users has been a hot topic during the last years motivated by the Netflix prize.

The concept of mining users preferences to predict a preference of a third user is called Collaborative Filtering, it involves large data sets and has been used by stores like Amazon and iTunes.

We can start by considering that the preferences of the users are determined by a number of unobserved factors (that later we will call components). These hidden variables can be, for example, music, screenplay, special effects, etc. These variables weight different and are rated independently, however; they, together, sum up for the final rating, the one we observe. Therefore; if we can factorize the original matrix (the one with the ratings) in a set of sub-matrices that represent these hidden factors, we may have a better chance to find the components and values to recover the missing ratings [1]. One approach to find these matrices is to use SVD (Single Value Decomposition), a matrix factorization method. With SVD the objective is to find matrices $\mathbf{U}$ $\mathbf{V}$ minimizing the sum-squared distance to the target matrix $\mathbf{R}$ [2].

For this project we consider matrix $\mathbf{Y}$ to be our only informative input. Matrix $\mathbf{Y}$ is, usually, large and disperse, i.e.; with lots of missing values. The observable values are the ratings given to movies (rows) by users (columns). Our objective is to recover the missing values, or a subset of them, with a small error. We can factorize matrix $\mathbf{Y}$ such that $\mathbf{Y} \approx \mathbf{WX}+\mathbf{m}$. Matrices $\mathbf{W}$ $\mathbf{X}$ $\mathbf{m}$ will let us recover the missing values, of course, the quality of the recovering depends on the quality of these matrices. Sampling will let us improve the fitness of matrices $\mathbf{W}$ $\mathbf{X}$ $\mathbf{m}$ to better recover matrix $\mathbf{Y}$. We can use VB-PCA (Variational Bayes PCA) for the initial decomposition of the input matrix $\mathbf{Y}$. VB-PCA is known to be less prone to over-fitting and more accurate for lager-scale data sets with lots of missing values compared to traditional PCA methods [3, 4]. However; VB-PCA is not compulsory for sampling, a random initialization method is also explored in this project.

2 Sampling PCA

Sampling can be seen as the generation of numerical values with the characteristics of a given distribution. Sampling is used when other approaches are not feasible.

For high-dimensional probabilistic models Markov chain Monte Carlo methods are used to go over the integrals with good accuracy. Gibbs sampling is a well known MCMC method [5, 6]. In Gibbs approach we sample one variable, for example $\mathbf{W}$, conditioned to the remaining variables, $\mathbf{X}$ $\mathbf{m}$. In the following step we sample another variable fixing the rest; we repeat this process generating as many samples as necessary.

In our project we have matrix $\mathbf{Y}$ that is a joint distribution of the form $\mathbf{Y} = \mathbf{WX} + \mathbf{m} + noise$ to predict the missing values in $\mathbf{Y}$ we need to solve:

$$P(\mathbf{Y}_{MIS}|\mathbf{Y}_{OBS}) = \int P(\mathbf{Y}_{MIS}|\mathbf{W}, \mathbf{X}, \mathbf{m}) P(\mathbf{W}, \mathbf{X}, \mathbf{m}|\mathbf{Y}_{OBS})\, d\mathbf{W}\, d\mathbf{X}\, d\mathbf{m} \quad (1)$$

Solving the integral is complex, therefore; we make use of Gibbs sampling to approximate its solution. To recover matrices $\mathbf{W}$ $\mathbf{X}$ $\mathbf{m}$ we need to solve $P(\mathbf{W}|\mathbf{Y}_{OBS}, \mathbf{X}, \mathbf{m})$, $P(\mathbf{X}|\mathbf{Y}_{OBS}, \mathbf{W}, \mathbf{m})$ and $P(\mathbf{m}|\mathbf{Y}_{OBS}, \mathbf{W}, \mathbf{X})$ each one following a Gaussian distribution, contrary to $P(\mathbf{W}, \mathbf{X}, \mathbf{m}|\mathbf{Y}_{OBS})$ that follows an unknown and complex distribution. The mean matrices, $\bar{\mathbf{X}}$ $\bar{\mathbf{W}}$ $\bar{\mathbf{m}}$, and covariance matrices, $\mathbf{\Sigma_x}$ $\mathbf{\Sigma_w}$ $\tilde{\mathbf{m}}$, are calculated according to the formulas provided at [4] Appendix D; this is done as follows:

