Hybrid Recommender System using MovieLens100K dataset

Sri Satya Sai Immani
Case Western Reserve University
Cleveland, Ohio, USA

Introduction

In this project we implemented a baseline hybrid collaborative filtering recommender system built using the MovieLens 100K dataset, which contains 100,000 ratings (1–5) from 943 users on 1,682 movies. The model predicts explicit user-item ratings by combining collaborative filtering principles with user and item metadata, implemented by extending the surprise library's AlgoBase class. It constructs a baseline prediction using the global average rating, and adjusts it using the mean of available user and item feature values, such as age, gender, occupation, and genre indicators. This setup allows the model to make informed predictions even in cold-start scenarios where traditional collaborative filtering may fail due to limited interaction history. The simplicity of the model ensures interpretability and fast computation while still providing meaningful results. Evaluation using 5-fold cross-validation and a held-out test set produced consistent performance, achieving an average Root Mean Square Error (RMSE) of approximately 1.12 and Mean Absolute Error (MAE) around 0.94, demonstrating the model's capability to generalize well. This baseline hybrid model offers a solid foundation for further experimentation and development, such as integrating feature weights, learning algorithms, or deep neural approaches.

Model Description

The model implemented in this project is a **baseline hybrid collaborative filtering recommender system** designed to predict explicit ratings on the `MovieLens 100K` dataset. This dataset contains 100,000 ratings from 943 users on 1,682 movies, with ratings on a scale from 1 to 5. The model extends the `AlgoBase` class from the `surprise` library and integrates both collaborative filtering signals and user/item side metadata into a simple, interpretable architecture.

Unlike traditional matrix factorization or KNN-based approaches that rely solely on user-item interaction matrices, this model incorporates **user features** (such as age, gender, occupation) and **item features** (such as genre tags) to make predictions. The prediction logic is based on computing a **baseline estimate** that combines the global average rating, the mean of user features, and the mean of item features. The final predicted rating $\hat{r}_{ui}$ for a user u and item i is given by:

$$\hat{r}_{ui} = \mu + 0.5 \times \text{mean}(user\_features_u) + 0.5 \times \text{mean}(item\_features_i)$$

where μ is the global mean rating across the training dataset. This model does not involve any trainable parameters or optimization routines, making it lightweight and highly interpretable.

Feature	**Hybrid Baseline Model (This Work)**	**Traditional CF (User-based, Item-based, SVD)**
Model Type	Hybrid (Collaborative + Content-Based)	Pure Collaborative Filtering
Uses Metadata	Yes (User + Item features)	No
Handles Cold Start	Yes	No
Training Required	No (Rule-based)	Yes (Learning similarities or latent factors)
Interpretability	High (Based on feature means)	Low (Latent dimensions are abstract)
Performance Tuning	Not applicable	Tunable (e.g., latent factors, neighbors)
Examples	This model	KNN, SVD, NMF, ALS

Table 1: Comparison between our hybrid model and traditional collaborative filtering methods

Use Cases and Applications

This model is especially suited for:

- **Cold-start scenarios** where users or items have limited or no interaction history.
- **Interpretability-focused systems** where model transparency is critical.
- **Resource-constrained environments** requiring fast, computation-light predictions.

Limitations and Future Work

While the model offers interpretability and simplicity, its accuracy is limited compared to modern learnable recommender systems. Future enhancements may include:

- Replacing mean-based aggregation with weighted linear combinations.
- Incorporating feature importance via regression or attention mechanisms.
- Extending to neural hybrid models using embeddings and side information.

Methodology

All code is provided at the end of the document. MovieLens dataset is downloaed from the grouplens link provided and while evaluating the code there is no need to explicitly downlaod the data, the code handles downlaoding the data too. Experiment is performed in code.ipynb, executed in VSCode. Reproduction steps are detailed in the code file as comments. To enhance modularity and code reuse, all models and helper functions are structured within the project_code module.

