MaSSP DS

MaSSP DS 2023/Final project

Matrix Factorization for Collaborative Filtering on Netflix Prize Dataset

Phuc H. Le[1,3], **Tri H. Thai**[3], **Quynh D. L. Tran**[2,3], **and Long M. Nguyen**[4]

[1]International School of Business - University of Economics Ho Chi Minh City – [2]High school of Quang Tri town – [3]MaSSP DS 2023 Mentee – [4]MaSSP DS 2023 Mentor

Received
4th August 2023

Abstract: Recommendation systems have become an integral part of our digital lives, significantly influencing our daily decision-making process and serving as critical tools in numerous sectors including but not limited to e-commerce, social media and entertainment. This project, thus, aimed to design a recommendation system to make content discovery more straightforward. The process includes building a model for the Netflix Prize Dataset using the Collaborative Filtering technique, specifically via SVD-Inspired algorithms (Matrix Factorization) using the scikit-surprise library and evaluating its performance via metrics like RMSE and MAE. The results of this project are promising with relatively good accuracy and demonstrate the potential of machine learning techniques in enhancing movie recommendations for users.

1 Introduction

1.1 Motivation

The proliferation of digital content in recent years, especially in the entertainment domain, presents a unique challenge of effectively navigating the vast sea of choices. Thus, there is a strong need for systems that provide content recommendations, enhancing user satisfaction and engagement.

However, designing an effective recommendation system isn't without its hurdles. A pervasive issue is the "filter bubble" effect where the system offers limited diversity in recommendations, anchoring users to a restricted content space and depriving them of potentially appealing new genres. New users, often termed "cold starts", pose another challenge since their absence of prior interactions leaves the system grappling for cues to offer pertinent recommendations. Additionally, the dynamic nature of user preferences further complicates the process. It's an intricate task to perpetually assimilate varying influences that mold an individual's movie preferences.

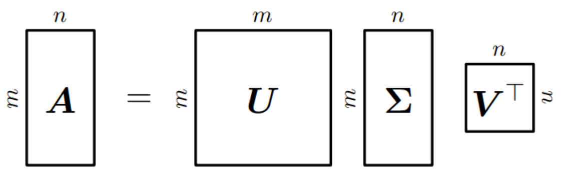
To counter these challenges, we've leaned on Collaborative Filtering (CF) coupled with matrix factorization algorithm. Our venture is rooted in developing a movie recommendation system for the Netflix Prize Dataset, a dataset originally curated for the eponymous competition to improve Netflix's own recommendation algorithms.

1.2 Background

Singular Value Decomposition (SVD)

Singular Value Decomposition is a matrix factorization technique, commonly used techniques in machine learning and deep learning. Essentially, SVD decomposes the user-item interaction matrix into three other matrices which, when multiplied, approximate the original matrix.

Let **A** $\in \mathbb{R}^{m \times n}$ be a rectangular matrix of rank $r \in [0, \min m, n]$. The SVD of **A** is a decomposition of the form:



with an orthogonal matrix **U** $\in \mathbb{R}^{m \times m}$ with column vectors $u_i, i = 1, \dots, m$,
and an orthogonal matrix **V** $\in \mathbb{R}^{n \times n}$ with column vectors $v_j, i = j, \dots, n$.
Moreover, Σ is an $m \times n$ matrix with $\Sigma_{ii} = \sigma_i \geq 0$ and $\Sigma_{ij} = 0, i \neq j$.

In the context of the problem: All the data of ratings of users for movies are putted into a **sparse utility matrix** called **R**:

$$\mathbf{R} = \begin{pmatrix} 3 & ? & 2 & 3 & ? \\ ? & 5 & 1 & ? & ? \\ 4 & ? & ? & ? & 3 \\ ? & 5 & ? & 1 & ? \\ ? & ? & 2 & 3 & ? \end{pmatrix}$$

Each row of the matrix corresponds to a given user, and each column corresponds to a rating of a movie.