$$\bar{x}_j = (\bar{\mathbf{W}}_j^T \bar{\mathbf{W}}_j + v\mathbf{I})^{-1} \bar{\mathbf{W}}_j^T (\mathring{\mathbf{Y}}_{:j} - \bar{\mathbf{M}}_j) \quad j = 1, \ldots, p$$
$$\Sigma_{\mathbf{x},j} = v(\bar{\mathbf{W}}_j^T \bar{\mathbf{W}}_j + v\mathbf{I})^{-1}$$

$$\bar{w}_i = (\mathring{\mathbf{Y}}_{i:} - \bar{m}_i)^T \bar{\mathbf{X}}_i^T (\bar{\mathbf{X}}_i \bar{\mathbf{X}}_i^T + v\, diag(w_k^{-1})) \quad i = 1, \ldots, m$$
$$\Sigma_{\mathbf{w},i} = v(\bar{\mathbf{X}}_i \bar{\mathbf{X}}_i^T + v\, diag(w_k^{-1}))$$

$$\bar{m}_i = \frac{w_m}{|O_i|(w_m + v/|O_i|)} \sum_{j \epsilon O_i} [y_{ij} - \bar{\mathbf{w}}_i^T \bar{\mathbf{x}}_j] \quad i = 1, \ldots, m$$
$$\tilde{m}_i = \frac{v w_m}{|O_i|(w_m + v/|O_i|)}$$

Indices $j = 1, \ldots, p$ and $i = 1, \ldots, m$ go over the rows (people) and columns (movies) of matrix $\mathbf{Y}$. $\bar{x}_j$ is the column j of matrix $\bar{\mathbf{X}}$, $\bar{w}_i$ is row i of matrix $\bar{\mathbf{W}}$, $\bar{m}_i$ is element i of matrix $\mathbf{m}$. v and w_m are hyper-parameter from PCA Full/Diag. $\mathring{\mathbf{Y}}$ is the data matrix where the missing values have been replaced with zeroes. O is the set of indices ij for which y_{ij} is observed. O_i is the set of indices j for which y_{ij} is observed. $|O_i|$ is the number of elements in O_i. $\mathbf{I}$ is the identity matrix. $diag$ is the diagonalizing of the referred values.

Using the mean and covariance matrices we are able to sample $\mathbf{W}'$ $\mathbf{X}'$ and $\mathbf{m}'$ using the methods presented in [6]. With the sampled and mean matrices we recover a *full* matrix $\mathbf{Y}'$, i.e.; including the missing values; more of this is explained in the following subsections.

2.1 Recovering the Missing Values

To recover the matrix $\mathbf{Y}$ we need to multiply matrix $\mathbf{W}$ by $\mathbf{X}$ and add the $\mathbf{m}$ bias vector to it. Referring to the ideas presented by [1] matrix $\mathbf{W}$ represents the *different and weighted* factors that conform a movie. On the other hand, matrix $\mathbf{X}$ represents the values assigned to each factor by the different users. The resulting matrix $\mathbf{Y}'$ has, therefore, the ratings given to movies m by users p. The bias term, $\mathbf{m}$, is used to compensate the differences in results from the recovered matrix $\mathbf{Y}'$ and the original observed values used during the training.

To prove the *quality* of the ratings in the recovered matrix $\mathbf{Y}'$ it is necessary to have a test set different from the training set. At every step during sampling when the values are recovered we calculate the Root Mean Square Error, RMSE, using the test set as baseline. RMSE is a well known measure to quantify the amount by which a predictor differs from the value being predicted.

For this project the sampling and recovering process is as follows:

1. Start point $i = 1$, with matrices $\mathbf{W}^i$ $\mathbf{X}^i$ and $\mathbf{m}^i$.
2. Calculate mean matrix $\bar{\mathbf{X}}$ and covariance matrix $\mathbf{\Sigma_x}$ using $\mathbf{W}^i$.
3. Recover $\mathbf{Y}'$ with $\mathbf{W}^i$ and $\bar{\mathbf{X}}$.
4. Increase i by one.
5. Sample $\mathbf{X}^i$ using $\bar{\mathbf{X}}$ and $\mathbf{\Sigma_x}$.
6. Calculate mean matrix $\bar{\mathbf{W}}$ and covariance matrix $\mathbf{\Sigma_w}$ using $\mathbf{X}^i$.
7. Recover matrix $\mathbf{Y}'$ with $\bar{\mathbf{W}}$ and $\mathbf{X}^i$.
8. Sample $\mathbf{W}^i$ using $\bar{\mathbf{W}}$ and $\mathbf{\Sigma_w}$.
9. Calculate bias mean $\bar{\mathbf{m}}$ and variance $\tilde{\mathbf{m}}$ using $\mathbf{W}^i$ $\mathbf{X}^i$.
10. Sample bias $\mathbf{m}^i$ using $\bar{\mathbf{m}}$ $\tilde{\mathbf{m}}$.
11. Loop from step 2.

