Movie Recommendation System using MCMC Methods
STATS551 - Bayesian statistics - Term Project Report

Xiaoyang Sheng (shengxy@umich.edu)
Yulin Gao (yulingao@umich.edu)
Zicong Xiao (zicongx@umich.edu)

Dec 2023

1 Introduction

Recommendation services have played an important part in our daily lives. After we just finish watching a video on YouTube, for instance, a new selection of videos promptly appears on our screen, tailored to our individual interests. This personalized recommendation list is typically generated according to our previous rating history or preferences via statistical inference methods. The objective of this project is to forecast the user's rating for a movie they have not yet watched, before recommendation, employing Bayesian inference, with a particular emphasis on Markov Chain Monte Carlo (MCMC) methods.

2 Data

The MovieLens 20M Dataset, available via Kaggle (www.kaggle.com/datasets/grouplens/movielens-20m-dataset) [1], encompasses a vast repository of 20,000,263 movie rating activities, across 27,278 movies, created by 138,493 customers, since 1995. The customers are selected at random for inclusion.

Use_id	Movie_id	Rating	Timestamp
1	5999	3.5	2005-04-02 23:55:50
...	...	...	...

Table 1: *Rating* table

Rating table documents the rating activities for movies by customers. Each customer or movie within the dataset can be identified by a unique User_id, or Movie_id. The Rating column takes value in the range from 0.5 to 5. The timestamp marks the point when the customer updates the rating.

Movie_id	Title	Genres
1	Toy Story (1995)	Adventure, Animation, Children, Comedy, Fantasy
...	...	...

Table 2: *Movie* table

Movie table shares in detail the information of each movie, including its title, publication year, and multiple genre classifications. Number of movies in different genres is displayed below in figure 1.

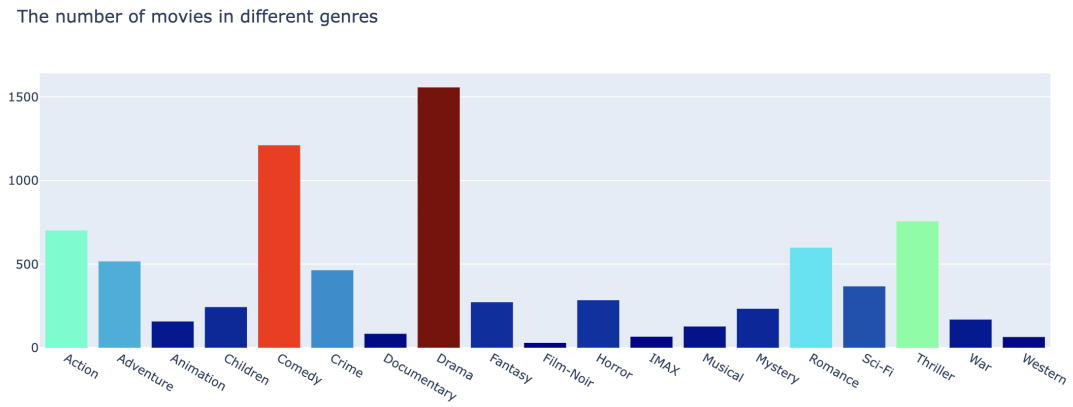


Figure 1: Number of movies in different genres

Notably, the average rating of movies for different genres is different, which is shown in figure 2.

