Data science lab project: collaborative filtering

Belgacem Ben Ziada, Lucas Mebille, Mohamed Aloulou

AlloCinAI

1 Introduction

In this project, we focus on the *collaborative filtering* setting for movie recommendation, where the only available signal is the pattern of ratings that users assign to movies. Formally, we are given a sparse rating matrix $R \in \mathbb{R}^{U \times I}$, where R_{ui} denotes the rating provided by user u for movie i, and the vast majority of entries are unobserved. Our main objective is to learn a predictive model that can accurately estimate the missing entries of R, i.e.,

$$\hat{R}_{ui} \approx R_{ui}, \quad \text{for all } (u, i) \text{ where } R_{ui} \text{ is unknown,}$$

and thereby enable personalized movie recommendations.

To address this problem, we adopt a structured approach built around *matrix factorization* (MF), a widely used and highly effective technique in collaborative filtering. MF is well-suited to this setting because it learns low-dimensional latent representations for users and movies, capturing underlying preference patterns even in the presence of sparse data.

2 Data Analysis and Visualization

We begin with an exploratory analysis of the user–item rating matrix to characterize its sparsity, interaction patterns, and rating calibration. Unless stated otherwise, zeros denote *missing* entries. For visualization, missing values are rendered as zeros in colormaps to highlight the sparsity structure.

2.1 Sparsity and Interaction Patterns

The dataset comprises **610 users** and **4,980 items**, with **63,196 observed ratings**, resulting in a sparsity of **97.92%**. This extreme sparsity is further evidenced by the distribution of interactions: per-user counts exhibit a mean of **103.60**, a median of **46.50**, and a maximum of **1,619**, while per-item counts show a mean of **12.69**, a median of **6.00**, and a maximum of **219**. Such long-tailed distributions where a small subset of users and items dominate the interaction volume are typical in collaborative filtering and underscore the need for robust modeling techniques to address data imbalance.

As shown in Figure 1, the rating matrix reveals a scattered, non-block structure, with isolated nonzero entries and no discernible clustering. This pattern reflects the heterogeneity in user activity and item popularity, motivating the inclusion of user/item bias terms (Bobadilla et al., 2024), and regularization methods to mitigate overfitting, particularly for rare users/items.

The empirical distribution of observed ratings (Figure 2) is right-skewed, with a clear peak at **4** and substantial mass at **3** and **5**, while low ratings (1–2) are comparatively rare. This positivity bias is consistent with explicit-feedback datasets and motivates the inclusion of bias terms (μ, b_u, c_i) to account for global leniency/harshness and item popularity effects.

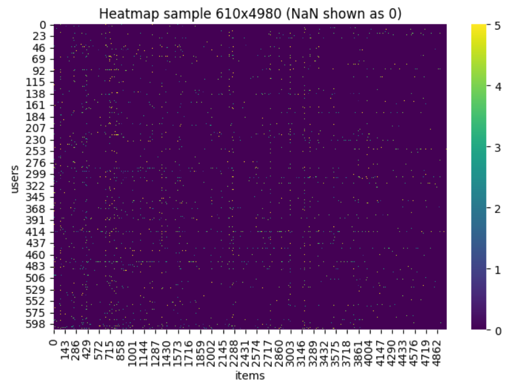

Figure 1: Heatmap of the rating matrix (color bar indicates the rating scale; missing values are displayed as 0 for visualization). The field is highly sparse, with scattered observations indicative of long-tailed activity on both axes.

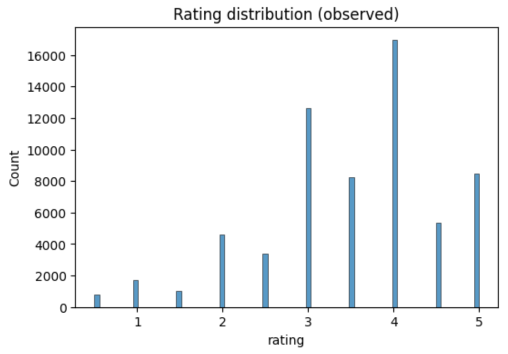

Figure 2: Distribution of observed ratings. The mass concentrates on ratings 3–5, with a peak on 4, indicating a positive-bias regime.