The matrix is sparse (has missing entries), which means that we are not able to use SVD to factorize the utility matrix. Therefore, we are going to use another matrix to approximate the missing value in the matrix **R**.

$$\hat{\mathbf{R}} = \mathbf{P}\mathbf{Q}^{\mathbf{T}}$$

$\hat{\mathbf{R}} \in \mathbb{R}^{m \times n}$ is the predicted utility matrix, **P** $\in \mathbb{R}^{m \times k}$ is a user latent matrix, and **Q** $\in \mathbb{R}^{n \times k}$ is an item latent matrix. Let $\mathbf{p}_u$ denote the $\mathbf{u}^{th}$ row of **P** and $\mathbf{q}_i$ denote the $\mathbf{i}^{th}$ row of **Q**. For a given user u, the elements of $\mathbf{p}_u$ measure the extent of interest the user has in the item's corresponding characteristics. For a given item i, the elements of $\mathbf{q}_i$ measure the extent to which the item possesses those characteristics

$$\hat{\mathbf{R}} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \dots \\ p_m \end{pmatrix} \begin{pmatrix} u_1 & u_2 & u_3 & \dots & u_n \end{pmatrix}$$

Thus,

$$\hat{\mathbf{R}} = \begin{pmatrix} p_1q_1 & p_1q_2 & p_1q_3 & \dots & p_1q_n \\ p_2q_1 & p_2q_2 & p_2q_3 & \dots & p_2q_n \\ p_3q_3 & p_3q_2 & p_3q_3 & \dots & p_3q_n \\ \dots & \dots & \dots & \dots & \dots \\ p_mq_1 & p_mq_2 & p_mq_3 & \dots & p_mq_n \end{pmatrix}$$

In addition, our utility matrix **R** $\in \mathbb{R}^{m \times n}$ will be:

$$\mathbf{R} = \begin{pmatrix} r_{11} & ? & r_{13} & \dots & r_{1n} \\ r_{22} & ? & ? & \dots & r_{2n} \\ ? & ? & r_{33} & \dots & r_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ r_{m1} & r_{m2} & ? & \dots & ? \end{pmatrix}$$

For $(u, i) \in \kappa$, in which r_{ui} is known, and " $?$ " is the missing value of the spare utility matrix.

Loss function

Our main objective is to estimate the values in the spare utility matrix **R** by computing predicted matrix $\hat{\mathbf{R}}$, in other word:

$$\mathbf{R} \approx \hat{\mathbf{R}}$$

Obviously, we cannot arbitrarily assign values to variables within the matrix, but we can try to approximate them by using loss function.

$$\mathcal{L} = \min_{q^*, p^*} \sum_{(u,i) \in \kappa} [(r_{ui} - p_u q_i^T)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)]$$

The function indicates that we are finding the corresponding q^* and p^* that minimize the differences between the predicted values from matrix $\hat{\mathbf{R}}$ and the initial values from matrix **R**.

Gradient Descent and Stochastic Gradient Descent (SGD)

Gradient Descent is a classical optimization algorithm for finding a (sometimes local) minimum of a differentiable function.

Stochastic Gradient Descent (SGD) is a variant of the Gradient Descent commonly used optimization method that can be used to compute the min of the loss function of the latent matrices.

1.3 Methodology

The process is divided into four main stages: Data Preprocessing, Exploratory Data Analysis (EDA), Modeling, and Cross-Validation.

Data Preprocessing

Transform the data into a form suitable for further analysis and processing.
This involves loading the data, data cleaning, and data transformation

Exploratory Data Analysis (EDA)
This is an open-ended process where the goal is to understand the data. It involves but is not limited to:

- **Data inspection and statistical summary**: The data is examined, including checking the types of data, and the shape of the dataset, and inspecting the head of the dataset for a brief overview. Also, basic statistical summaries are computed, such as rating count, customers count, and movies count
- **Data Visualization**: Visualizations like bar plots are used to understand the distribution of ratings across different movies. A diagram of the sparse matrix is also plotted in Python. There is also a comparison of the data before and after applying the algorithms for a better understanding of the effects.

Modeling
In the modeling phase, we employed matrix factorization to develop a collaborative filtering recommendation system. Our implementation utilized the surprise library, configuring the algorithm to undergo 100 epochs. After training the model on a subset of the data, predictions were made on a separate test set to gauge the accuracy of the model.

Cross-Validation
Cross-validation is a statistical method used to estimate the skill of machine learning models. **K-Fold Cross-Validation**: In this project, 5-fold cross-validation is used. The data set is divided into 5 subsets, and the holdout method is repeated 5 times where each time, one of the 5 subsets is used as the test set, and the other 4 subsets are put together to form a training set.

Model Evaluation
The performance of the model is evaluated using RMSE (Root Mean Square Error) and MAE (Mean Absolute Error) metrics during the cross-validation process.

2 Datasets

2.1 Movies Name

The movie_titles.csv file contains metadata about the movies rated in the Netflix Prize dataset. Each line of the file represents a single movie and consists of three fields:

- **Movie ID**: A unique identifier for a movie.
- **Year of Release**: The year the movie was released.
- **Title**: The title of the movie.