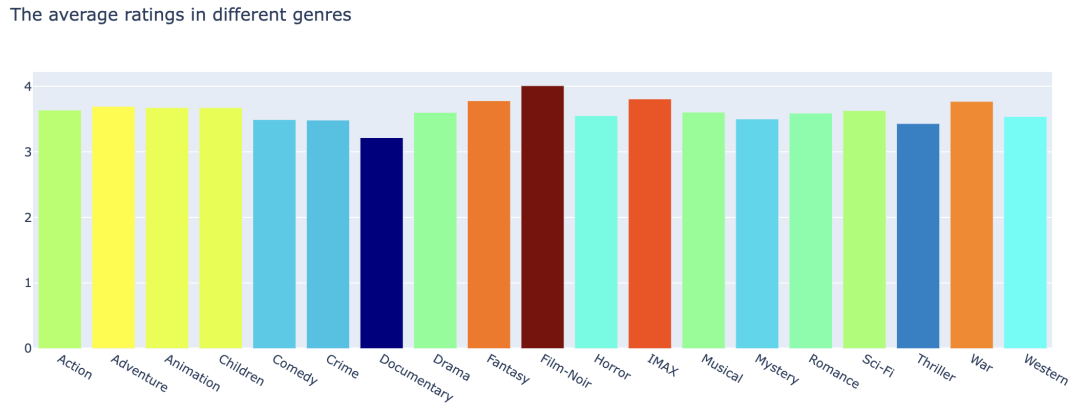


Figure 2: Average rating of movies in different genres

User_id	Action	Adventure	Animation	...	Western
1	0	1	2	...	0
...	...	...	...	...	...

Table 3: *User* table

Moreover, we have applied one-hot encoding to the movie genres and integrated the encoded columns with the rating table. User table, as a result, counts the number of movies watched in each genre for each individual user. User_id serves as the primary key. This table will play an essential part in the user clustering in the subsequent methods section.

A potential limitation in the dataset is the sparse nature of customer ratings, where each customer rates only a limited number of movies, and each movie receive ratings from a relatively small pool of customers, compared with the whole population. We would do data cleaning work by removing inactive customers or outdated movie information.

3 Research Question

The central challenge is predicting a customer's rating, $R \in [0.5, 5]$, for a newly introduced, yet unrated, movie. The prediction relies on a comprehensive history of customer-movie-rating tuples at hand. Our ultimate goal is to construct a system capable of predicting the rating score for any customer-movie pair within our dataset and subsequently recommending the movie to the user if

the prediction falls above a predefined threshold, like $R \geq 3.5$. The report will primarily focus on enhancing the precision of predictions.

4 Methodology

4.1 Traditional Bayesian Probabilistic Matrix Factorization (BPMF)

Having pivoted the user-movie-rating table, we structure it into a matrix R with dimensions $M \times N$, where M represents the number of users and N counts number of movies in original set. Bayesian Probabilistic Matrix Factorization (BPMF) is introduced as a solution to the research question. BPMF focuses on decomposing the single matrix into two distinct matrices, denoted as U and V, dedicated to representing users and movies, respectively [2].

$$R = U^T V \tag{1}$$

where $\dim(U) = D \times M$ and $\dim(V) = D \times N$.

Each column of U, U_i, is a D-dimensional vector containing D latent features of the user i, and the same holds true for V_j of movie j. Thus, the rating R_{ij} assigned by the i^{th} user to the j^{th} movie can be expressed as,

$$E[R_{ij}] = U_i^T V_j = \sum_{d=1}^{D} U_{id} * V_{jd} \tag{2}$$

where D is a hyper-parameter that needs to be fine-tuned in a subsequent stage. To achieve accurate predictions for ratings, the task can be formulated as training and validating a model on non-blank cells in matrix R, followed by making predictions for values in other originally blank cells.

The model training process can be realized by Gibbs sampling, a Markov Chain Monte Carlo (MCMC) algorithm used for statistical inference. The basic idea behind Gibbs sampling is to iteratively sample from the conditional distributions of each variable given the values of all other variables in the system. To initiate the process, certain assumptions are established,

$$U_i \sim \mathrm{N}(\mu_U, \Sigma_U) \tag{3}$$

$$V_i \sim \mathrm{N}(\mu_V, \Sigma_V) \tag{4}$$

$$R_{ij} \sim \mathrm{N}(U_i^T V_j, \sigma^2) \tag{5}$$

and, in turn, multivariate normal priors as well as inverse Wishart priors are specified for μ, Σ,

$$\mu_{U,V} \sim \mathrm{N}(\mu_0, T_0) \tag{6}$$

$$\Sigma_{U,V} \sim \text{inv-Wishart}(\nu_0, S_0^{-1}) \tag{7}$$

where μ_0, T_0, ν_0 and S_0 are hyper-parameters.

