# Extending Self-Organizing Maps with Uncertainty Information of Probabilistic PCA

Dušan Sovilj, Tapani Raiko, Erkki Oja

***Abstract*— We introduce a probabilistic version of the self-organizing map (SOM) where we model the uncertainty of both the model vectors and the data. While uncertainty information about the data is often not available, this property becomes very useful when the method is combined in a hierarchical manner with probabilistic principal component analysis (PCA), where we do estimate uncertainty of the principal components and the weights. We apply the hierarchical model to the domain of collaborative filtering, where probabilistic PCA is a popular approach due to its robustness for tackling many missing values in the data. The main focus in this paper is for recommendation systems about movies, where the movie rating data matrix of size people times movies is available, but contains lots of missing values. The matrix is first decomposed into a matrix product of people times features and features times movies by PCA. Then we apply the probabilistic SOM to both of those matrices separately. The uncertainty is large when a person (or a movie) has only a few ratings. The experiments with Movielens and Netflix data show an improvement over traditional SOM.**

## I. Introduction

Collaborative filtering (CF) is the task of predicting preferences or producing personal recommendations by using other people's preferences. One such problem is the Netflix prize [1] problem which involves ratings for movies given by people. The task is to predict the rating for a certain (*person*, *movie*) pair for which the rating is unknown. All ratings are integer numbers ranging from 1 to 5. The data is split into training and validation sets for the same group of people and a group of movies, with the training part having only 1.2% of observed or actual ratings, while the rest 98.8% are missing.

The collaborative filtering has recently become popular with announcement of the Netflix prize and a lot of methods have already been tried in this domain, for a list of methods refer to websites [2], [3]. More recent methods in CF are employing large number of predictors and combining them into one big linear model which have proven to be the most effective. The earlier works are focused on improving or boosting the accuracy of single models [4], [5], which is the approach we adopt for this paper.

For this work, we use a combination of two widely known methods in machine learning: Principal Component Analysis (PCA) [6] and Self-Organizing Maps (SOM) [7]. Several researchers have used PCA to estimate the missing values in the data [8], [9], but none provide the reliability of those estimates. For that purpose, we use PCA in its probabilistic variant [10] as the first step of our approach. Probabilistic PCA has the advantage of returning the likelihoods of the estimates, indicating their reliability. The second step involves using two SOMs, one for the principal components and the other for the weight matrix (transformation), in their standard form for possible improvement. An extension of the SOM is also tested which uses likelihood information from the probabilistic PCA. We call this new version Probabilistic SOM, which has modified update rule for the weights of the map that incorporates obtained likelihoods. There are also many existing combinations of PCA and SOM. Typically they work such (see e.g. [11], [12], [13]) that each map unit of the SOM has a separate PCA model for the data vectors that are associated to it. The combination proposed here is of course rather different.

Modelling partially observed data often means that some elements of the data are observed while the others are missing. We are studying the generalization to the case where we know the uncertainty (e.g. variance) of each element in the data. Pearl [14] suggested to use so called virtual evidence for this situation in Bayesian networks. Raiko [15] studied the same issues in the context of variational Bayesian learning. Real applications where the uncertainty of each observation is explicitly known are rare. Ilin and Kaplan [16] studied the reconstruction of historical sea surface temperatures from mostly ship data, where the uncertainty at each grid point is estimated based on the number of measurements located in it during each month.

A method with the same name – probabilistic Self-Organizing Map – was used in [17] for facial recognition, but in a different setting. The SOM in its standard form is first used for testing image similarity, and a probabilistic model for the class distribution is associated with each map unit. Thus, the uncertainty is modelled only for the class distribution and not for the model vectors or observations as in our case. Another model with the same name was recently proposed in [18]. This model aims at preserving topological structure of the data clusters by optimizing likelihood based criterion on the map neurons. To facilitate likelihood optimization, neurons in SOM are modeled as multivariate Gaussian distributions. In our method, the aim is quite different. For the rest of the paper, we also use Probabilistic SOM as the name of our model to explicitly state the connection to uncertainty modelling.

The Generative-Topographic Mapping (GTM) [19] can also be considered to be a probabilistic version of the SOM. The GTM is a generative model with latent variables which are mapped nonlinearly to the original space. Compared to the proposed method, the GTM can only model depen-

Authors are from Aalto University School of Science and Technology, Department of Information and Computer Science, Konemiehentie 2, Espoo, Finland. Emails: dusans@cis.hut.fi (Dušan Sovilj), tapani.raiko@tkk.fi (Tapani Raiko), erkki.oja@hut.fi (Erkki Oja).

978-1-4244-8126-2/10/$26.00 

dencies between the observation vectors, but not between the weights, whereas in the proposed method, the weights and the latent variables are treated symmetrically. In other words, our approach includes one map for movies and one for people, whereas the GTM has only one latent space. Another difference is that in GTM, the nonlinear mapping is done directly to the high-dimensional observation space, whereas in the proposed method, the mapping between the observation space and feature space is linear. This should make the proposed approach more robust against overfitting.