Data Preprocessing

To ensure fair comparisons across all models, the datasets undergo the following pre-processing steps, DataPreProcessor() class is created to handle all the Data Pre-processing steps:

- **Data Acquisition and Inspection:** The dataset folder is downloaded from a trusted source. `ratings, users, items, genres, occupations` objects are created and used to load the necessary data. `load_data()` function loads data from a file and save it to the instance variable. This method assumes that the data is in a tab-separated format with no header but we also enabled this function to use any separator. `show_info()` function is used to print basic info about the dataframe.
- **Handling Missing Values:** We implemented a `clean_data()` function to handle null columns and rows, remove duplicate values and finally storing cleaned data in an instance variable called *cleaned_data*.
- **Data Splitting:** Both datasets were split into training and testing sets using `split_data()` for reliable model evaluation. We chose 80-20 split between training and test data sets.

Implementation Environment

The implementation was carried out using the following environment:

- **Programming Language:** Python.
- **Libraries and Frameworks:** The development process utilized several libraries and frameworks, including Scikit-learn Surprise, NumPy, Pandas, Matplotlib, os, sys.
- **Hardware Configuration:** The experiments were conducted VSCode.

Evaluation

To ensure robust performance estimation, **K-fold cross validation** was employed. WE tried to hardcode the prediction rule to predict.

$$\begin{aligned}\text{prediction} = \text{global\_mean} & \\ & + 0.5 \times \text{mean(user\_features)} \\ & + 0.5 \times \text{mean(item\_features)}\end{aligned}$$

Since, we are just using a base line model, there are no learnable hyper parameters in this model. Performance is assessed using the following metrics:

- **Root Mean Square Error (RMSE):** A standard regression metric used to measure the average magnitude of prediction error. It penalizes large errors more heavily than small ones, making it suitable for evaluating rating prediction tasks. A lower RMSE indicates better model performance and higher predictive accuracy.
- **Mean Absolute Error (MAE):** This metric represents the average of the absolute differences between predicted and actual ratings. It provides an intuitive measure of how close predictions are to the true values, with lower values indicating better performance.

These metrics were computed using both 5-fold cross-validation and a held-out test set, providing a robust evaluation of the model's ability to generalize.

Results

The recommendation model performance was evaluated on the MovieLens 100K dataset using two standard regression-based evaluation metrics: **Root Mean Square Error (RMSE)** and **Mean Absolute Error (MAE)**. These metrics quantify the deviation between the predicted and actual ratings. Lower RMSE and MAE scores indicate higher prediction accuracy, with RMSE penalizing larger errors more severely.

Cross-Validation Performance

To assess generalization, we performed 5-fold cross-validation on the training set. The hybrid model, which combines the global average rating with the mean values of user and item metadata, achieved a cross-validation RMSE of **1.1261** and an MAE of **0.9452**, as shown in Table 2. These results suggest that the model produces reasonably accurate predictions given its simple baseline nature and lack of learnable parameters.

Fold	RMSE	MAE
Fold 1	1.1283	0.9470
Fold 2	1.1247	0.9440
Fold 3	1.1251	0.9431
Fold 4	1.1257	0.9463
Fold 5	1.1267	0.9460
Average	**1.1261**	**0.9452**

Table 2: 5-Fold Cross-Validation RMSE and MAE on MovieLens 100K

Test Set Evaluation

After cross-validation, the final model was evaluated on a held-out 20% test set to assess its real-world prediction ability. The test RMSE was **1.1239**, and the test MAE was **0.9420**, as shown in Table 3. These results are consistent with the cross-validation performance, indicating strong generalization and confirming the model's stability and reliability.

Metric	**Score**
RMSE	1.1239
MAE	0.9420

Table 3: Final Evaluation on Held-Out Test Set

Conclusion

Across the `MovieLens 100K` dataset, the hybrid baseline model demonstrated promising performance by effectively leveraging both collaborative signals and side metadata. Despite its simplicity, the model achieved comparable baseline metrics (e.g., RMSE and MAE) to traditional collaborative filtering techniques such as KNN and SVD, particularly excelling in cold-start scenarios where user-item interactions were sparse. The integration of user and item features significantly enhanced interpretability, making the model suitable for applications demanding transparency.

However, the model's reliance on mean-based feature aggregation limited its ability to capture complex nonlinear interactions between users and items. Unlike trainable models, it lacked the flexibility to adapt to intricate preference patterns, which constrained its predictive accuracy in dense data regions.

To overcome these limitations, future improvements could focus on learning feature weights through linear regression or attention mechanisms, enabling more expressive combinations of side information. Additionally, embedding-based models or neural hybrid architectures could be explored to capture deeper user-item dynamics while still retaining the benefits of metadata integration.