4.2 Updated Bayesian Probabilistic Matrix Factorization (BPMF)

Before delving into the Gibbs process, we make one additional update to the structure of BPMF.

$$R = U^T B V, \quad E[R_{ij}] = U_i^T B V_j = U_i^T V_j^* \tag{8}$$

where B is a diagonal matrix of shape $D \times D$. The main diagonal of B is denoted as β,

$$\beta = \text{diag}(B), B = \begin{bmatrix} b_{11} & 0 & 0 \\ 0 & \dots & 0 \\ 0 & 0 & b_{DD} \end{bmatrix} \rightarrow \beta = \begin{bmatrix} b_{11} \\ \dots \\ b_{DD} \end{bmatrix} \tag{9}$$

We introduce an additional multivariate normal assumption for β priors,

$$\beta \sim \text{N}(\mu_\beta, \Sigma_\beta), \quad \mu_\beta \sim \text{N}(\mu_0, T_0), \quad \Sigma_\beta \sim \text{inv-Wishart}(\nu_0, S_0^{-1}) \tag{10}$$

The derivation of the full conditional distribution unfolds through a successive process,

$$P\left(\mu_U \mid U, \Sigma_U\right) \sim \text{dnorm}\left(\left(T_0^{-1} - M\Sigma_U^{-1}\right)^{-1}\left(T_0^{-1}\mu_0 + M\Sigma_U^{-1}\bar{U}\right), T_0^{-1} + M\Sigma_U^{-1}\right) \tag{11}$$

$$P\left(\Sigma_U \mid U, \mu_U\right) \sim \text{dinv-Wis}\left(\nu_0 + M, \left[S_0 + \sum (U_i - \mu_U)(U_i - \mu_U)^T\right]^{-1}\right) \tag{12}$$

$$P(U|R, B, V, \mu_U, \Sigma_U) \propto \prod_i^M P(U_i|\mu_U, \Sigma_U) \prod_i^M \prod_j^N [P(R_{ij}|U_i^T B V_j, \sigma^2)]^{1\{R_{ij}>0\}} \tag{13}$$

where $1\{R_{ij} > 0\}$ signifies that the training and validation processes only consider the non-blank cells.

$$P\left(U_i \mid \dots\right) \propto P\left(U_i|\mu_U, \Sigma_U\right) \prod_j^N [P(R_{ij}|U_i^T B V_j, \sigma^2)]^{1\{R_{ij}>0\}} \tag{14}$$

$$\propto \exp\left\{-\frac{1}{2}(U_i - \mu_U)^T \Sigma_U^{-1}(U_i - \mu_U)\right\} \exp\left\{-\frac{1}{2\sigma^2}\sum_j^N 1\{R_{ij} > 0\}\left[U_i^T V_j^* V_j^{*T} U_i - 2R_{ij}U_i^T V_j^*\right]\right\} \tag{15}$$

Hence, $P\left(U_i \mid \dots\right)$ can be simplified as $\text{dnorm}(\text{U}_\text{i}, \mu_\text{i}^*, \Sigma_\text{i}^*)$, where

$$\Sigma_i^* = \Sigma_U^{-1} + \frac{1}{\sigma^2}\sum_j^N 1\{R_{ij} > 0\} V_j^* V_j^{*T} \tag{16}$$

$$\mu_i^* = \Sigma_i^*\left(\frac{1}{\sigma^2}\sum_j^N 1\{R_{ij} > 0\}(V_j^* R_{ij}) + \Sigma_U^{-1}\mu_U\right) \tag{17}$$

The scenario is analogous for the posterior of β, as it can also be written like,

$$P(\beta|R, U, V, \mu_\beta, \Sigma_\beta) \propto P(\beta|\mu_\beta, \Sigma_\beta) \prod_i^M \prod_j^N [P(R_{ij}|U_i^T B V_j, \sigma^2)]^{1\{R_{ij}>0\}} \tag{18}$$

$$\propto \exp\left\{-\frac{1}{2}(\beta - \mu_\beta)^T \Sigma_\beta^{-1}(\beta - \mu_\beta)\right\} \exp\left\{-\frac{1}{2\sigma^2}\sum_i \sum_j 1\{R_{ij} > 0\}\left[\beta^T A A^T \beta - 2R_{ij}\beta^T A\right]\right\} \tag{19}$$