The paper is organized as follows. Section II explains PCA and probabilistic variant, the standard SOM and discusses the probabilistic version with necessary equations. Section III shows the results of experiments performed on Movielens [20] and Netflix datasets. The first dataset has less ratings than Netflix, and allows for easier investigation of new methods in the domain of CF. Final thoughts on the proposed method are given in Section IV.

## II. Method

In this section, we briefly give the basics for two main methods to be merged: PCA and SOM (first two subsections), and introduce the notation that is used in this work. Then, the combination of these two methods is explained in the last two subsections.

### A. Principal Component Analysis

Principal Component Analysis (PCA) is a widely used technique for data analysis. It can be derived from different starting points and optimization criteria. The most important are: 1) minimization of sum-of-squares reconstruction cost; 2) finding mutually orthogonal directions in the data having maximal variance. Assuming there are $N$ data vectors in input space with $D$ dimensions, i.e. a $D \times N$ matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N]$, PCA decomposes the matrix $\mathbf{X}$ into

$$\mathbf{X} \approx \mathbf{AS}, \tag{1}$$

where $\mathbf{A}$ is a $D \times q$ matrix, $\mathbf{S}$ is a $q \times N$ matrix and $q \leq D \leq N$. Principal subspace methods [21] find $\mathbf{A}$ and $\mathbf{S}$ such that reconstruction error

$$\|\mathbf{X} - \mathbf{AS}\|_F^2 , \tag{2}$$

is minimized, with $F$ denoting Frobenius norm. Before PCA is applied, 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. However, the subspace spanned by the column vectors of the matrix $\mathbf{A}$ is unique. This spanned subspace is called principal subspace. In PCA, the column vectors are mutually orthogonal and have unit length and by taking any first $k \leq q$ columns a $k$–dimensional principal subspace is formed. There are many ways to determine the principal subspace, also called components, and the most common ones are Singular Value Decomposition, EM Algorithm and Subspace Learning Algorithm.

In the case of missing values present in the data $\mathbf{X}$, methods for PCA can be extended to cope with such situation (see e.g. [6]). Matrices $\mathbf{A}$ and $\mathbf{S}$ are computed using only observed values in the data. Missing values are estimated simply through reconstruction, i.e. Equation (1). If data matrix has both large number of dimensions and lots of missing values, overfitting can easily arise in such situation. One way to overcome overfitting is to penalize the model with a proper term which will restrict values in $\mathbf{A}$ and $\mathbf{S}$ to small numbers. Even stronger approach against overfitting is to use Variational Bayesian (VB) Learning framework for PCA. VB version of the PCA proposed in [22] approximates the joint posterior of the unknown quantities using a simple multivariate distribution. The rows of $\mathbf{A}$ and the columns of $\mathbf{S}$ are described *a posteriori* using independent Gaussian distributions. The means of these distributions can be used as point estimates of the parameters, while the covariances give at least a crude estimate of the reliability of these points estimates. In [10], a version where all parameters are assumed to be independent, was applied to the case of missing values. In this paper, we call it *the probabilistic PCA*.

### B. Self-Organizing Map

Self-Organizing Maps (SOM) [7] are neural networks for unsupervised learning schemes, in which output classes or responses are unknown. The neurons have a specified neighborhood structure in lower-dimensional space, arranged in a *lattice* consisting of 2 or 3 dimensions. In addition, all of the neurons have a corresponding weight vector in the input data space ($D$ dimensions).

Main idea behind SOM is to move map neurons in patches toward the current sample $\mathbf{x}_n$ under consideration and this is accomplished in two steps. First, for sample $\mathbf{x}_n$ we find the best-matching unit $b_i$ among all map neurons. Second, $b_i$ and its neighbors $Ne(b_n)$ in the lattice are moved towards the sample $\mathbf{x}_n$. Best-matching unit (BMU) is the neuron whose weight vector is closest to the sample $\mathbf{x}_n$. Finding BMU involves a metric to measure the similarity between points, and in SOM the Euclidean distance is widely used as a similarity measure. The neighborhood function $Ne(b)$ considers a lattice in order to find neighbors, and possible choices are a simple ball function or a Gaussian function. After constructing the neighborhood, all the neurons are moved toward the sample $\mathbf{x}_n$. This update is repeated for all samples in the dataset, and all these updates constitute 1 epoch. The algorithm then runs for certain number of epochs or it can stop in case of convergence. The whole setting is set up in Robbins-Monro procedure with a parameter that controls the degree of movement for neurons. This parameter, denoted $\alpha \in (0, 1]$, is called a learning rate, and is a monotonically decreasing function of epochs. Assume that map has $P$ neurons arranged in a lattice, then the update of the weight vectors of all neurons in epoch $t+1$ is done in two steps:

