Probabilistic Matrix Factorization for Music Recommendation

Lu Liu [1]

1. Introduction

Recommender systems are one of the most successful and widespread application of machine learning technologies. Recommender for music is immensely useful as it help us sift through an unprecedented scale of readily available content. Being able to effectively provide personalized music recommendations is essential for the competitiveness of any music listening service.

Collaborative filtering(CF) is an effective way to implement recommender system (Ekstrand et al., 2011). Two of the key challenges in developing CF models are to deal with sparse and imbalanced rating data and to be able to scale well to large dataset (Mnih & Salakhutdinov, 2008). These two challenges are often present in the context of music recommendation (Dror et al., 2011).

Probabilistic Matrix Factorization(PMF) introduced by (Mnih & Salakhutdinov, 2008) has been shown to be a particularly flexible and effective framework to address large, sparse and very imbalanced dataset. Constrained Probabilistic Matrix Factorization (cPMF), a variation on the simple PMF model is also introduced in the same paper, designed to deal with prediction for infreqent users. Subsequent to the introduction of PMF, many PMF-based models were developed. In particular, Kernelized Probabilistic Matrix Factorization(KPMF) model introduced by (Zhou et al., 2012) is able to effectively incorporate side information from user and item to improve recommender's performance.

In this project, we compare and contrast several PMF-based models by applying them to music recommendation. Motivated by the observation that incorporating user network information is not as effective as constraining the user feature vector with latent constraint similarity matrix, we developed Constrained Kernelized PMF (cKPMF) model. We show that cKPMF is the most effective model for our task at hand among the models explored in this project.

The main input to all our models are the logarithm of the listening count(Figure 1(b)), which we will use as pseudo-ratings. Our models output predicted ratings for any [user - artist] pair we query.

[1]SCPD. Correspondence to: Lu Liu <luliu7@stanford.edu>.

For the Kernelized PMF models, we used 2 additional inputs: user network(Figure 1(a)), and artist tag assignments (Figure 1(c)).

2. Related Work

(Fang et al., 2014) extend PMF by decomposing trust information into four general trust aspects, i.e. benevolence, integrity, competence, and predictability, and incorporate them into the PMF model with support vector regression. (Salakhutdinov & Mnih, 2008) propose a Bayesian PMF , which generalizes PMF to handle nonzeromean and non-spherical Gaussian priors. The advantage of BPMF is that it is less prone to overfitting, however it suffers from high computing complexity. In the context of music recommendation where model scalability is an important issue, it is often not the most ideal choice.

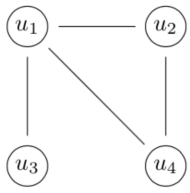
Several works explored augmenting MF models with kernel methods to alleviate the problem of cold-start users or items. Besides the work of (Zhou et al., 2012) whose KPMF model we will use, (Liu et al., 2016) incorporates kernel methods for matrix factorization, which embeds the low-rank feature matrices into a much higher dimensional space, enabling the ability to learn nonlinear correlations upon the rating data in original space. (Gönen et al., 2013) form a Bayesian MF that models the priors of latent factors through a linear combination of multiple kernels on multiple side information.

3. Dataset

We are using hetrec2011-lastfm-2k (Cantador et al., 2011), a set of social networking, tagging, and music artist listening information from Last.fm[1] online music system.

To speed up our experiments, we used a subset with the top 1000 most frequently rated artist. The statistics of the dataset is given in Table [1]. Observe that the rating density is very low. Figure [2] shows the histogram for the number of ratings of each artist and each user. We observe that the artist rating frequencies are more balanced, since most artists have similar rating frequencies. But user rating frequencies varies more significantly. So this dataset suffers from the typical sparsity and data imbalance problem.