This can be graphically visualized at Figure 1. At every loop, when calculating the mean matrices $\bar{\mathbf{W}}$ $\bar{\mathbf{X}}$ (steps 2 and 6), we use the original matrix $\mathbf{Y}$, this leads to an improvement in the recovered values (better representing the original matrix with the observed values) and hence and improvement in the future sampled matrices.

Every time matrix $\mathbf{Y}'$ is calculated (steps 3 and 7) the missing values are recovered. At every recovering step the missing values are averaged with the previously recovered ones, Formula 2, where k is the step, $\bar{y}$ is the average of the previous values and y are the new recovered values. Using the average will lead to better results than just using the single-samples alone. Single-samples may recover well some of the values, nevertheless; averaging them will reduce the distance to the expected values, step by step, and at some point, even, match them.

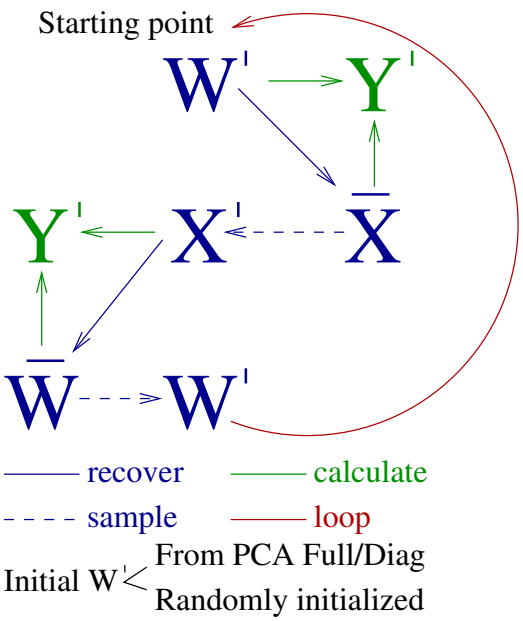

Figure 1: Sampling PCA process.

$$\bar{y}^{k+1} = \frac{k\bar{y}^k + y^{k+1}}{k+1} \tag{2}$$

3 Tests and Results

The Sampling PCA method was tested with an artificial data set and the MovieLens data set. For the MovieLens test the missing values were also predicted randomly to observe how close a random prediction is from the sampling approach, i.e.; to grossly measure the benefit of using sampling. With the artificial data we will focus on recovering all missing values while with MovieLens data only a subset of the missing values.

3.1 Artificial Data

The initial testing was done using artificially generated data. The artificial data consists on generating matrices $\mathbf{W}[m, c]$ (normally distributed $\mathcal{N}(0, 1)$, random values); $\mathbf{X}[c, p]$ (uniformly distributed $[0 \dots 1]$, random values) and, an additional noise matrix $\mathbf{N}[m, p]$ (normally distributed $\mathcal{N}(0, var)$ where *var* is given as an input). Matrix $\mathbf{Y}[m, p]$ is generated as $\mathbf{Y} = \mathbf{WX} + \mathbf{N}$. From matrix $\mathbf{Y}$ a given percentage of ratings is selected at random and set to NaN in matrix $\mathbf{Y}_t$, i.e.; set to be missing values[1].

Three data sets were generated with the following characteristics:

[1]Where m stands for Movies; p for People and c for Components.

		c=10	c=20	c=30
A	PCA_Full	0.264886	0.264909	0.264939
	Sampling	0.265208	0.265511	0.266457
B	PCA_Full	0.965070	0.865517	0.992878
	Sampling	0.959550	0.866838	0.989643
C	PCA_Full	1.238677	1.163651	1.238233
	Sampling	1.232581	1.160960	1.230279

Table 1: RMSE results on artificial data.

Set	m	p	c	Noise Variance	Missing Values
A	100	125	8	0.05	50%
B	150	200	15	0.3	70%
C	300	450	18	0.5	85%

Using the VB-PCA approach, PCA_FULL function, we recover $\mathbf{W}$ $\mathbf{X}$ and $\mathbf{m}$ (plus hyper-parameters) from matrix $\mathbf{Y}_t$. We do this using 10, 20 and 30 components. With the recovered matrices we run the Sampling PCA algorithm; 500 samples are generated from each input.