1) Find the BMU $c$ for a sample $\mathbf{x}_n$

$$c = \arg\min_p \|\mathbf{x}_n - \mathbf{m}_p\| , \quad p = 1, ..., P \tag{3}$$

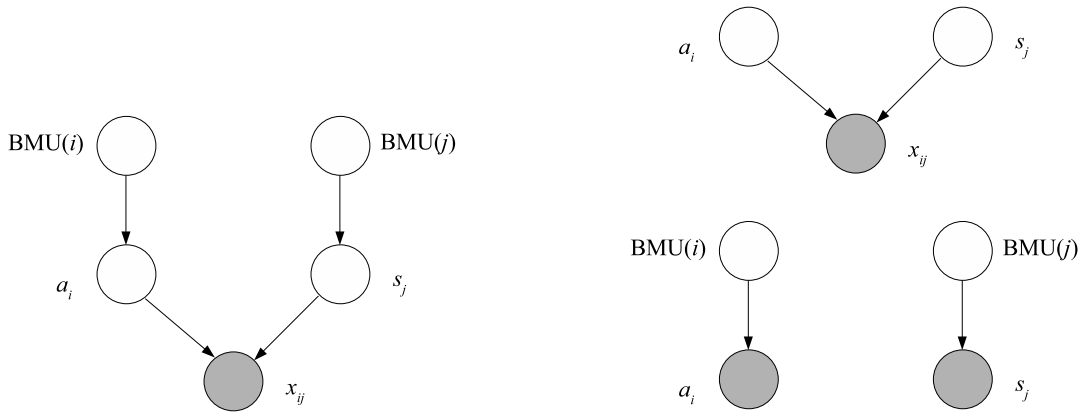


Fig. 1. Graphical models for the PSOM: complete model (left) and separated models (right).

2) Update the weights of each neuron $\mathbf{m}_p$

$$\mathbf{m}_p(t+1) = \mathbf{m}_p(t) + \alpha(t) h_{cp}(t) \left[\mathbf{x}_n - \mathbf{m}_p(t)\right] \quad (4)$$

In Equation (4), $\alpha(t)$ is the learning rate at time step $t$, $h_{cp}(t)$ is neighboring kernel around winning neuron $c$. This kernel computes the influence of the winning neuron $c$ on all the neurons in the map and is a non-increasing function of time $t$ and distance from $c$. Equations (3) and (4) constitute one epoch or time step, which is repeated until stopping criterion is fulfilled.

### C. Probabilistic Self-Organizing Map (PSOM)

The update rule given by Equation (4) moves the BMU toward the sample $\mathbf{x}_n$, which we can think of as the center of mass, and thus the center of attraction. The SOM framework can be extended to have a probabilistic approach, where the weights of neurons and samples are random variables. The simplest way is to assume Gaussian distribution in both cases, that is, the weights are random variables with $\mathcal{N}(\boldsymbol{\mu}_{Bk}, \Sigma_{Bk})$ for $k = 1, ..., K$, and each sample $\mathbf{x}_n$ has its own distribution $\mathcal{N}(\boldsymbol{\mu}_{Sn}, \Sigma_{Sn})$ for $n = 1, ..., N$ where both covariance matrices $\Sigma_k$ and $\Sigma_n$ are diagonal. The update rule in Equation (4) requires a vector which acts as an attractor. In the probabilistic framework this vector is computed by applying Bayes rule on two mentioned Gaussian distributions. The distribution for the weights $\mathcal{N}(\boldsymbol{\mu}_{Bk}, \Sigma_{Bk})$ could be considered as a *prior*, while the distributions over samples $\mathcal{N}(\boldsymbol{\mu}_{Sn}, \Sigma_{Sn})$ are the *likelihood* estimates (as suggested in [14], [15]), and through the Bayes rule we obtain the posterior distribution for weights.

Since both of them have diagonal covariance matrices, the product is a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_O, \Sigma_O)$ where each dimension is a product of one dimensional Gaussians from $\mathcal{N}(\boldsymbol{\mu}_{Bk}, \Sigma_{Bk})$ and $\mathcal{N}(\boldsymbol{\mu}_{Sn}, \Sigma_{Sn})$. Let us denote diagonal of $\Sigma_{Bk}$ as a vector $\left[\sigma^2_{Bk1}, \sigma^2_{Bk2}, ..., \sigma^2_{BkD}\right]$, and similarly for $\Sigma_{Sn}$ we have $\left[\sigma^2_{Sn1}, \sigma^2_{Sn2}, ..., \sigma^2_{SnD}\right]$. The means of distributions are $\boldsymbol{\mu}_{Bk} = [\mu_{Bk1}, ..., \mu_{BkD}]$ and $\boldsymbol{\mu}_{Sn} = [\mu_{Sn1}, ..., \mu_{SnD}]$. The product of two distributions $\mathcal{N}(\boldsymbol{\mu}_{Bk}, \Sigma_{Bk})$ for some $k = 1, ..., K$ and $\mathcal{N}(\boldsymbol{\mu}_{Sn}, \Sigma_{Sn})$ for some $n = 1, ..., N$ is a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_O, \Sigma_O)$ with mean $\boldsymbol{\mu}_O$ given by Equation (6) and covariance matrix $\Sigma_O$ with diagonal elements given by Equation (5).