[1]www.last.fm



(a) User Network

$$\begin{array}{c} \\ u_1 \\ u_2 \\ u_3 \\ u_4 \end{array} \begin{array}{c} \begin{array}{ccc} i_1 & i_2 & i_3 \end{array} \\ \begin{pmatrix} 2.3 & 9.5 & 0.3 \\ 1.3 & ? & 6.7 \\ ? & 1.8 & ? \\ ? & 2.5 & ? \end{pmatrix} \end{array}$$

(b) Rating Matrix

$$\begin{array}{c} \\ i_1 \\ i_2 \\ i_3 \end{array} \begin{array}{c} \begin{array}{cccc} tag_1 & tag_2 & tag_3 & tag_4 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

(c) Artist Tag Assignments

Figure 1. Example input data:(a) social network among 4 users; (b) observed rating matrix for the 4 users on 3 artists; (c) tags for the 3 artists.

ITEM	STATS
# USERS	1871
# ITEMS	1000
# RATINGS	56620
RATING DENSITY	3.03%
# RELATIONS	25424
# TAGS	87366

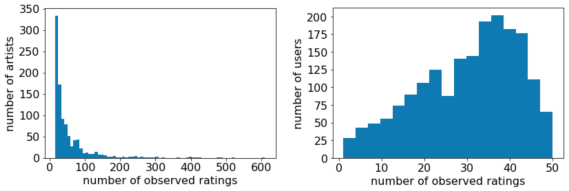
Table 1. Statistics of the dataset used.

We observe that the distribution of the 56620 ratings are highly skewed, only 1300 ratings are larger than 5000. We therefore choose to use log(listening count) as our rating. Figure [3] shows the rating distribution before and after the log transformation. Notice that rating distribution is approximately log-normal.

4. Methods

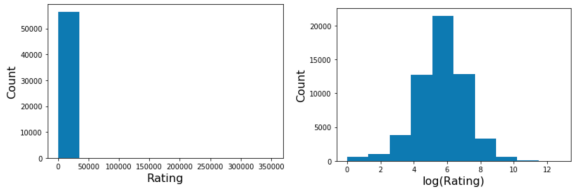
4.1. Notation

We define the main notations used in this paper as follows:
N - Number of users
M - Number of items
$R \in \mathcal{R}^{N \times M}$ Rating matrix
D - latent dimension
$U \in \mathcal{R}^{N \times D}$ - user latent matrix
$V \in \mathcal{R}^{M \times D}$ - items latent matrix
$K_u \in \mathcal{R}^{N \times N}$ - Covariance matrix for rows of R
$K_v \in \mathcal{R}^{M \times M}$ - Covariance matrix for columns of R
$S_u \in \mathcal{R}^{N \times N}$ - inverse of K_u
$S_v \in \mathcal{R}^{M \times M}$ - inverse of K_v
$I \in \mathcal{R}^{N \times M}$- indicator matrix taking value 1 if $R_{i,j}$ is an observed entry, and 0 otherwise.
α - learning rate



(a) Artist rating frequency distribution (b) User rating frequency distribution

Figure 2. Artist and User rating frequency distributions



(a) Rating distribution (b) Log-rating distribution

Figure 3. Rating and Log-rating distribution of dataset

4.2. Baseline: PMF

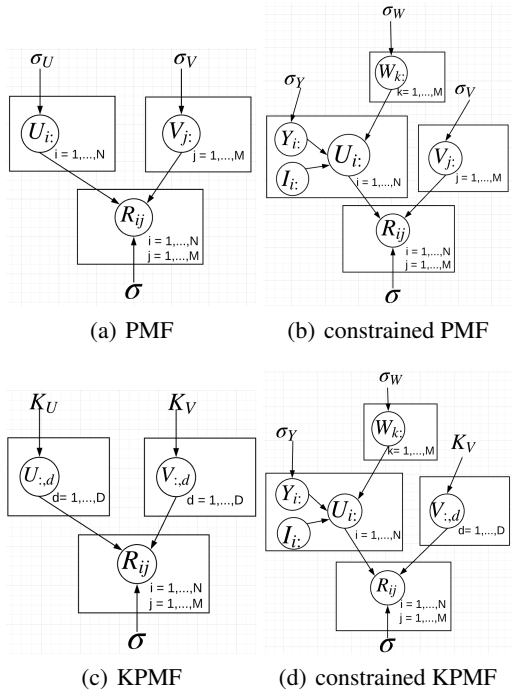
Probabilistic Matrix Factorization is a generative algorithm introduced by (Mnih & Salakhutdinov, 2008). It assumes the following generative process for the rating matrix R(see figure [4(a)]:
1. Generate $U_{i,:} \sim \mathcal{N}(0, \sigma_U^2 I)$ for $i \in \{1, ..., N\}$
2. Generate $V_{j,:} \sim \mathcal{N}(0, \sigma_V^2 I)$ for $j \in \{1, ..., M\}$
3. For each non-missing entry $R_{i,j}$, generate $R_{i,j} \sim \mathcal{N}(U_{i,:} V_{j,:}^T, \sigma^2 I)$
In this model, we are making the following independence assumptions: between U and V, between the rows of U and between the rows of V, between each non-missing entry $R_{i,j}$.

