# Paper One

## Probabilistic Matrix Factorization

**Motivation**

For recommendation systems, collaborative filtering is a common used algorithm. But it can neither work well on very large dataset nor deal with users with very few ratings. Therefore, authors want to propose a novel algorithm, Probabilistic Matrix Factorization(PMF), to learn a small number of unobserved factors of users. The matrix factorization can help us to get a low-dimensional representation, while the usual way like SVD cannot work on sparse dataset. Thus, PMF introduced by the authors aims to solve this problem. Moreover, the authors also show us how PMF can be controlled to scale up and down automatically.

**Main Theory**

Firstly, I want to introduce the common Matrix Factorization:

Many collaborative filtering algorithms are based on low-dimensional factor models, since they can handle large dataset and save storage memory. Moreover, they can work better when some users rate movies rarely. The loss function of common Matrix Factorization is: $\min_{U,V} \sum_{(i,j)\in\kappa} (R_{ij} - U_i^T V_j)^2 + \lambda(\|U_i\|^2 + \|U_j\|^2)$, the latter two term are penalization term to avoid overfitting problem.

Here, the authors introduce probability model into common MF to build up a new model – PMF(Probabilistic Matrix Factorization):

The whole model is based two main assumptions:

1. The observation noise obeys the Gaussian distribution, thus its probability density function is: $p(R|U,V) = N(\hat{R}, \sigma^2) = N(U^T V, \sigma^2)$
2. The users' features and movies' features obey the Gaussian distribution, and their mean value are zero, thus the probability density function are: $p(U) = N(0, \sigma_U^2)$, $p(V) = N(0, \sigma_V^2)$

Therefore, according to Bayes formula and the fact that U、V are distributing independently, we can obtain the following equation:

$$p(U,V|R) = p(U,V,R)/p(R) \propto p(U,V,R) = p(R|U,V)p(U)p(V)$$

from which we can express p(U,V|R). Obviously, our goal is to maximize the p(U,V|R), which is equal to $\ln p(U,V|R) = \ln p(R|U,V) + \ln p(U) + \ln p(V)$

(It is a method that we usually use by doing the log operation on both sides in order to make multiplication operation into addition operation.)

The log-expression of Gaussian function is: $\ln p(x) = -\ln(\sqrt{2\pi}\sigma) - \frac{(x-\mu)^2}{2\sigma^2}$

Thus we can obtain the cost function: $E(U,V) = \frac{(R-U^TV)^2}{2\sigma^2} + \frac{U^TU^2}{2\sigma_U^2} + \frac{V^TV^2}{2\sigma_V^2}$ , which is equal to $E(U,V) = \frac{1}{2}(R-U^TV)^2 + \frac{\lambda_U}{2}U^TU^2 + \frac{\lambda_V}{2}V^TV^2$ .

The above deduction is done for simplification. The complete version of cost function

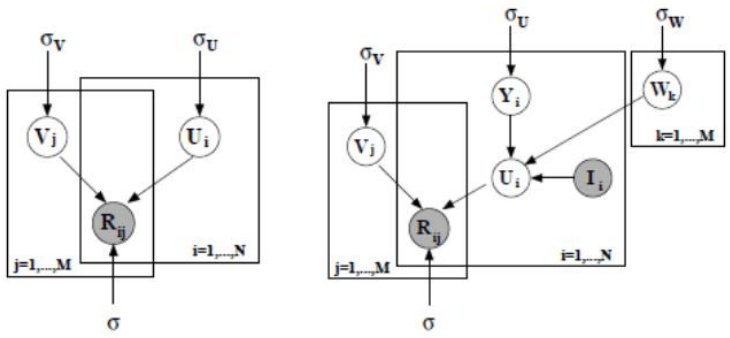
$$E(U,V) = \frac{1}{2}\sum_{ij} I_{ij}(R_{ij} - U_i^TV_j)^2 + \frac{\lambda_U}{2}\sum_i U_i^TU_i^2 + \frac{\lambda_V}{2}\sum_j V_j^TV_j^2$$

The last, authors want to make the rating score into [0,1], thus they use sigmoid function to constrain UV and PMF's final cost function is:

$$E(U,V) = \frac{1}{2}\sum_{ij} I_{ij}(R_{ij} - g(U_i^TV_j))^2 + \frac{\lambda_U}{2}\sum_i U_i^TU_i^2 + \frac{\lambda_V}{2}\sum_j V_j^TV_j^2$$

**Summary**

The authors introduce the probability model into the common Matrix Factorization model, making it more robust. More importantly, the paper also illustrates constrained PMF model, which is a little different from PMF as the picture shows:




PMF constrainted PMF

The only difference is $U_i = Y_i + \frac{1}{\sum_k I_{ik}}\sum_k I_{ik}W_k$ , or we can rewrite U into:

$p(U) = N(\frac{\sum_{k=1}^{M} I_{ik}W_k}{\sum_{k=1}^{M} I_{ik}}, \sigma_U^2)$ , which takes the consideration of the users with few ratings.