$$\sigma^2_{Oj} = \left( \frac{1}{\sigma^2_{Bkj}} + \frac{1}{\sigma^2_{Snj}} \right)^{-1}, \quad j = 1, \ldots, D, \quad (5)$$

$$\mu_{Oj} = \left( \frac{\mu_{Bkj}}{\sigma^2_{Bkj}} + \frac{\mu_{Snj}}{\sigma^2_{Snj}} \right) \sigma^2_{Oj}, \quad j = 1, \ldots, D. \quad (6)$$

Thus, we can compute the *posterior* probability of the weight vectors of neurons in SOM and we need parameter values that characterize the distributions $\mathcal{N}(\boldsymbol{\mu}_{Bk}, \Sigma_{Bk})$ and $\mathcal{N}(\boldsymbol{\mu}_{Sn}, \Sigma_{Sn})$. For $\boldsymbol{\mu}_{Sn}$ we take the sample $\mathbf{x}_n$ itself, while the $\sigma^2_{Bnj}$ are likelihoods taken from probabilistic PCA. For $\mathcal{N}(\boldsymbol{\mu}_{Bk}, \Sigma_{Bk})$ we have $\boldsymbol{\mu}_{Bk} = \mathbf{m}_p(t)$, that is, the mean is set to the weights of the BMU in the input space domain. For the covariance matrix $\Sigma_{Bk}$ we do not have any information, and it has to be computed based on some heuristic. Instead of having covariance matrix for all neurons in SOM, which would increase number of parameters drastically, we resort to having only one covariance matrix $\Sigma_{B0}$ that is shared across all map neurons. In our case, we compute the $j$-th element on the main diagonal of $\Sigma_{B0}$ as

$$\sigma^2_{B0j} = \frac{1}{N} \sum_{k=1}^{N} \left[ (\mu_{Bkj} - \mu_{Oj})^2 + \sigma^2_{Oj} \right], \quad j = 1, \ldots, D. \quad (7)$$

The initial value for $\sigma^2_{B0j}$ is computed from the values obtained with probabilistic PCA, taking into account all the samples:

$$\sigma^2_{B0j} = \frac{1}{N} \sum_{n=1}^{N} (\mu^2_{Snj} + \sigma^2_{Snj}), \quad j = 1, \ldots, D. \quad (8)$$

With Equations (7) and (8), the algorithm for probabilistic SOM first finds the BMU of a sample $\mathbf{x}_n$, then computes new variance (5) and mean (6), and finally uses the modified update rule for the weights using following equation:

$$\mathbf{m}_p(t+1) = \mathbf{m}_p(t) + \alpha(t) h_{cp}(t) \left[ \boldsymbol{\mu}_O - \mathbf{m}_p(t) \right]. \quad (9)$$

Comparing Equations (4) and (9), the sample $\mathbf{x}_n$ is replaced with posterior mean of the weights (Equation (6)), while the learning rate and neighboring function remain the same. Thus, we are moving the BMU $\mathbf{m}_p$ of a sample toward its most probable direction $\boldsymbol{\mu}_O$, instead of moving it toward the sample.

The only parameter left to be updated is prior variance for the weights given by Equation (7). This is done after completing one epoch and when all samples contributed to updating weights of neurons.

When a sample $\mathbf{x}_n$ is processed, that is, it contributed to neurons' weights, the BMU for this sample is remembered, as well as the posterior mean $\boldsymbol{\mu}_O^{(n)}$. Denoting the BMU for sample $\mathbf{x}_n$ as $\boldsymbol{\mu}_{Bk}^{B(n)}$, Equation (7) becomes

$$\sigma_{B0j}^2 = \frac{1}{N} \sum_{n=1}^{N} \left[ (\mu_{Bkj}^{B(n)} - \mu_{Oj}^{(n)})^2 + \sigma^{2(n)}_{Oj} \right], \; j = 1, \ldots, D. \quad (10)$$

Thus, $\boldsymbol{\mu}_O^{(n)}$ is paired with the BMU $\boldsymbol{\mu}_{Bk}^{B(n)}$ at the time when sample $\mathbf{x}_n$ was used to update the weights. In other words, once the BMU is found it will remain fixed for the current epoch, even if there is closer neuron to the sample $\mathbf{x}_n$ after completing the epoch. Algorithm 1 present the idea of training the PSOM once we obtain information from probabilistic PCA.

**Algorithm 1** Probabilistic SOM

**Input:** $\mu_{Snj}, \sigma^2_{Snj}$, SOM parameters
**Output:** trained map
**main:**

```
initialize map
initialize σ²_{B0j} (Equation (8))
for each epoch do
    for each sample x_n do
        μ^{B(n)}_{Bk} ← BMU of x_n
        compute σ^{2(n)}_{Oj} (Equation (5))
        compute μ^{(n)}_{Oj} (Equation (6))
        update μ_{Bk} toward μ^{(n)}_O (Equation (9))
    end for
    update σ²_{B0j} (Equation (10))
end for
return map weights μ_{Bk}
```

### D. Combination of PCA and SOM

The complete model is given in Figure 1. The PSOM acts as a prior for a given dataset $\mathbf{X}$. In our case, datasets are reconstruction matrices $\mathbf{A}$ and $\mathbf{S}^{\mathrm{T}}$ obtained with probabilistic PCA. In principle, the whole model can be learned together, but in this paper it is done separately as displayed on the same Figure 1.