Note that in (Mnih & Salakhutdinov, 2008), in order to limit the predicted ratings to be within the valid range, they pass the dot product between user - item feature vector through the logistic function. However, in our context, since our rating is the logarithm of listening count, whose range is $[-\infty, \infty]$, we use the dot product directly.

The log-posterior over the latent matrices U and V is given by:

$$\begin{aligned} &\log p(U, V | R, \sigma^2, \sigma_U^2, \sigma_V^2) \\ &= \log p(R | U, V, \sigma^2) + \log p(U | \sigma_U^2) + \log p(V | \sigma_V^2) + C \end{aligned} \tag{1}$$

Where C is a constant that does not depend on U and V.
We perform MAP estimate to learn U and V which maximizes (1) by Stochastic Gradient Descent. For each non-missing $R_{i,j}$ the update rule is as follows:



(a) PMF (b) constrained PMF

(c) KPMF (d) constrained KPMF

Figure 4. The Generative process of R in different models

$$err_{i,j} = R_{i,j} - U_{i,:}V_{j,:}^T$$
$$U_{i,:} := U_{i,:} + \alpha(err_{i,j}V_{j,:} - \frac{\sigma^2}{\sigma_U^2 \sum_{p=1}^{M} I_{i,p}}U_{i,:}) \quad (2)$$
$$V_{j,:} := V_{j,:} + \alpha(err_{i,j}U_{i,:} - \frac{\sigma^2}{\sigma_V^2 \sum_{p=1}^{N} I_{p,j}}V_{j,:})$$

4.3. KPMF

Recall that in the PMF models, rows of U and V are assumed to be independent, and we are only using the rating matrix as input.

The Kernelized Probabilistic Matrix Factorization model as introduced in (Zhou et al., 2012) allows U and V to capture the covariances between any 2 rows of U and V by assuming the columns of U and V are generated from a zero-mean Gaussian Process(GP). By generating the covariance matrices from user and item side information, we can easily incorporate them into the model.

The generative process for KPMF is as follows (see figure 4(c)):

1. Generate $U_{:,d} \sim GP(0, K_u)$ for $d \in \{1, ..., D\}$
2. Generate $V_{:,d} \sim GP(0, K_v)$ for $d \in \{1, ..., D\}$
3. For each non-missing entry $R_{i,j}$, generate $R_{i,j} \sim \mathcal{N}(U_{i,:}V_{j,:}^T, \sigma^2)$

Note that when the covariance matrices K_u and K_v are both diagonal, KPMF reduces to PMF.

We are still assuming the independence between U and V and independence between the observed ratings.

Due to page limit, we refer reader to the original paper (Zhou et al., 2012) for the details of the model and Stochastic Gradient Descent update rules.

4.4. cPMF

For the simple PMF model, the latent vector for an infrequent user i will be close to the user average in the trained model. The predicted ratings for User i will be close to the global average ratings. This is a very crude way to make predictions for infrequent users.