Therefore, the constrained version PMF is more robust than PMF. The latter one is going to ignore the influence of the users with few ratings, which seems okay, but the recommender system goes bad when it recommend something for the users with few ratings.

# Paper Two

## Collaborative Deep Learning for Recommender Systems

**Motivation**

Nowadays, recommender algorithms can be classified into three classes: content-based methods, collaborative filtering methods and hybrid methods. As for collaborative filtering, there exist two problems which are cold start and sparse ratings. Therefore, collaborative filtering method is supposed to utilize the auxiliary information thus becoming hybrid method. The authors want to introduce such a novel hybrid method combining collaborative filtering methods with deep learning methods based on Bayesian model.

**Related Theory**

Collaborative Filtering(CF): It is a method based on the similarity of users-users or items-items. Take users-users collaborative filtering for example, the algorithm needs users rating matrix as an input.

1. Firstly, computing the similarity of each pair of users based on the Person Coefficient:

$$\boldsymbol{sim}(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{s\in S_{xy}}(\boldsymbol{r}_{xs}-\overline{\boldsymbol{r}_{\boldsymbol{x}}})(\boldsymbol{r}_{ys}-\overline{\boldsymbol{r}_{\boldsymbol{y}}})}{\sqrt{\sum_{s\in S_{xy}}(\boldsymbol{r}_{xs}-\overline{\boldsymbol{r}_{\boldsymbol{x}}})^2}\sqrt{\sum_{s\in S_{xy}}(\boldsymbol{r}_{ys}-\overline{\boldsymbol{r}_{\boldsymbol{y}}})^2}}$$

2. If we want to recommend one item to the specific user j, we find the top-k users similar to the user j
3. Then, we compute the ratings of the items that user j did not buy:

$$r_{xi} = \frac{\sum_{y\in N} s_{xy}\cdot r_{yi}}{\sum_{y\in N} s_{xy}}$$

4. Finally, we recommend the item with highest rating to user j.

Stacked Denoising Autoencoders(SDAE): It is an typical autoencoder model, which is a neural network, learning input features and trying to output the same features as similar as possible. The weights of hidden layer in autoencoder can be seen as the digest of the features since they can transfer the input features to the same output features.

SDAE is a strengthened autoencoder model since noises are introduced into the model. Denoising Autoencoder aims to raise the ability of generalization, and the stacked denoising autoencoder has many hidden layers to enhance the training ability. The model is described as:

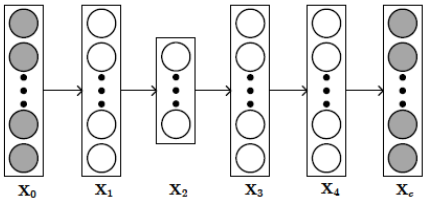



**Figure 2: A 2-layer SDAE with $L = 4$.**

And the cost function is $\min_{\{\mathbf{W}_l\},\{\mathbf{b}_l\}} \|\mathbf{X}_c - \mathbf{X}_L\|_F^2 + \lambda \sum_l \|\mathbf{W}_l\|_F^2,$ , where Xc is considered as the clean input, and X0 is corrupted input. Therefore, the former term is going to learn the XL similar to Xc as much as possible, thus becoming more robust. The latter term is a regularization term, which is trying to penalize the weights and avoid overfitting problem.