where $A = U_j \circ V_j$ Hadamard product of latent vectors U_i and V_j.
Hence, $P(\beta \mid \dots)$ can be simplified as $\text{dnorm}(\beta, \mu^*_\beta, \Sigma^*_\beta)$, where

$$\Sigma^*_\beta = \Sigma^{-1}_\beta + \frac{1}{\sigma^2} \sum_j^N 1\{R_{ij} > 0\} AA^T \tag{20}$$

$$\mu^*_\beta = \Sigma^*_\beta \left(\frac{1}{\sigma^2} \sum_j^N 1\{R_{ij} > 0\}(AR_{ij}) + \Sigma^{-1}_\beta \mu_\beta \right) \tag{21}$$

The Gibbs sampling procedure then operates as follows:

Initialize $U, V, \sigma^2, \mu_0, T_0, \nu_0, S_0$
For it in $1, ..., It$:
    Sample μ_U, μ_V, μ_β
    Sample $\Sigma_U, \Sigma_V, \Sigma_\beta$
    For i in $1, ..., M$:
        Sample U_i
    For i in $1, ..., N$:
        Sample V_j
    Sample β
    Compute $R^* = U^T BV$ and append to Results.

We initiate more quantitative analysis by partitioning the non-blank cells into a training set and a test set, with a split ratio of 0.8 for training and 0.2 for testing. These sets will be employed in the evaluation of the traditional BPMF model. The evaluation aims to investigate the runtime and the approximate number of iterations required for convergence, considering a fixed configuration with $D = 10$ and $It = 100$.

Furthermore, we will turn to the updated BPMF model and employ the same training and testing sets. This step will serve as a comparative analysis of the performance between the updated BPMF model and the traditional one, assessing metrics such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE).

We will additionally conduct the experiments to explore the effect from varying the number of latent features, i.e., the hyper-parameter D. Through a series of trials involving different values of $D = 2, 5, 10, 20, 50, 100$, we will investigate how this hyper-parameter influences the overall prediction performance of the model.

4.3 BPMF with Gaussian Mixture Model Clustering

While we were training the BPMF model, we found that the size and time consumed is quite a big challenge. If the size of the data (users or movies) get larger, the size of the matrix would also increase, and though the matrix could be sparse, but the memory it cost while creating the matrix is still a challenge and programming language could set certain limitation as well. Meanwhile, when the size gets larger, the time consumed to train the model increase significantly, as the Gibbs sampler would cost more time.

Therefore, we decide to come up with a method that divide the user group and then the task and huge matrix could be divided into multiple parts. Moreover, it is reasonable to customize the recommendation system by different groups of users, which could make the recommendation better fit people's needs.

In this section, we apply the Gaussian Mixture Model to accomplish the unsupervised clustering.

4.3.1 Gaussian Mixture Model (GMM)

The Gaussian Mixture Model could be considered as a Bayesian model with the normal prior distribution, and further determines all the parameters including cluster mean, variance, labels through many kinds of methods like Expectation-Maximization (EM) algorithm and Bayesian non-parametrics (BNP).

Here we display a basic algorithm structure for Gaussian Mixture Model like the one we used in the assignments of this course.

The overall Mixture Model:

Each data point X_i is either drawn from $N(\mu_k, \Sigma_k)$, depending on the value of a latent variable Z_i. $Z_i = k$ indicates $X_i \sim N(\mu_k, \Sigma_k)$. The probability of $Z_i = k$ is p_k for $k = 1, 2, 3, ..., m$.

Density Function:

The density function of each X_i is a mixture of the m Gaussian densities:

$$p(x) = p_1 N(x|\mu_1, \Sigma_1) + p_2 N(x|\mu_2, \Sigma_2) + ... + p_m N(x|\mu_m, \Sigma_m)$$

Prior Distributions:

1. Mixture Probabilities $p = (p_1, p_2, ..., p_m)$:

   Dirichlet prior: $p \sim \text{Dir}(\alpha_1, \alpha_2, ..., \alpha_m)$.