A better way is to find other users who listened to the same(or similar) artists as user i, and constrain the latent vectors of the similar users to be similar. the Constrained PMF is a variation of PMF introduced by (Mnih & Salakhutdinov, 2008) that utilizes this intuition. It constrains U with a latent similarity constraint matrix $W \in \mathcal{R}^{M \times D}$ as follows:

$$U_{i,:} = Y_{i,:} + \frac{\sum_{k=1}^{M} I_{i,k}W_{k,:}}{\sum_{k=1}^{M} I_{i,k}} \quad (3)$$

Where the k-th row of W captures the effect that rating artist k has on the user's feature vector. Where $Y \in \mathcal{R}^{N \times D}$ captures the user offset.

See figure 4(b) for the generative process of cPMF.

The Stochastic Gradient Descent rules for finding MAP estimate of Y, V, W is as follows:

for each observed rating $R_{i,j}$

$$U_{i,:} = Y_{i,:} + \frac{\sum_{k=1}^{M} I_{i,k}W_{k,:}}{\sum_{k=1}^{M} I_{i,k}}$$
$$err_{i,j} = R_{i,j} - U_{i,:}V_{j,:}^T$$
$$Y_{i,:} := Y_{i,:} + \alpha(err_{i,j}V_{j,:} - \frac{\sigma^2}{\sigma_Y^2 \sum_{p=1}^{M} I_{i,p}}Y_{i,:})$$
$$V_{j,:} := V_{j,:} + \alpha(err_{i,j}U_{i,:} - \frac{\sigma^2}{\sigma_V^2 \sum_{p=1}^{N} I_{p,j}}V_{j,:})$$
$$W := W + \alpha err_{i,j}\frac{I_{i,:} \otimes V_{j,:}}{\sum_{k=1}^{M} I_{i,k}}$$
$$W_{j,:} := W_{j,:} - \alpha\frac{\sigma^2}{\sigma_W^2 \sum_{p=1}^{N} I_{p,j}}W_{j,:} \quad (4)$$

4.5. cKPMF

(Zhou et al., 2012) showed that incorporating user network information with KPMF is very effective at improving prediction accuracy. However, in our context, the improvements were minimal. Meanwhile, constraining the user latent matrix with cPMF as well as incorporating item side information with KPMF was very effective.

Motivated by these observation, we developed the Constrained Kernelized Probabilistic Matrix Factoriza-

tion(cKPMF) model, where we constrain the user latent matrix as well as assuming Gaussian process distribution for the columns of item latent matrix.
The generative process for cKPMF is as follows (see figure 4(d)):
1. Generate $W_{k,:} \sim \mathcal{N}(0, \sigma_W^2 I)$ for $k \in \{1, ..., M\}$
2. Generate $Y_{i,:} \sim \mathcal{N}(0, \sigma_Y^2 I)$ for $i \in \{1, ..., N\}$
3. Generate $V_{:,d} \sim GP(0, K_v)$ for $d \in \{1, ..., D\}$
4. Generate indicator matrix I such that $I_{i,j} = 1$ if $R_{i,j}$ is observed,$I_{i,j} = 0$ otherwise.
5. For each non-missing entry $R_{i,j}$, generate $R_{i,j} \sim \mathcal{N}((Y_{i,:} + \frac{\sum_{k=1}^{M} I_{i,k} W_{k,:}}{\sum_{k=1}^{M} I_{i,k}}) V_{j,:}^T, \sigma^2)$
Notice that when K_v is a diagonal, cKPMF reduce to cPMF. The Stochastic Gradient Descent rules for Y and W are the same with the cPMF model, the update rule for V is as follows:

$$V_{j,:} := V_{j,:} + \alpha(err_{i,j} U_{i,:} - \frac{\sigma^2}{2\sum_{p=1}^{N} I_{p,j}} (\sum_{k=1}^{M} S_{vj,k} V_{k,:} + S_{vj,j} V_{j,:})) \quad (5)$$

4.6. Construct Covariance Matrices for GP

A valid kernel function for GP should generate a covariance matrix that is positive semi-definite. There are many available choices(Hofmann et al., 2008).

4.6.1. Construct K_v

To incorporate the artist side information into the covariance matrix, we first constructed a artist tag matrix using one-hot encoding (figure 1(c)). We then applied the Radial Basis Function(RBF) kernel to the feature vectors to obtain K_v. RBF kernel effectively calculates a similarity measure between any 2 artists' feature vectors.