2.2 Rating Distribution and Genre Analysis

Items in the dataset are also associated with multiple genres. Since we don't use these features in our methods, we don't detail the analysis of those labels.

3 Baseline: Matrix Factorization with Gradient Descent

As a simple yet effective baseline, we learn a low-rank factorization of the sparse user–item matrix R, which captures shared structure in user preferences and item attributes. We evaluate two standard variants:

MF without bias. Each user u and each item i is associated with a learnable latent vector $\boldsymbol{u}_u, \boldsymbol{v}_i \in \mathbb{R}^k$, and the predicted rating is given by:

$$\hat{r}_{ui} = \boldsymbol{u}_u^\top \boldsymbol{v}_i, \quad \boldsymbol{u}_u, \boldsymbol{v}_i \in \mathbb{R}^k.$$

MF with bias. We incorporate the global mean rating μ, user bias b_u, and item bias c_i, in addition to the inner product $\boldsymbol{u}_u^\top \boldsymbol{v}_i$, which models the interaction between user u and item i:

$$\hat{r}_{ui} = \mu + b_u + c_i + \boldsymbol{u}_u^\top \boldsymbol{v}_i.$$

Here, k denotes the latent dimension, representing the number of hidden features for users and items.

We minimize the regularized squared error over the observed entries S:

$$\min_{\{\boldsymbol{u}_u\},\{\boldsymbol{v}_i\},\{b_u\},\{c_i\}} \sum_{(u,i)\in S} (r_{ui} - \hat{r}_{ui})^2 + \lambda_U \sum_u \|\boldsymbol{u}_u\|_2^2 + \lambda_V \sum_i \|\boldsymbol{v}_i\|_2^2 + \lambda_b \sum_u b_u^2 + \lambda_c \sum_i c_i^2. \quad (1)$$

The MF without bias is recovered by setting $b_u = c_i = 0$ and omitting their corresponding regularization terms. For simplicity, we may also set all regularization weights to be equal, i.e., $\lambda_U = \lambda_V = \lambda_b = \lambda_c = \lambda$.

3.1 SVD++

SVD++ (Koren, 2008) is an extension of the standard matrix factorization (MF) model that improves recommendation quality by incorporating additional information about *which movies a user has rated*, beyond the rating values themselves.

The classical MF formulation relies only on the explicit rating values and ignores potentially useful information contained in the set of movies a user has rated. Even if two users give very different scores, the fact that they have rated many of the same movies still signals shared interests. SVD++ captures this signal by enriching the user representation with information from all rated movies. Let

$$N(u) = \{ j : (u,j) \in S \}$$

denote the set of movies rated by user u. The SVD++ predictor becomes:

$$\hat{r}_{ui} = \mu + b_u + c_i + \Big(\boldsymbol{u}_u + \frac{1}{\sqrt{|N(u)|}} \sum_{j \in N(u)} \boldsymbol{y}_j \Big)^{\top} \boldsymbol{v}_i, \tag{2}$$

where each $\boldsymbol{y}_j \in \mathbb{R}^k$ is a learnable embedding associated with movie j. Intuitively, we augment the user vector $\boldsymbol{u}_u$ with the average contribution of all movies they have rated. The normalization factor $|N(u)|^{-1/2}$ ensures that users who have rated many movies do not dominate the model.

This enriched user representation allows the model to capture shared structure between users based on their rating histories. For instance, if two users have rated many of the same science-fiction movies, their latent representations will become more similar even if the scores they gave were different.

Objective and Regularization. The model parameters are learned by minimizing the regularized squared loss over all observed ratings:

$$\mathcal{L} = \sum_{(u,i) \in S} \Big(r_{ui} - \hat{r}_{ui} \Big)^2 + \lambda_P \|\boldsymbol{u}_u\|^2 + \lambda_Q \|\boldsymbol{v}_i\|^2 + \lambda_Y \sum_{j \in N(u)} \|\boldsymbol{y}_j\|^2 + \lambda_b (b_u^2 + c_i^2), \tag{3}$$

where $\lambda_P, \lambda_Q, \lambda_Y, \lambda_b$ are regularization hyperparameters that control overfitting for user vectors, item vectors, movie-embedding vectors, and biases. For simplicity, we set all regularization parameters to the same value λ.