Note that the proposed method is not a generative model, since the best matching unit is not found based on a probabilistic model, and because the SOM learning is not probabilistic. On the other hand, when the SOMs have been learned and the best matching unit for each $i$ and $j$ is fixed, the rest becomes a generative model. In this sense, the SOMs can be seen here as priors.

## III. Experiments

The PSOM and standard SOM are first compared on the Movielens 100k dataset with 100 000 movie ratings for 943 persons and 1682 movies. When considered as a full matrix, the dataset has 93.7% of the values missing. The task is to predict the missing value for a given (person, movie) pair. The Movielens 100k dataset is first randomly split into training and validation parts, each having 95000 and 5000 ratings respectively. The dataset is formed to have more columns than rows, that is, final dataset has 943 rows (persons) and 1682 columns (movies). The probabilistic PCA is first applied to obtain the reconstruction matrices $\mathbf{A}$ and $\mathbf{S}$ using the training set only. The PCA is run with 10 components as the upper limit and for 1500 iterations, where the number of components is decided by validating the reconstructed values on the validation set. After training, the final training and validation root mean-square errors (RMSE) were 0.7678 and 0.8878, respectively.

The result of probabilistic PCA are two matrices $\mathbf{A}_{943\times10}$ and $\mathbf{S}_{10\times1682}$ with likelihood estimates for all elements of both matrices. The $\mathbf{A}$ matrix contains feature vectors for each *person* in the new principal space. Similarly, $\mathbf{S}^{\mathrm{T}}$ represents feature vectors for each *movie* of the full matrix $\mathbf{X}$. The advantage of probabilistic PCA is the information about the reliability of the values inside these two matrices. For a person $i$ with few ratings (a row in $\mathbf{X}$), the corresponding feature vector $\mathbf{a}_i$ in $\mathbf{A}$ has large uncertainty $\boldsymbol{\sigma}_i^2$. Same reasoning applies for a movie with few ratings (a column in $\mathbf{X}$), where the corresponding vector in $\mathbf{S}^{\mathrm{T}}$ has large variance. After this step, both standard SOM and PSOM are used to visualize and quantify $\mathbf{A}$ and $\mathbf{S}^{\mathrm{T}}$. The parameters common to both SOM types are initialized to the same values and include:

- Learning rate $\alpha = 0.85$
- Learning rate is decreasing linearly
- Topology is a lattice represented as rectangular sheet
- Gaussian neighborhood function
- Initial radius set to half the map size
- Final radius set to 0.1
- Random initialization of the weights
- Map training done for 1500 epochs

A set of different values for learning rate was tested, but experiments showed that the larger values produced better results in terms of RMSE. Other important parameter is the size of the map, and the tested values ranged from 100 to 900 neurons arranged into a lattice with equal width and height. After training the SOMs, each sample in $\mathbf{A}$ and $\mathbf{S}^{\mathrm{T}}$ is replaced with the weight of its best matching unit giving $\mathbf{A}_r$ and $\mathbf{S}_r^{\mathrm{T}}$, and the initial dataset is then reconstructed with

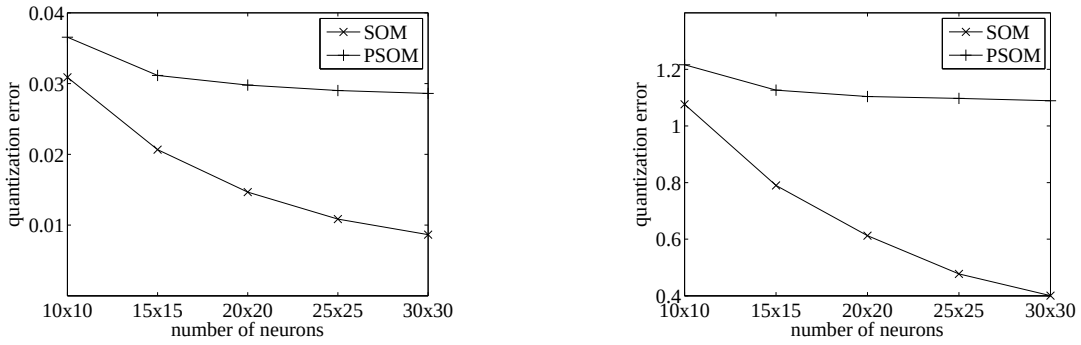


Fig. 2. Quantization error for two SOM types on $\mathbf{A}$ matrix (left) and $\mathbf{S}^{\mathrm{T}}$ matrix (right).