4.6.2. Construct K_u

Following (Zhou et al., 2012), we considered the users' social network as an undirected, unweighted graph G with nodes and edges representing users and their connections (1(a)).
We compared the 3 graph kernels described in (Zhou et al., 2012): Diffusion, Commute Time(CT), Regularized Laplacian. In addition, we generated a node2vec (Grover & Leskovec, 2016) kernel matrix by first generating node embeddings with node2vec then convert the embedding matrix into a kernel matrix with RBF kernel.
Our experiment results show CT kernel outperforms the rest.

5. Experiments

5.1. Experimental Setting

For all our experiments, we used Root Mean Square Error (RMSE) for performance evaluation:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (r_i - \hat{r}_i)^2}{n}} \quad (6)$$

where r_i denotes the ground-truth rating value, $\hat{r}_i$ denotes its predicted value, and n is the total number of ratings to be predicted.
For hyperparameters, we fixed maximum epoch at 100, latent dimension at 30. For each (model, training size) pair, we perform grid search to find the learning rate and regularization coefficient that performed best on validation data.

We then retrain the models with the selected hyperparameters until validation RMSE starts increasing or maximum epoch has been reached. We then use this "best model" to obtain RMSE on the test set.

The general framework of our code was based off (Bolmier, 2019). We also utilized the RBF_kernel and mean_squared_error functions from scikit-learn (Pedregosa et al., 2011) in our code.

5.2. Results

Figure [5] and Table [2] shows the RMSE on test set for different model using 80% and 20% of the ratings. The main observations are as follows:

1. All models achieve lower RMSE score using more ratings for training. The smallest RMSE score with 20% data (1.224) is still higher than the largest RMSE score with 80% data (1.139). This shows that if possible, having more training data is more important than picking the best model.

2. The effect of constraining the user latent matrix in the cPMF model is significant. Even without utilizing any side information, it achieves comparable performance with the kernelized models that do utilize extra side information. When the training data is extremely sparse, it provides over 40% reduction in RMSE compared to the PMF baseline model. Since a lot of the times, side information are not readily available, cPMF can be extremely useful in those settings.

3. Exploiting user and artist side information are both effective at reducing RMSE. However, the effect of including artist tag assignment is much more significant than user network. The ineffectiveness of exploiting user network could be due to the fact that user interactions on last.fm website is not a major feature that's actively used by its users. So the network data may be

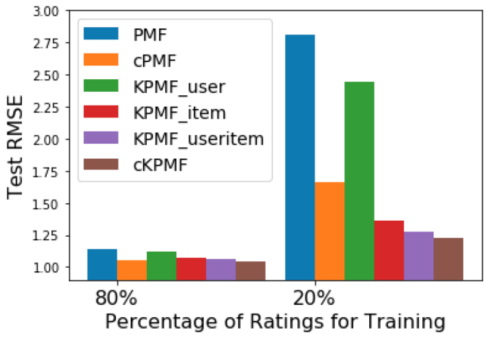


Figure 5. Test RMSE for different models

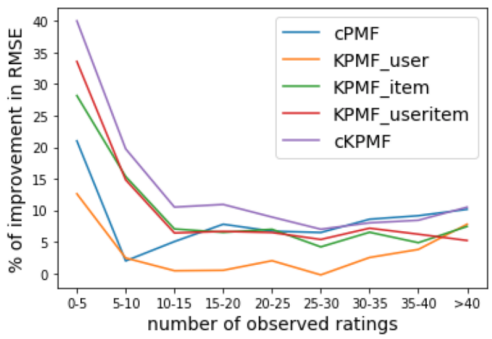


Figure 6. Percentage of improvement in RMSE over PMF by user group

MODEL	80%TRAIN	20%TRAIN
PMF	1.139	2.808
CPMF	1.056	1.662
KPMF_USER	1.119	2.444
KPMF_ITEM	1.070	1.361
KPMF_USERITEM	1.064	1.273
CKPMF	**1.039**	**1.224**

Table 2. RMSE comparision on test set. Smaller is better.

very noisy.