2. Mean Vectors μ_k for $k = 1, 2, 3, ..., m$:

   Normal prior: $\mu_k \sim N(\mu_0, \Sigma_0)$. μ_0 is the overall empirical mean of the data, and Σ_0, is the empirical covariance matrix.

3. Covariance Matrices Σ_k for $k = 1, 2, 3, ..., m$:

   Inverse-Wishart prior: $\Sigma_k \sim \text{InverseWishart}(\nu_0, S_0^{-1})$. $\nu_0 = d + 1$, larger than the number of dimensions of the data. S_0 is the sample covariance matrix of the data, making the prior centered around the empirical covariance of the data.

Posterior Full Conditional Distributions:

1. Label:

   $P(Z_i = k|x_i, p, \mu, \Sigma) = \frac{p_k \times N(x_i;\mu_k,\Sigma_k)}{\sum_{j=1}^{m} p_j \times N(x_i;\mu_j,\Sigma_j)}$

2. Mixture Probabilities:

   $p|x, z, \mu, \Sigma \sim Dir(\alpha_1 + n_1, \alpha_2 + n_2, ..., \alpha_m + n_m)$, where $n_1, n_2, ..., n_m$ is the number of samples in each cluster.

3. Mean Vectors:

   $\mu_k|x, z, p, \Sigma \sim \mathcal{N}(\mu_k; A_{n_k}^{-1} b_{n_k}, A_{n_k}^{-1})$ where $A_{n_k} = \Sigma_0^{-1} + n_k \Sigma_k^{-1}$ and $b_{n_k} = \Sigma_0^{-1}\mu_0 + \Sigma_k^{-1} \sum_{i:Z_i=k} x_i$

4. Covaraince Matrices:

   $\Sigma_k|x, z, p, \mu, \sim InverseWishart(v_0 + n_k, [\sum_{i:Z_i=k}(x_i - \mu_k)(x_i - \mu_k)^T + S_0]^{-1})$.

By GMM model structure like above, we could run it to cluster the user data.

4.3.2 User Clustering

We apply the GMM model on the user data, which we aggregate all the categories of movies each user has rated, and use the one-hot encoding to get the final user matrix. Note that each movie could be classified as multiple categories and we counter them all in the user matrix.

We use the Python scikit-learn GMM model package to implement our clustering tasks. We also use the hyper-parameter tuning with grid cross validation by BIC scores.

Once we have clustered the users into distinct groups, we will proceed to execute the Updated BPMF algorithm on each cluster individually. This approach allows us to obtain the final results for each cluster, facilitating a more comprehensive comparative analysis.

5 Result

5.1 Traditional BPMF

We have extracted a sample set comprising 10,000 user-movie-rating tuples from the original MovieLens dataset. The sampled set includes rating records from 8,959 unique users spanning 3,368 distinct movies. The training set consists of 80 percent of the sampled records, 8,000 instances in total, while the remaining 20 percent serve as the test set. We have fixed the initial values and hyper-parameters, ensuring a consistent configuration,

$$
\begin{aligned}
&D = 10, \quad It = 100 \\
&U = \mathbf{1}^{D\times M}, V = \mathbf{1}^{D\times N} \\
&\mu_0 = \mathbf{0}^{D\times 1}, \quad \nu_0 = D \\
&T_0 = \mathrm{diag}(D), \quad S_0 = \mathrm{diag}(D)
\end{aligned}
$$

The first trail is on **MAC - Apple M1 Pro - 16 GB Mem**, taking 1417 seconds for 100 loops.