Optimization with SGD. We optimize the loss equation 3 using stochastic gradient descent (SGD). For each observed pair (u, i), we compute the prediction error

$$e_{ui} = r_{ui} - \hat{r}_{ui}$$

and update the parameters by moving them in the negative gradient direction. For example:

$$\begin{aligned}
&\boldsymbol{u}_u \leftarrow \boldsymbol{u}_u + \eta\big(e_{ui}\boldsymbol{v}_i - \lambda_P \boldsymbol{u}_u\big),\\
&\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + \eta\big(e_{ui}(\boldsymbol{u}_u + \frac{1}{\sqrt{|N(u)|}} \textstyle\sum_{j \in N(u)} \boldsymbol{y}_j) - \lambda_Q \boldsymbol{v}_i\big),\\
&\boldsymbol{y}_j \leftarrow \boldsymbol{y}_j + \eta\big(e_{ui}|N(u)|^{-1/2}\boldsymbol{v}_i - \lambda_Y \boldsymbol{y}_j\big), \quad \forall j \in N(u),\\
&b_u \leftarrow b_u + \eta(e_{ui} - \lambda_b b_u), \quad c_i \leftarrow c_i + \eta(e_{ui} - \lambda_b c_i).
\end{aligned}$$

Here η is the learning rate. To make training more efficient, we cache the term

$$\boldsymbol{y}_u = |N(u)|^{-\frac{1}{2}} \sum_{j \in N(u)} \boldsymbol{y}_j$$

for each user and update it incrementally.

Hyperparameters. We tune the latent dimension k, learning rate η, regularization weights $\lambda_P, \lambda_Q, \lambda_Y, \lambda_b$ and the number of training epochs.

4 Results

To compare our models before publishing them on the external evaluation platform, we trained and evaluated them on the initial `ratings_train.npy` and `ratings_test.npy` datasets. The training set contains 31,598 ratings with a mean of 3.52/5.0 and a standard deviation of 1.05. The test set contains 31,598 ratings, with a mean of 3.51/5.0 and a standard deviation of 1.04. For all methods, we applied clipping to the predicted ratings to ensure they remained within the range (0.5, 5]. We also evaluated the impact of rounding predicted values to the nearest 0.5 unit, as in the original data.

4.1 Metrics

We evaluate each model using the following metrics:

RMSE (Root Mean Squared Error): Measures the square root of the average squared difference between predicted and actual ratings. Formally,

$$\text{RMSE} = \sqrt{\frac{1}{|S|}\sum_{(u,i)\in S}(r_{ui} - \hat{r}_{ui})^2}.$$

Lower values indicate better performance.

MAE (Mean Absolute Error): Measures the average absolute difference between predicted and actual ratings. Formally,

$$\text{MAE} = \frac{1}{|S|}\sum_{(u,i)\in S}|r_{ui} - \hat{r}_{ui}|.$$

Lower values indicate better performance.

Exact Accuracy: The percentage of predictions that exactly match the true rating. Formally,

$$\text{Exact Accuracy} = \frac{100}{|S|}\sum_{(u,i)\in S} 1\{\hat{r}_{ui} = r_{ui}\}.$$

4.2 Search Procedure

We conducted a grid search over the hyperparameters $(k, \eta, \lambda.)$, using the same regularization parameter for all regularization terms to reduce search complexity. The best configuration for each model was selected based on the mean validation RMSE and is summarized in Table 1.

Table 1: Best hyperparameters for each method.

Model	k	η	λ
MF with SGD (w/ bias)	64	0.01	0.01
MF with SGD (w/o bias)	32	0.005	0.02
SVD++	128	0.01	0.005

4.3 Results

The performance of each model, using the best hyperparameters, is reported in Table 2.

Table 2: Model performance on the test set.