4. Our novel cKPMF model outperforms all other models in both training data settings, with improvement in RMSE more significant when the training data is sparse.

To observe the models' performances for infrequent users, we grouped the users by their number of observed ratings. We then plot the percentage of improvement in RMSE over the baseline PMF for the various user groups when trained under 80% ratings (Figure [6]). The main observations are as follows:

1. All models achieves higher percentage of improvements for users with very few observed ratings (0 to 5). This is very promising, as we developed these model variations on PMF specifically to address the situations when ratings data are sparse. cPMF's ability to generalize for users with few ratings is especially impressive since it is not utilizing any side information at all.
2. Our cKPMF model outperforms all other models in nearly all user groups (only slightly exceeded by cPMF for users with large enough number of ratings).
3. Comparing performance of cPMF and cKPMF, we see that, when the users rating data is lacking, cKPMF is able to leverage the artist tag information to achieve a boost in performance. But when the rating data is abundant, cPMF and cKPMF achieve similar RMSE.
4. Comparing performance of KPMF_item and KPMF_useritem, we once again observe that, the effect of incorporating user network information is minimal. But there is still an observable effect for users with nearly no ratings at all.

6. Conclusion

In this project, we applied several existing variations of PMF models to music recommendation, and we proposed cKPMF model by combining the techniques in these variations. We explored different kernel functions to best incorporate side information into the KPMF models. Our experiments show that, cPMF model, by constraining the user latent matrix with a latent similarity matrix, is extremely effective at enhancing model performance when side information is unavailable, even when training data is extremely sparse. For the KPMF models, we see that leveraging artist tag assignment is much more useful than leveraging user network information. We also observed that our novel cKPMF model is superior to all other models compared in this project under both training size settings and among all user groups. For future work, we want to explore the effect of adding user and artist bias into all these models. We want to add memory based models that uses only the side information as baselines. We would also like to experiment with the computational efficiency and convergence behaviour of these models in different settings.

7. Code

https://drive.google.com/open?id=199isUAS3stnQjOzEVgw-Tatht_TlhddO

References

Bolmier, G. funk-svd, May 2019. URL https://github.com/gbolmier/funk-svd.

Cantador, I., Brusilovsky, P., and Kuflik, T. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *Proceedings of the 5th ACM conference on Recommender systems*, RecSys 2011, New York, NY, USA, 2011. ACM.

Dror, G., Koenigstein, N., Koren, Y., and Weimer, M. The yahoo! music dataset and kdd-cup'11. In *Proceedings of the 2011 International Conference on KDD Cup 2011-Volume 18*, pp. 3–18. JMLR. org, 2011.

Ekstrand, M. D., Riedl, J. T., Konstan, J. A., et al. Collaborative filtering recommender systems. *Foundations and Trends® in Human–Computer Interaction*, 4(2):81–173, 2011.

Fang, H., Bao, Y., and Zhang, J. Leveraging decomposed trust in probabilistic matrix factorization for effective recommendation. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

Gönen, M., Khan, S., and Kaski, S. Kernelized bayesian matrix factorization. In *International Conference on Machine Learning*, pp. 864–872, 2013.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, 2016.

Hofmann, T., Schölkopf, B., and Smola, A. J. Kernel methods in machine learning. *The annals of statistics*, pp. 1171–1220, 2008.

Liu, X., Aggarwal, C., Li, Y.-F., Kong, X., Sun, X., and Sathe, S. Kernelized matrix factorization for collaborative filtering. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pp. 378–386. SIAM, 2016.

Mnih, A. and Salakhutdinov, R. R. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pp. 1257–1264, 2008.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

Salakhutdinov, R. and Mnih, A. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pp. 880–887. ACM, 2008.

Zhou, T., Shan, H., Banerjee, A., and Sapiro, G. Kernelized probabilistic matrix factorization: Exploiting graphs and side information. In *Proceedings of the 2012 SIAM international Conference on Data mining*, pp. 403–414. SIAM, 2012.