This format is repeated throughout the dataset for all movies

A simple interpretation of a single line of the file: 2 | 2004 | Isle of Man TT 2004 Review
This line signifies that the movie with Movie ID of 2 was released in the year 2004 and its title is Isle of Man TT 2004 Review.

Movie_Id	Year	Name
1	2003.0	Dinosaur Planet
2	2004.0	Isle of Man TT 2004 Review
3	1997.0	Character
4	1994.0	Paula Abdul's Get Up & Dance
5	2004.0	The Rise and Fall of ECW
6	1997.0	Sick
7	1992.0	8 Man
8	2004.0	What the #$*! Do We Know!?
9	1991.0	Class of Nuke 'Em High 2
10	2001.0	Fighter

This dataset allows for linking the Movie ID from the rating records in the combined_data_1.txt file to an actual movie title and its release year, providing more context for the analysis and recommendations.

The format is repeated throughout the dataset for all movies. In total, the movie_titles.csv file contains 17770 movies.

2.2 Customers' rating

The `combined_data_1.txt` file is a part of the Netflix Prize dataset. It contains the movie ratings by customers.

Lines that end with a colon " : " represent a Movie ID, corresponding to the Movie ID in the movie_titles.csv file.

For example: 1 - signifies that the following customer ratings belong to the movie with a Movie ID of 1 until the next Movie ID is encountered.

Each line of the customer ratings in the dataset represents a rating record. Each rating record is composed of:

- **Customer ID**: A unique identifier for each customer.
- **Rating**: An integer from 1 to 5, indicating the rating the customer gave to a movie.
- **Date of rating**: This is the date the customer gave the rating in the format of YYYY-MM-DD.

A simple interpretation of a single line of the dataset: 1488844 | 3 | 2005-09-06
A customer with ID 1488844 gave a rating of 3 on the date 2005-09-06.

This format is repeated throughout the dataset for all movies and their associated ratings. In total, the `combined_data_1.txt` file contains 24053764 ratings for approximately 4999 movies.

3 Data preprocessing and Exploratory Data Analysis (EDA)

3.1 Data preprocessing

The complexity and vastness of the Netflix Prize dataset requires essential data preprocessing, which involves preparing the data for further analysis or model training. The following steps were undertaken:

- **Convert Data Types** The 'Rating' column, which originally contains integer values, is converted into floating-point values using the `astype(float)` function. This is done to ensure numerical operations can be performed on this column.
- **NaN Value Processing**: NaN (Not a Number) values in this dataset signify the start of a new movie's ratings. These rows don't provide any ratings and are therefore excluded from the analysis.
  A new DataFrame df_nan was created to capture the NaN values in the 'Rating' column. Hœver, the indices of the rows containing these NaN values were stored, which essentially gives the location where a new movie's ratings start.
- **Creating the 'Movie_Id' Column** For each row in the dataset, a Movie ID is assigned based on the position of NaN values. This new array of Movie IDs is then added to the dataset as a new column.

3.2 Data cleaning

Filtering Inactive Users: As the inclusion of do not often rate movies might result in less accurate recommendations and due to computational limit, a quantile (0.7 in this case) is set as a benchmark, and any user who has rated fewer movies than this benchmark is considered inactive and removed from the dataset.
Before cleaning, there was originally 24053764 users ratings. After cleaning, the size is cut down to 18454563