Model	RMSE	MAE	Exact Acc. (%)	Train Time (s, CPU)
Average	*1.037*	*0.824*	*0.00*	***2***
MF SGD (w/o bias)	*0.921*	*0.710*	*0.18*	*22*
MF SGD (w/o bias, round)	*0.932*	*0.700*	*23.30*	
MF SGD (w/ bias)	*0.876*	*0.674*	*0.10*	*25*
MF SGD (w/ bias, round)	*0.889*	*0.664*	*24.52*	
SVD++	***0.871***	*0.670*	*0.10*	*85*
SVD++ (round)	*0.885*	***0.660***	***24.61***	

We first recover the results from Bobadilla et al. (2024), showing that the biased version of MF outperforms the unbiased version on distance-based metrics. We also observe that the rounding trick significantly increases exact accuracy across all methods, though at the cost of a slight increase in RMSE, as expected.

The results confirm that SVD++ outperforms both biased and unbiased matrix factorization across all metrics, achieving the lowest RMSE (0.871) and MAE (0.670) on the test set without rounding. This superiority is further validated by the cold-start vs. warm user analysis, which reveals nuanced insights into the model's behavior. For warm users (> 10 ratings), SVD++ reduces RMSE (0.866 vs. 0.889 for MF with bias) and MAE (0.691 vs. 0.708), demonstrating its ability to refine predictions as more user interactions become available. However, the improvement for cold-start users (≤ 10 ratings) is more subtle: while SVD++ slightly lowers RMSE (0.920 vs. 0.933) and MAE (0.760 vs. 0.763), the persistent performance gap between cold-start and warm users, coupled with the high RMSE variance (0.368) for cold-start users, suggests that sparsity remains a fundamental challenge, even for models leveraging implicit feedback. Notably, the smaller relative improvement for cold-start users may reflect the limitations of collaborative filtering in extreme sparsity, where implicit signals are less informative due to the lack of rated items. Nonetheless, SVD++ remains the most performant approach among those tested.

5 IGMC (NOT RETAINED)

We also implemented *IGMC* (Inductive Matrix Completion based on Graph Neural Networks) Zhang & Chen (2020) to probe a strong modern baseline. IGMC builds a small h-hop *enclosing subgraph* around each target pair (u, i) in the user–item bipartite graph (edges typed by rating value), encodes this subgraph with relation-aware GNN layers (R-GCN), then concatenates the target user/item embeddings and feeds them to a small MLP head to regress the rating (squared loss; optionally with adjacent-rating regularization). Although our preliminary runs yielded competitive RMSE, the project guidelines prohibit deep-learning components (including MLP heads). We therefore report MF and SVD++ as our official results and exclude IGMC from the final comparison.

6 CONCLUSION AND FUTURE WORK

We studied classical collaborative filtering for explicit ratings, implementing regularized MF (with and without biases) and SVD++. The bias-augmented MF provides a strong baseline, while SVD++ further improves RMSE by leveraging implicit feedback from users' rated items. We also prototyped IGMC, but excluded it from the official comparison to comply with the no–deep-learning constraint. We did not include a genre-aware model in this report; integrating the provided `namesngenre.npy` (e.g., genre-augmented item biases or feature-aware MF/Factorization Machines) is a natural next step to address cold start and popularity bias. Additional extensions include temporal effects (TimeSVD++), broader hyperparameter search, and simple ensembling of non–DL models.

REFERENCES

Jesús Bobadilla, Jorge Dueñas Lerín, Fernando Ortega, and Abraham Gutierrez. Comprehensive evaluation of matrix factorization models for collaborative filtering recommender systems. *International Journal of Interactive Multimedia and Artificial Intelligence*, 8(6):15–23, June 2024. ISSN 1989-1660. doi: 10.9781/ijimai.2023.04.008. URL `http://dx.doi.org/10.9781/ijimai.2023.04.008`.

Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08)*, pp. 426–434, New York, NY, USA, 2008. ACM. doi: 10.1145/1401890.1401944. URL `https://doi.org/10.1145/1401890.1401944`.

Muhan Zhang and Yixin Chen. Inductive matrix completion based on graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020. URL `https://openreview.net/forum?id=ByxxgCEYDS`.