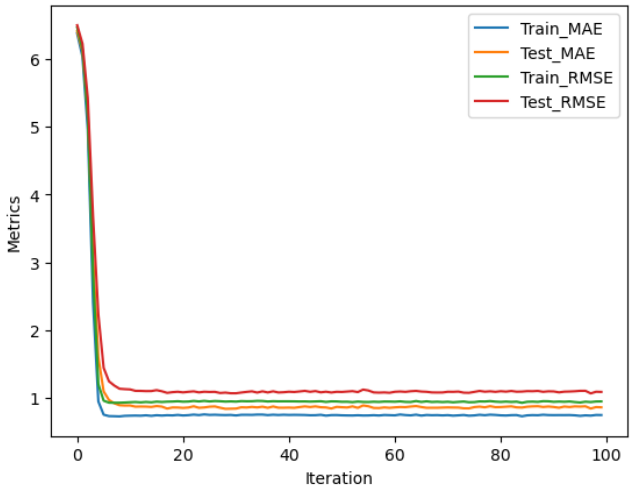
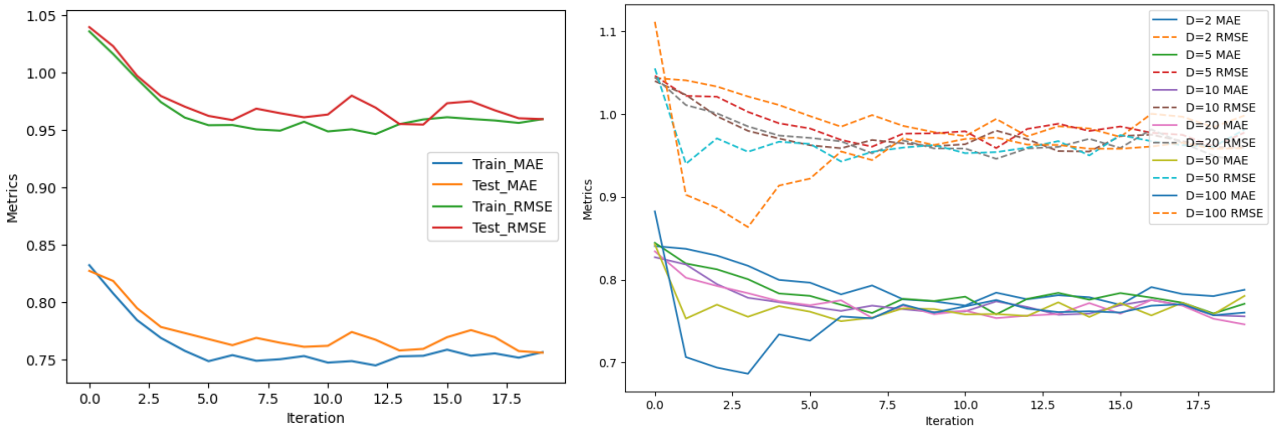


Figure 3: Tradition BPMF results

In the initial iteration (Iteration 0), both the training/testing MAE and RMSE values exceed 6, signaling a notable lack of precision in the model's early predictions. As the iteration count increases, the errors consistently decline before reaching a convergence point. The final errors (Iteration 99) are $\mathrm{Train}_{MAE} = 0.7469, \mathrm{Test}_{MAE} = 0.8608, \mathrm{Train}_{RMSE} = 0.9480, \mathrm{Test}_{RMSE} = 1.088$. Despite the starting point, all metrics converge within 20 iterations (Iteration), with $\mathrm{Train}_{MAE} = 0.7486, \mathrm{Test}_{MAE} = 0.8577, \mathrm{Train}_{RMSE} = 0.9504, \mathrm{Test}_{RMSE} = 1.090$. For the upcoming analysis, we will adjust the iteration count to $It = 20$ for the enhanced time efficiency.

5.2 Updated BPMF

With B introduced, we initialize the value as $B = \text{diag}([1, \ldots, D])$,



(a) D = 10

(b) D = 2, 5, 10, 20, 50, 100

Figure 4: Updated BPMF results

The updated BPMF starts at a more favorable point, achieving approximately 1 for both training/testing MAE as well as RMSE values. The final metrics are $\text{Train}_{MAE} = 0.7466, \text{Test}_{MAE} = 0.7462, \text{Train}_{RMSE} = 0.9488, \text{Test}_{RMSE} = 0.9420$, all of which are lower than those achieved by the traditional BPMF. For the traditional approach, the time complexity is given by $O(It \cdot (M + N) \cdot D)$, and this remains unchanged at $O(It \cdot (M + N + 1) \cdot D) = O(It \cdot (M + N) \cdot D)$ for the updated version. The introduction of additional diagonal matrix B does not add to the time complexity in Gibbs sampling, yet it results in a notable improvement, with a 13% decrease in Test MAE and a 13.5% decrease in Test RMSE.