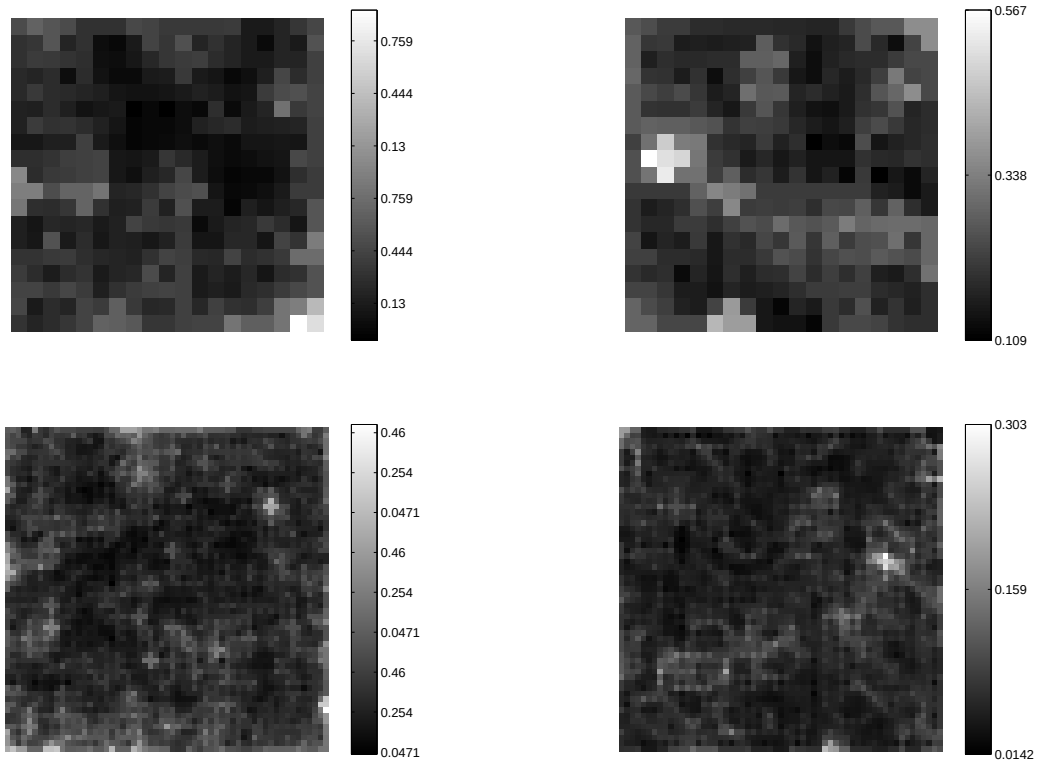


Fig. 3. U matrices for the trained two SOM types on the $\mathbf{A}$ matrix. In the top row, the map size is $10 \times 10$, while in the bottom row, the map size is $30 \times 30$. Left column is standard SOM, and right column is Probabilistic SOM.

$$\mathbf{X} \approx \mathbf{A}_r \mathbf{S}_r + \mathbf{M}_{943\times 1}\mathbf{1}_{1\times 1682}$$

Vector $\mathbf{M}$ in the above equation is the sample mean of data matrix, which is obtained as maximum likelihood solution of log probability of the parameters [22], while $\mathbf{1}_{1\times 1682}$ is just a row vector of ones. The reconstructed matrix contains the values for elements of the validation set, which is used to measure the performance of all three methods: probabilistic PCA, SOM and PSOM. The RMSE for both SOM types is shown in Table I. Since SOM is used on $\mathbf{A}$ and $\mathbf{S}$ the experiments are done by using the same map size for both matrices and then reconstructing the matrix. For example, the first column in Table I shows the RMSE when both $\mathbf{A}$ and $\mathbf{S}^{\mathrm{T}}$ are given to a map of size $10 \times 10$.

TABLE I

RMSE FOR SOM AND PSOM

| Map size | $10 \times 10$ | $15 \times 15$ | $20 \times 20$ | $25 \times 25$ | $30 \times 30$ |
|---|---|---|---|---|---|
| SOM | 0.9302 | 0.9199 | 0.9123 | 0.9040 | 0.8995 |
| PSOM | 0.9216 | 0.9089 | 0.9013 | 0.8977 | 0.8992 |

As can be seen from Table I, the RMSE decreases when the number of neurons increases, as expected since more neurons allow for a more refined quantization. One final experiment involved using map sizes that have more neurons than samples. In this case, map sizes were $31\times 31$ and $51\times 51$

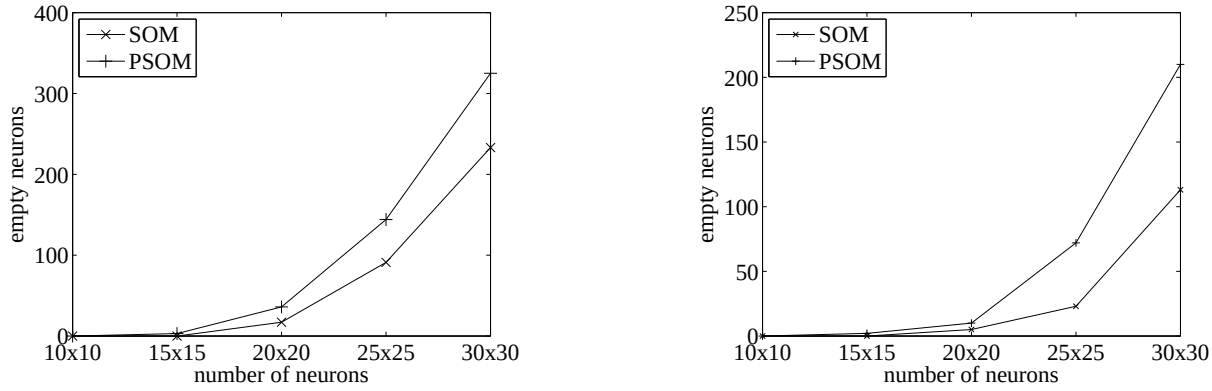


Fig. 4. Number of empty neurons when training SOMs for $\mathbf{A}$ matrix (left) and $\mathbf{S}^{\mathrm{T}}$ matrix (right).