We can observe at Table 1, how the noise, size and proportion of missing values of the original matrix $\mathbf{Y}$ affect the quality of the recovered missing values. It is also noticeable that when the problem is simple, as it is in with data set A, PCA_FULL recovers the matrix with a small error, therefore; no improving can be expected, or achieved, when sampling. On the other hand, with data set C, where the missing values are many and the matrix is noisy and large the recovering achieved from PCA_FULL is just good but it is improved with the Sampling PCA algorithm. An important value affecting the results is the number of components, c. Because we *do not know* the original number of components we try with 10, 20 and 30, and notice that as we get closer to the original number of components our results improve. At Figure 2, are the sampling RMSE error progress through 500 samples compared to the PCA_FULL RMSE error using the best results within each data set.

From the artificial testing we can conclude, first; the number of components used play an important role and, second; as more complex is the problem better results can be expected when using Sampling PCA.

3.2 MovieLens Data

The MovieLens [7] data set consist of 100,000 ratings given by 943 users to 1682 movies. Each rating is a triplet, the value of the rating, the user giving the rating and the movie being rated. The ratings go from 1 to 5, not all movies have been rated nor all users have given rates. Having 100,000 ratings mean that less than 10% of the total possible triplets are available. The data set was divided into Training $\mathbf{Y}_t$ and Probing $\mathbf{Y}_p$ sets after empty columns/rows were removed, i.e.; users without ratings or movies no rated. The Training set is a matrix of 943x1674 and contains 95,000 ratings. The Probing set is a matrix of the same size but contains, only, 4999 ratings.

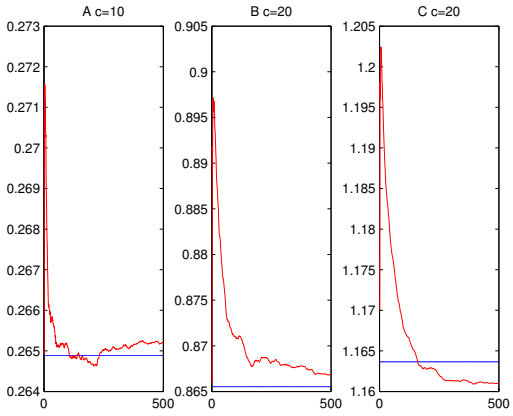

Figure 2: Sampling progress with artificial data.

		c=10	c=20	c=30
PCA_FULL	$\mathbf{Y}'$ vs $\mathbf{Y}_t$	0.743154	0.743614	0.744083
	$\mathbf{Y}'$ vs $\mathbf{Y}_p$	0.892615	0.892397	0.892211
PCA_DIAG	$\mathbf{Y}'$ vs $\mathbf{Y}_t$	0.762655	0.768235	0.768326
	$\mathbf{Y}'$ vs $\mathbf{Y}_p$	0.889250	0.889069	0.888687

Table 2: RMSE results on PCA_FULL/DIAG for Training and Probing data.

The first step consists on recovering matrices $\mathbf{W}^{'}$ $\mathbf{X}^{'}$ and $\mathbf{m}^{'}$ (and hyper-parameters) from matrix $\mathbf{Y}_t$ using the VB-PCA implementations PCA_FULL and PCA_DIAG, using 10, 20 and 30 as number of components. The RMSE of the recovered matrix, $\mathbf{Y}'$ against $\mathbf{Y}_t$ and $\mathbf{Y}_p$ can be seen at Table 2. PCA_FULL performed better with the Training matrix while PCA_DIAG was better for the Probing values. For both approaches more components mean worse results against the Training set but better against the Probing one.

With the recovered matrices and hyper-parameters we perform Sampling PCA. Two options are explored, the first option consist in using all the recovered data as starting point for sampling. The second option consists on only using the hyper-parameters; $\mathbf{W}^{'}$ $\mathbf{X}^{'}$ and $\mathbf{m}^{'}$ matrices are initialized with random values.

3.2.1 Sampling From PCA Full/Diag

In this first approach sampling is performed using the recovered matrices and hyper-parameters. For each set of variables 2000 samples are generated, the numeric results can be observed at Table 3. Results show an improvement compared to the $\mathbf{Y}'$ vs $\mathbf{Y}_p$ RMSE values at Table 2. The use of 20 components seems to return the best results, also, the use of PCA_DIAG shows better results. The best results (shadowed) represent a small improvement, less than 1% against the top result obtained using the VB-PCA approach alone (shadowed at Table 2). However; a small improvement for recovering missing values tasks its an important gain.

	c=10	c=20	c=30
PCA_FULL	0.888123	0.887418	0.887837
PCA_DIAG	0.884606	0.883733	0.884129

Table 3: RMSE results after sampling (2000 samples).