**Main Theory**

The generative process of CDL based on Bayesian SDAE is defined as follows:

1. For each layer $l$ of the SDAE,
   (a) For each column $n$ of weight matrix $W_l$, draw $W_{l,*n} \sim \mathcal{N}(0, \lambda_w^{-1}\mathbf{I}_{D_l})$.
   (b) Draw the bias vector $b_l \sim \mathcal{N}(0, \lambda_w^{-1}\mathbf{I}_{D_l})$.
   (c) For each row $i$ of $X_l$, draw $X_{l,i*} \sim \mathcal{N}(\sigma(X_{l-1,i*}W_l + b_l), \lambda_s^{-1}\mathbf{I}_{D_l})$.
2. For each item $i$,
   (a) Draw a clean input $X_{c,i*} \sim \mathcal{N}(X_{L,i*}, \lambda_n^{-1}\mathbf{I}_{I_i})$.
   (b) Draw a latent offset vector $\epsilon_i \sim \mathcal{N}(0, \lambda_v^{-1}\mathbf{I}_D)$ and set the latent item vector: $V_i = \epsilon_i + X_{\frac{L}{2},i*}^T$.
3. Draw a latent user vector for each user $u$, $U_u \sim \mathcal{N}(0, \lambda_u^{-1}\mathbf{I}_D)$.
4. Draw a rating $r_{ui}$ for each user-item pair $(u, i)$, $r_{ui} \sim \mathcal{N}(U_u^T V_i, C_{ui}^{-1})$

And the cost function is to maximize the joint log-likelihood of U,V,{Xl},Xc,{Wl},{bl}:

$$\begin{aligned}\mathscr{L} = &-\frac{\lambda_u}{2}\sum_i \|\mathbf{u}_i\|_2^2 - \frac{\lambda_w}{2}\sum_l (\|\mathbf{W}_l\|_F^2 + \|\mathbf{b}_l\|_2^2)\\ &-\frac{\lambda_v}{2}\sum_j \|\mathbf{v}_j - \mathbf{X}_{\frac{L}{2},j*}^T\|_2^2 - \frac{\lambda_n}{2}\sum_j \|\mathbf{X}_{L,j*} - \mathbf{X}_{c,j*}\|_2^2\\ &-\frac{\lambda_s}{2}\sum_l\sum_j \|\sigma(\mathbf{X}_{l-1,j*}\mathbf{W}_l + \mathbf{b}_l) - \mathbf{X}_{l,j*}\|_2^2\\ &-\sum_{i,j}\frac{\mathbf{C}_{ij}}{2}(\mathbf{R}_{ij} - \mathbf{u}_i^T\mathbf{v}_j)^2.\end{aligned}$$

Updating rules:

For current W, update u,v first:

$$\begin{aligned}\mathbf{u}_i &\leftarrow (\mathbf{V}\mathbf{C}_i\mathbf{V}^T + \lambda_u\mathbf{I}_K)^{-1}\mathbf{V}\mathbf{C}_i\mathbf{R}_i\\ \mathbf{v}_j &\leftarrow (\mathbf{U}\mathbf{C}_i\mathbf{U}^T + \lambda_v\mathbf{I}_K)^{-1}(\mathbf{U}\mathbf{C}_j\mathbf{R}_j + \lambda_v f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T),\end{aligned}$$

Then set the u,v and update W and b(set their gradients to zero):

$\nabla_{\mathbf{W}_l}\mathscr{L} = 0$, where

$$\begin{aligned}\nabla_{\mathbf{W}_l}\mathscr{L} = &-\lambda_w\mathbf{W}_l\\ &-\lambda_v\sum_j \nabla_{\mathbf{W}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T(f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j)\\ &-\lambda_n\sum_j \nabla_{\mathbf{W}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*})\end{aligned}$$

$$\nabla_{\mathbf{b}_l}\mathscr{L} = -\lambda_w \mathbf{b}_l$$
$$- \lambda_v \sum_j \nabla_{\mathbf{b}_l} f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T (f_e(\mathbf{X}_{0,j*}, \mathbf{W}^+)^T - \mathbf{v}_j)$$
$$- \lambda_n \sum_j \nabla_{\mathbf{b}_l} f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+)(f_r(\mathbf{X}_{0,j*}, \mathbf{W}^+) - \mathbf{X}_{c,j*}).$$

$\nabla_{b_l}\mathscr{L} = 0$, where

**Summary**

Honestly speaking, I did not quite understand the entire frame of CDL. What I can understand is why and how authors want to apply the deep learning models into collaborative filtering recommender algorithms. The problems authors want to solve are cold start problem and rating sparse problem. And authors' general idea is incorporate item information into the CF model. For one thing, CF model use matrix factorized to learn user-item representations. For another, authors use stacked denoising autoencoder to learn distributed representation of items. Therefore, the final goal is to minimize the cost function mixed by cost function of SDAE and cost function of user-item learning.