for matrices $\mathbf{A}$ and $\mathbf{S}^{\mathrm{T}}$ respectively. The RMSE on validation data were 0.8918 for SOM and 0.8932 for PSOM.

Table I shows advantage of PSOM over SOM for smaller sizes maps. In order to see the differences in the BMUs of two map types we compare quantization error (QE) and U matrices. QE is the average distance between samples and their respective BMU, which is computed for all samples. U matrix is another way of visualizing the arrangement of neurons of the trained map. The U matrix has double the number of elements as there are neurons in the map. Each element represents the average distance of a neuron $m_p$ to its neighbors in the map $Ne(m_p)$, computed in the input $D$-dimensional space.

Comparing quantization error in Figure 2, we see that in PSOM it is always larger than in SOM, since the weights updates are no longer in direction toward the sample itself, but towards $\boldsymbol{\mu}_O$, which is likely not to be on the same direction between the BMU and the sample. The QE is in some cases larger by a factor of 3, but this eventually has positive effect when reconstructing the data.

Examining U matrices for both SOM types and different map sizes in Figure 3, the distribution of neurons for PSOM is more spread out, and neurons clearly form more clusters. On the other hand, neurons become more equidistant as the their number increases which is shown in the same figure. Also worth mentioning are the actual distances in U matrices. In all the experiments with different map sizes the average distance between neurons in PSOM is always less than in standard SOM.

Figure 4 shows the number of 'empty' neurons, that is, the neurons which are not the BMU for any of the samples. For all map sizes the number of empty neurons is larger for PSOM than for SOM, which would also explain the higher quantization error for PSOM, since less neurons are used to explain the samples.

Finally, Figure 5 shows the evolution of the posterior variance for only one component. The result is almost the same for all other components (not shown), exhibiting the same decreasing tendency towards very small values. These small values actually prevent the PSOM from moving the neurons with a considerable degree, and can also increase the convergence speed. This side effect remains to be further explored.

The experiments using Netflix data are done in the same manner, except that certain parameters have been altered to more adequately represent the data. Probabilistic PCA is first used to obtain the decomposition of the data into two matrices, with the number of principal components set to 50 and the number of iterations set to 1500. The number of components is chosen based on the probe RMSE, and the final RMSE on the Netflix probe data is 0.9055. Then, both SOM and PSOM are used to cluster new data ($\mathbf{A}$ and $\mathbf{S}^{\mathrm{T}}$). For $\mathbf{A}$ and $\mathbf{S}^{\mathrm{T}}$, map sizes of $75 \times 75$ and $75 \times 50$ are used respectively. The parameters for training the SOMs are same as for the Movielens data, except that in the Netflix case, a batch version of SOM algorithm is used that does not require a learning rate parameter $\alpha$.

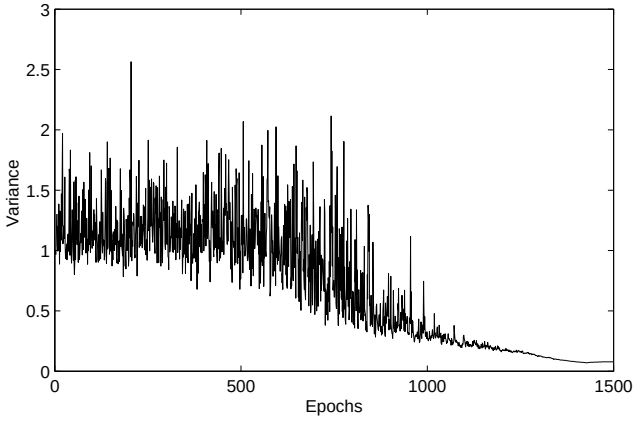


Fig. 5. Evolution of the variance of one component ($\sigma^2_{B01}$) for PSOM on $\mathbf{S}^{\mathrm{T}}$ matrix.

After the rows of both matrices have been replaced with their respective BMUs, the RMSE for the probe data is 0.9754 for the standard SOM and 0.9665 for the PSOM. As is the case for Movielens data, PSOM is able to give better reconstruction error than standard SOM. On the other

hand, there is evident degradation in performance since data is replaced with their nearest approximations. The increase in RMSE is quite higher for Netflix than for Movielens, as the number of map units in the SOM is very small compared to the actual number of data vectors (for $\mathbf{S}^{\mathrm{T}}$ its $75 \cdot 50 = 3750$ map units compared to roughly 480000 data samples). The quantization error of the two SOM types has the same trend as in the Movielens case: for PSOM it is always higher than for SOM, since the change is not toward the samples themselves.