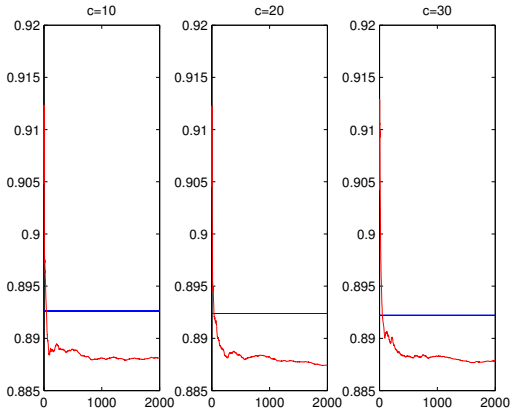

Figure 3: Sampling process using PCA_FULL data.

At Figure 3, we can observe the RMSE value of each sample through the 2000 samples taken, with different number of components and using PCA_FULL data as baseline; the values are compared against the RMSE of VB-PCA approach. At Figure 4, a similar plot is observable but in this case using PCA_DIAG data as baseline. For both Figures, in all sub-plots, we can notice that the sampling algorithm is unstable for the initial samples, the RMSE value *jumps* around the RMSE recovered from the VB-PCA approach. However; for the last hundreds of samples stabilization is noticeable, showing small differences after each sample.

3.2.2 Sampling Using Random Initialization

Another approach to perform Sampling PCA consist in only using the hyper-parameters recovered from PCA_FULL/DIAG. Matrices $\mathbf{W}^{'}$ $\mathbf{X}^{'}$ and $\mathbf{m}^{'}$ are randomly initialized (uniformly distributed values$[0 \ldots 1]$). This is possible because the algorithms used to recalculate matrices $\mathbf{W}'$ $\mathbf{X}'$ and $\mathbf{m}'$ and their covariances take into account the training matrix $\mathbf{Y}_t$. At each iteration of the sampling the matrices $\mathbf{W}'$ $\mathbf{X}'$ and $\mathbf{m}'$ values are updated to better fit $\mathbf{Y}_t$.

The initial samples will be highly deviated from the objective value, therefore; they can be eliminated before the real prediction is made. In our test we remove the initial 30 samples. Later, we generate 1000 new samples to make the predictions of the missing values. Again 10, 20 and 30 components are used and the hyper-parameters from PCA_FULL/DIAG. The Figure 5, shows the discarded samples and how spread they were compared to the final RMSE. The first 10 samples are the most disperse ones, latest

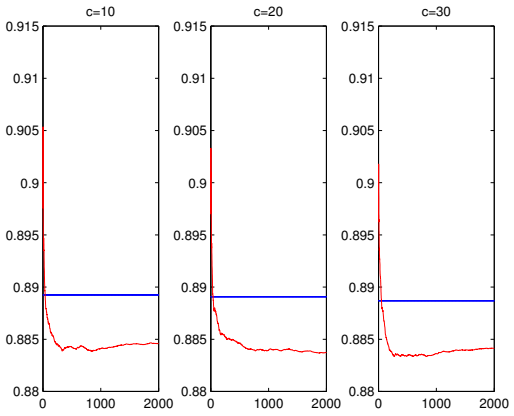

Figure 4: Sampling process using PCA_DIAG data.

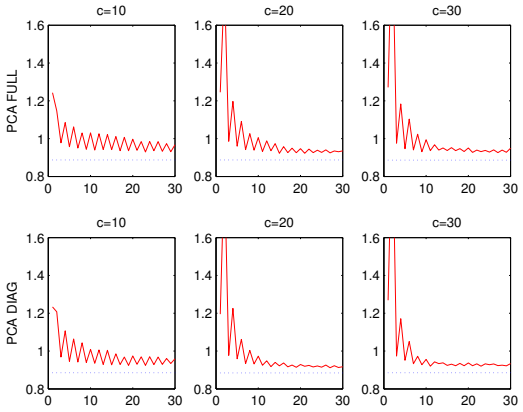

Figure 5: RMSE for discarded samples.

samples are more stable in their RMSE value, specially, when the number of components is 20 and 30.