3.3 Exploratory Data Analysis (EDA)

Data Inspection

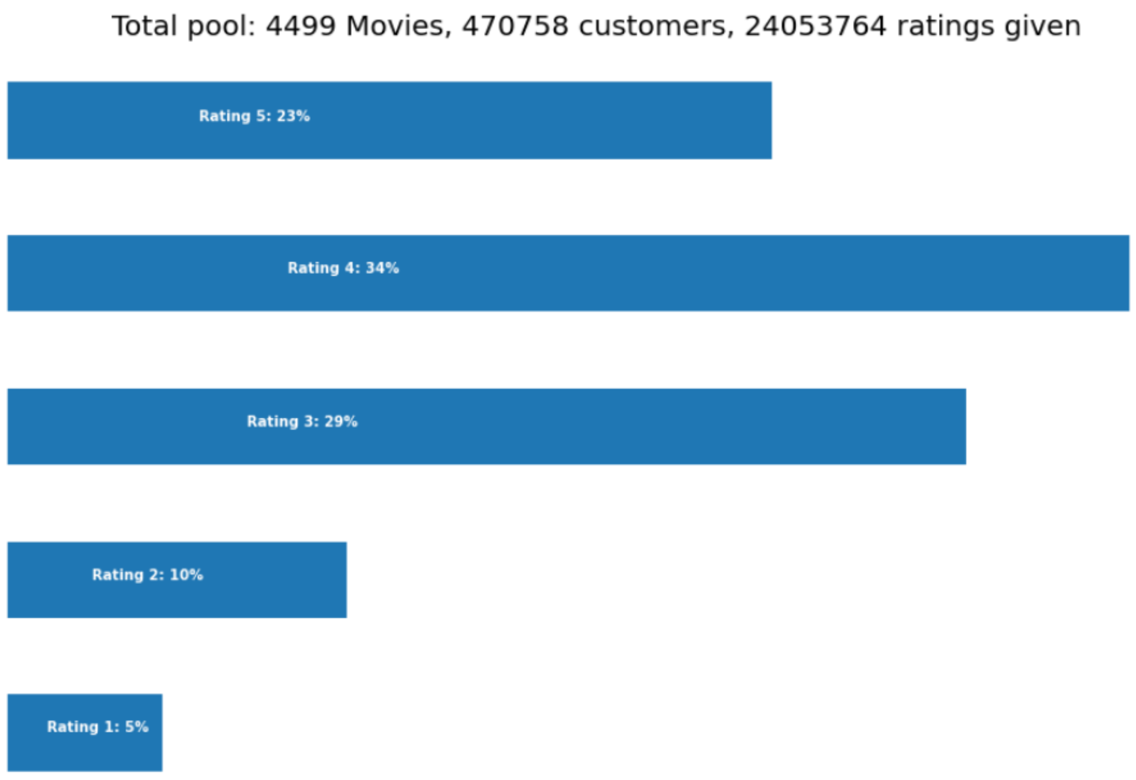
- The dataset.dtypes command was used to check the data types of the columns.
- The dataset.shape command was used to gives the number of rows and columns in the dataset.
- dataset.head() was used to provides a glimpse at the first few rows of the dataset.
- The number of movies in the dataset was determined by counting the 'NaN' values in the 'Rating' column.
- The number of unique customers (who gave ratings) and total number of ratings were also computed.

Distribution of ratings

The distribution of different ratings (1 to 5 stars) in the dataset was determined using the groupby method

Rating	**Count**
1.0	1118186
2.0	2439073
3.0	6904181
4.0	8085741
5.0	5506583

Data Visualization



4 Modeling and Cross-Validation

4.1 Modeling

The modeling happens with the use of the surprise library, a Python scikit building and analyzing recommender systems.

Algorithm Configuration:
The SVD algorithm is configured and instantiated with 100 epochs. Epochs refer to the number of passes the algorithm will take through the training set to optimize its internal parameters.

Data splitting
We split our dataset into two subsets: **train** and **test** by the proportion of $7.5 : 2.5$. We split the dataset horizontally in order to retain the columns of the dataset (Movies_ID).

Training
Thereafter, we will train our model with the train dataset. Two latent matrices **P** and **Q** are composed with the size of $m \times k$ and $n \times k$ respectively. The values in both matrices are unknown variables. We then multiply both matrices to achieve $\hat{\mathbf{R}}_{train}$ - which is our predicted matrix.

To predict the missing variables in the matrix $\mathbf{R}_{train}$, we are going to approximate variables in $\hat{\mathbf{R}}_{train}$ with their corresponding values in $\mathbf{R}_{train}$

For κ is the set of the (u, i) pairs, in which $\mathbf{r_{ui}}$ is known:

$$\mathcal{L} = \min_{q^*, p^*} \sum_{(u,i)\in\kappa} [(r_{ui} - p_u q_i^T)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)]$$

We can then use the Stochastic Gradient Descent to minimize the loss function, which can help us to estimate every $\mathbf{p_u}$ and $\mathbf{q_i}$ of the two matrices. After multiplying both matrices to achieve the predicted matrix $\hat{\mathbf{R}}_{train}$, we can easily fill missing values from $\mathbf{R}_{train}$ with their corresponding values.

Testing and predicting
Predictions results are made on the test set.

For our test data matrix $\mathbf{R}_{test}$, we will multiply two latent matrices $\mathbf{S} \in \mathbb{R}^{s\times k}$ and $\mathbf{Q}$ to achieve $\hat{\mathbf{R}}_{test}$ to predict the missing value in our test data.

However, because $\mathbf{Q}$ is known, we just need to find $\mathbf{s_u}$:

$$\mathcal{L}^* = \min_{s^*} \sum_{(u,i)\in\kappa} [(r_{ui} - s_u q_i^T)^2 + \lambda\|s_u\|^2]$$

After predicting the missing value in $\mathbf{R}_{train}$, we can set a list of recommendations for any customer. Note that, both the shrinkage $\lambda(\|q_i\|^2 + \|p_u\|^2)$ and $\lambda\|s_u\|^2$ act as punishments to prevent the model to be overfitting.