## IV. Conclusion

In this paper, we presented an extended version of SOM using the likelihoods from probabilistic PCA method. Comparing results of both the PSOM and SOM, the extended version with likelihood information finds better solution in terms of neurons' weight adjustments. The update rule of standard SOM moves neurons towards samples themselves, while probabilistic SOM uses additional information and updates the center of attraction accordingly. New update equations try to fit a map which has evenly spread out neurons in the input space. For maps of various sizes the probabilistic approach always gives slightly better results, where the difference starts to diminish and eventually turns negative (combination of $31 \times 31$ SOM for $\mathbf{A}$ and $51 \times 51$ SOM for $\mathbf{S}^{\mathrm{T}}$) as the number of neurons increases. Since PSOM has more empty neurons it can be effectively used for data compression, while retaining the good spread of neurons over SOM. The drawback of the proposed extension are small values for the variances in the later stage of the algorithm, preventing significant updates to the weights of the map. One possible way of preventing this effect is to use validation data to determine this parameter.

Even though the proposed extension of the SOM has negative impact on the final performance, it still can be used in an ensemble with other models. This is the approach of many top teams in the Netflix competition. With ensemble approach, the performance of each separate method is not as important as their wide variety for the performance of the ensemble.

## References


[1] "Netflix prize webpage," http://www.netflixprize.com.

[2] "List of datasets, software and articles maintained by Jun Wang," 2007, http://ict.ewi.tudelft.nl/∼jun/CollaborativeFiltering.html.

[3] "List of research articles maintained by James Thornton," 2005, http://jamesthornton.com/cf/.

[4] S. Ding, S. Zhao, Q. Yuan, X. Zhang, R. Fu, and L. Bergman, "Boosting collaborative filtering based on statistical prediction errors," in *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*. New York, NY, USA: ACM, 2008, pp. 3–10.

[5] R. J. Melville., Melville M. and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Eighteenth national conference on Artificial Intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 187–192.

[6] I. T. Jolliffe, "Principal component analysis," in *Principal Component Analysis*. New York: Springer Verlag, 1986.

[7] T. Kohonen, *Self-Organizing Maps*, 2nd ed., ser. Springer Series in Information Sciences, 30. Springer, 1997.

[8] K. Honda, N. Sugiura, H. Ichihashi, and S. Araki, "Collaborative filtering using principal component analysis and fuzzy clustering," in *WI '01: Proceedings of the First Asia-Pacific Conference on Web Intelligence: Research and Development*. London, UK: Springer-Verlag, 2001, pp. 394–402.

[9] D. Kim and B.-J. Yum, "Collaborative filtering based on iterative principal component analysis," *Expert Systems with Applications*, vol. 28, no. 4, pp. 823–830, May 2005. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2004.12.037

[10] 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)*, ser. Lecture Notes in Artificial Intelligence, vol. 4701. Warsaw, Poland: Springer-Verlag, Berlin, September 2007, pp. 691–698.

[11] J. L. Alba Castro, A. Pujol, and J. J. Villanueva, "Novel SOM-PCA network for face identification," in *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation IV*, 2001, pp. 186–194.

[12] E. López-Rubio, J. M. Ortiz-de Lazcano-Lobato, and D. López-Rodríguez, "Probabilistic pca self-organizing maps," *IEEE Transactions on Neural Networks*, vol. 20, no. 9, pp. 1474–1489, 2009.

[13] E. Lopez-Rubio, J. Munoz-Perez, and J. A. Gomez-Ruiz, "A principal components analysis self-organizing map," *Neural Networks*, vol. 17, pp. 261–270, 2004.

[14] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[15] T. Raiko, "Partially observed values," in *Proc. Int. Joint Conf. on Neural Networks (IJCNN 2004)*, Budapest, Hungary, 2004, pp. 2825–2830.

[16] A. Ilin and A. Kaplan, "Bayesian pca for reconstruction of historical sea surface temperatures," in *Proc. of the Int. Joint Conf. on Neural Networks (IJCNN 2009)*, Atlanta, USA, 2009, pp. 1322–1327.

[17] G. Lefebvre and C. Garcia, "A probabilistic self-organizing map for facial recognition," in *ICPR*, 2008, pp. 1–4.

[18] S.-S. Cheng, H.-C. Fu, and H.-M. Wang, "Model-based clustering by probabilistic self-organizing maps," *IEEE Transactions on Neural Networks*, vol. 20, no. 5, pp. 805–826, 2009.

[19] C. M. Bishop and C. K. I. Williams, "GTM: The generative topographic mapping," *Neural Computation*, vol. 10, pp. 215–234, 1998.

[20] "Movielens database webpage," http://www.movielens.org/.

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

[22] C. M. Bishop, "Variational principal components," in *In Proceedings of Ninth International Conference on Artificial Neural Networks, ICANN99*, 1999, pp. 509–514.