Figure 6, shows the RMSE value at each sample during the sampling process. The results of the sampling process are at Table 4. The results are similar to those obtained using the first approach, however; its worth noticing that for PCA_FULL the results are better in all the instances and only half the samples were generated (the same hyper-parameters were used). This may be related on how the recovered matrices, learning $\mathbf{Y}_t$, directly affect the sampling process.

3.2.3 Random Guessing

The results obtained by Sampling PCA, under the MovieLens data set, are good. In all the runs the algorithm reduced the error of the recovered missing values compared to the VB-PCA approach. A way to compare the re-

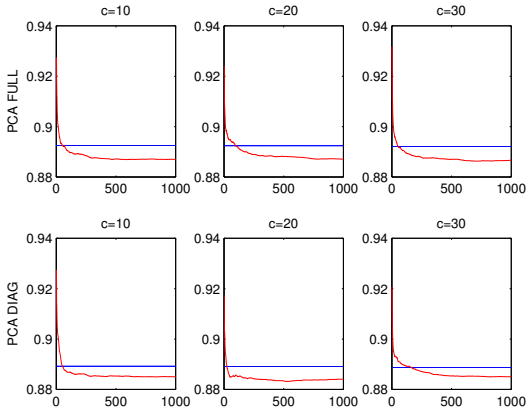

Figure 6: Sampling process for random initialization.

	c=10	c=20	c=30
PCA_FULL	0.887070	0.887121	0.886675
PCA_DIAG	0.885025	0.884074	0.885066

Table 4: RMSE results after sampling, random initialization.

sults and the real gain obtained by the use of the algorithm is to *guess* all the missing values and calculate the error of the guesses. The guessing was done in two ways. On one hand, the missing values are generated at random and the values are uniformly distributed $[1 \dots 5]$. On the other, the process is similar to the initialization explained at 3.2.2, where matrices $\mathbf{W}$ $\mathbf{X}$ and $\mathbf{m}$ are filled with uniformly distributed values $[0 \dots 1]$, using different number of components. The RMSE results of these random approaches are at Table 5. We notice, in general, a large improvement in the RMSE when using a formal methodology vs guessing.

4 Conclusions

This project lead to interesting results. The artificial tests let us know that small matrices with small portion of missing values are not easily improved by sampling. For the MovieLens test we observed that sampling improved the quality of the recovered missing values over VB-PCA using the later

1st	Test A	Test B	Test C
RMSE	1.685530	1.711611	1.698252
2nd	c=10	c=20	c=30
RMSE	1.433975	2.45591410	1.454340

Table 5: RMSE random guessing. 1st; random values [1..5] 2nd; random matrices [0..1]

as an initial step. We also noticed that the random initialization does not affect sampling and the results are good. The best results were obtained using PCA_DIAG and 20 components; the worst results were obtained using PCA_FULL and 10 components. A future improvement could be achieved rounding the recovered values that are outside the range of the expected ones, i.e.; values ≤ 1 to 1 and ≥ 5 to 5. A look at the recovered vector, for the best results, shows 6 values below 1 and 32 above 5.

The project, in general, helped me to better understand the ideas behind PCA, Sampling methods and treatment of large matrices. The topic was challenging but interesting with lots of learning.

5 Acknowledgments

- Alexander Ilin and Tapani Raiko for their support, time and tips.
- Luis De Alba is supported by the Programme AlBan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E07M402627MX

References

[1] Simon Funk. *Netflix update: Try this at home*. December 2006. http://sifter.org/∼simon/journal/20061211.html

[2] Ruslan Salakhutdinov and Andriy Mnih. *Probabilistic Matrix Factorizatio*n. 2008. Advances in Neural Information Processing Systems 20. Cambridge, MA. MIT Press.

[3] Tapani Raiko, Alexander Ilin and Juha Karhunen. *Principal Component Analysis for Large Scale Problems with Lots of Missing Values*. 2007. Proceedings of the 18th European conference on Machine Learning.

[4] Alexander Ilin, Tapani Raiko. *Practical Approaches to Principal Component Analysis in the Presence of Missing Values*. 2008. Helsinki University of Technology. Faculty of Information and Natural Sciences.

[5] Ruslan Salakhutdinov and Andriy Mnih. *Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo*. 2008. Proceedings of the 25th International Conference on Machine learning.

[6] Christopher M. Bishop. *Pattern recognition and Machine Learning*. Springer. 2006. Chapter 11 “Sampling Methods”.

[7] “MovieLens”. Movie Recommendations. GroupLens Research at the University of Minnesota. http://movielens.umn.edu
Dataset: http://www.grouplens.org/node/73