The figure at right illustrates the impact on model performance from different values of D. The highest Test_{MAE} and Test_{RMSE} are 0.7879 and 0.9981 respectively, observed when $D = 2$. The lowest Test_{MAE} and Test_{RMSE}, in contrast, are 0.7461 and 0.9419 respectively, observed when $D = 20$. We can conclude that the model performance exhibits an initial increase followed by a decrease as the parameter D increases. A suitable choice for the number of latent features is either $D = 10$ or $D = 20$. Additionally, we observe that the outcomes achieved with $D = 2$ in the updated model surpass those of the traditional model with $D = 10$. The update allows for a reduction in the value of D while maintaining favorable results, contributing to time efficiency.

5.3 BPMF with Gaussian Mixture Model Clustering

We applied GMM to the user data matrix and present the BIC scores resulting from grid cross-validation parameter tuning below:

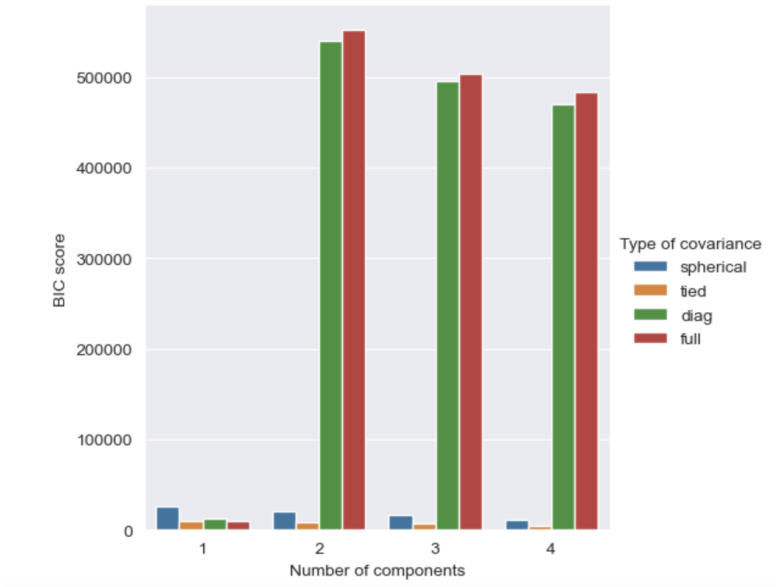
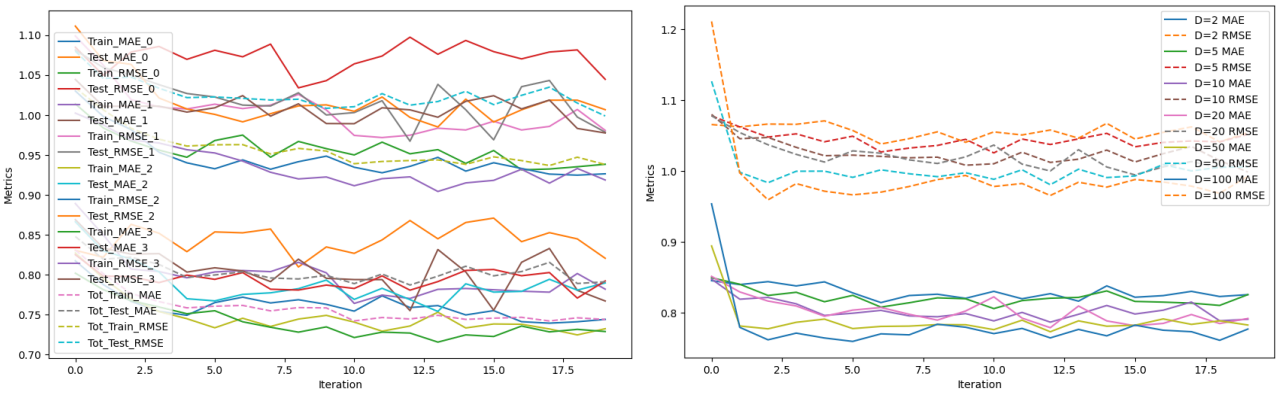


Figure 5: BIC score on different GMM parameters.