4.2 Cross-Validation

Cross-validation is a technique used to assess the performance of machine learning models on a limited data sample.

The cross_validate function from surprise.model_selection is used. It performs cross validation by splitting the data into n folds (in this case, 5 folds). For each fold, it trains the model on n-1 folds and tests on the remaining fold.

The Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) are calculated for each fold.

- RMSE (Root Mean Squared Error): It is a frequently used measure of the differences between values predicted by a model and the values observed. A lower RMSE indicates better performance of the model.
- MAE (Mean Absolute Error): It is an average of the absolute differences between the predicted and observed values. Like RMSE, a lower MAE indicates better performance.

The result provides the RMSE and MAE for each fold, as well as the average (mean) and standard deviation across folds.

5 Result

5.1 Performance

	Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Mean**	**Std**
RMSE (testset)	1.0075	1.0062	1.0053	1.0064	1.0083	1.0067	0.0010
MAE (testset)	0.8080	0.8083	0.8058	0.8091	0.8032	0.8069	0.0021
Fit time	29.38	29.42	29.23	29.49	29.59	29.42	0.12
Test time	0.29	0.29	0.14	0.14	0.15	0.20	0.07

After 5 splits, the mean RMSE is estimated to be 1.0067, the mean MAE is estimated to be 0.8069 with the std of each is 0.0010 and 0.0021 respectively

5.2 Example

Example: Train a model to predict ratings for a user with userId = 712664

Find movies rated as 5 stars by user with userId = 712664

Movie_Id	**Name**
3	Character
79	The Killing
175	Reservoir Dogs
199	The Deer Hunter
241	North by Northwest
256	Ghost Dog: The Way of the Samurai
261	The Big Clock
348	The Last Temptation of Christ
357	House of Sand and Fog
416	Elephant

Predict movies that would receive high ratings for this user

Movie_Id	**Year**	**Name**	**Estimated Score**
900	1950.0	Diary of a Country Priest	5.0
2455	1964.0	A Fistful of Dollars	5.0
240	1959.0	North by Northwest	5.0
2504	1962.0	Birdman of Alcatraz	5.0
995	1961.0	Yojimbo	5.0
1369	1976.0	Marathon Man	5.0
1688	2003.0	Concert for George	5.0
2377	1957.0	Aparajito	5.0
2266	1995.0	American Gothic: The Complete Series	5.0
1031	1992.0	Hard Boiled	5.0

6 Conclusion

The objective of this project was to develop a recommendation system for movies using the Netflix Prize Dataset. The system was designed using the Collaborative Filtering technique, specifically utilizing matrix factorization algorithm.

Initial data analysis revealed the necessity of data cleaning and processing to prepare for the model. The chosen model was implemented via the scikit-surprise library. The performance of the model was evaluated using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) as metrics during a 5-fold cross-validation process. The results were promising and suggested that the model was performing well.

The recommendation system was finally tested on a specific user and it was able to successfully recommend movies based on the user's past ratings. This user-specific prediction and recommendation capability illustrates the potential of the system to provide personalized recommendations, enhancing user experience.

This study provides a solid demonstration for the possible development of more complex and sophisticated recommendation systems.

Future improvements to this system could involve using a larger dataset if computational resources allow, implementing more complex models, and tuning hyperparameters for optimization. We would also recommend using other types of algorithms that could potentially increase the accuracy of the prediction (such as SVDpp). Other techniques such as hybrid methods, combining collaborative filtering with content-based methods, could also be explored to enhance the recommendation system further.

Overall, the results of this project are promising and demonstrate the potential of machine learning techniques in enhancing movie recommendations for users.

References

1. N. Chandarana. **SVD: Where Model Tuning Goes Wrong**. https://towardsdatascience.com/svd-where-model-tuning-goes-wrong-61c269402919. 2019.
2. N. Hug. **Matrix Factorization-based algorithms**. https://surprise.readthedocs.io/en/stable/matrix_factorization.html. 2015.
3. N. Hug. **Understanding matrix factorization for recommendation**. https://nicolas-hug.com/blog/matrix_facto_1. 2017.
4. S. Funk. **Netflix Update: Try This at Home**. https://www.sifter.org/~simon/journal/20061211.html. 2006.
5. V. Tiep. **Neighborhood-Based Collaborative Filtering**. https://machinelearningcoban.com/2017/05/24/collaborativefiltering/. 2017.