Hence, we opt for 4 components and employ tied covariance for clustering. Following the clustering process, we execute the updated BPMF on each cluster, varying the hyper-parameter D to observe its impact on the model within each specific cluster.



(a) D = 10

(b) D = 2, 5, 10, 20, 50, 100

Figure 6: BPMF with GMM results

The GMM has partitioned all users into four clusters, each with a population of 1907, 1593, 2754, and 2705, respectively. Considering $D = 10$, the individual Test_{MAE} values for each cluster are 0.7836, 0.8309, 0.7576, and 0.7555. The overall Test_{MAE} converges to approximately 0.7772. Furthermore, the Test_{RMSE} values for each cluster are separately 1.045, 0.9784, 0.9262, and 0.9185, with the overall Test_{RMSE} settling around 0.9987. These results are slightly higher than those achieved by directly applying the updated BPMF to the entire population. They remain, however, lower than those obtained from the traditional BPMF approach. In the series of trials with varying values of D, the optimal overall Test_{MAE} and Test_{RMSE} reaches 0.7772 and 0.9927, respectively, when $D = 100$.

6 Discussion

In terms of model performance, the models are ranked as follows: the updated BPMF, the updated BPMF with GMM, and lastly, the traditional BPMF. The incorporation of the diagonal matrix B

increases the overall parameter count, enhancing the model's flexibility and ultimately resulting in a reduced prediction error. Nevertheless, limitation appears when we integrate the updated BPMF with GMM, given the observed increase in error.

One potential explanation is that the existing clustering method may not effectively capture the distinctive characteristics of each user group. The present GMM exclusively considers the representation of users' movie-watching histories through one-hot encoding. This implies that the model's understanding and clustering of users are solely based on their quantity of movies watched within each genre, neglecting more detailed information about the content and characteristics of the movies, such as the director, the publication year, etc. Solution to address this limitation is gathering more comprehensive information about the movies themselves for more accurate clustering.

Another potential reason is that a higher number of users engaged in a single BPMF process may contribute to more accurate predictions in the resulting matrix. The collaborative information from a larger user population can enhance the model's ability to capture the patterns and relationships, i.e., the latent features. In such a scenario, it becomes unnecessary to address the limitation, especially considering that the introduction of additional GMM block just results in an increase in error by 0.3. When a new user joins the system, on the contrary, we only need to re-execute the BPMF among the specific user group to which they should belong, instead of the whole population. When the total number of clusters is set to 4, for instance, this operation results in saving 3/4 of the original execution time, which makes it a reasonable trade-off.

7 Conclusion

In this project, we have introduced updates to traditional BPMF methods ($R = U^T V$), involving two key modifications. We have initially introduced an extra diagonal matrix B to augment the model's expressiveness. Additionally, we have implemented a GMM clustering block before BPMF stage breaking down the original task into several subtasks. The former modification can bring a starting point closer to the convergence point in Gibbs sampling, and produces a lower error when utilizing the same number of iterations. The latter modification can help save the prediction time when a new user is invited to the system, which is valuable in real-world application. Overall, the optimal Test_{MAE} achieved throughout the process is less than 0.75. The outcome aligns with our expectations for the recommendation system, which can give accurate predictions and recommendations.

8 Reference

[1] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. http://dx.doi.org/10.1145/2827872

[2] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In Proceedings of the 25th international conference on Machine learning (ICML '08). Association for Computing Machinery, New York, NY, USA, 880–887. https://doi.org/10.1145/1390156.1390267

Appendix: Code

For the code implementation, please refer to:

github.com/Carlos-Xiao/Bayesian-Matrix-Factorization

Appendix: Contribution

Zicong Xiao

Responsible for the formula derivation and implementation of the updated BPMF process, including the parameter tuing. Also report the results found from comparative analysis.

Xiaoyang Sheng

Responsible for the updated BPMF with Gaussian Mixture Model Clustering, including EDA, GMM modeling and hyper-parameter tuning. Also write the report and presentation slides for this part.

Yulin Gao

Responsible for research of relevant Bayesian topics and data cleaning. Also writing the report and presentation slides.