Recommender System for Movies

Using Matrix Factorization

Binh Minh TRAN

February 2025

Mentor : Prof. Célia Da Costa Pereira

Maître de Conférences (PhD, HDR), Université Côte d'Azur

STID - SPARKS (I3S Lab)

A Personal Project

GitHub: github.com/RecSys-Movies

Contents

1 Introduction

This project focuses on building and exploring a recommendation system using Matrix Factorization to improve the accuracy of predictions regarding user preferences. By applying Matrix Factorization, where users and movies are modeled as feature vectors, I optimize the alignment between them through a training process and Loss function optimization. I combine existing research with my own ideas and explore ways to develop them further. The final results show an improvement in the accuracy of predicting movies that users may be interested in. Furthermore, I tried to use PCA to reduce the matrix dimension with the objective of decreasing computation time while keeping the error increase minimal and drawing a few observations.

2 Notation

Notation	Description
$r(i,j)$	scalar; $= 1$ if user j rated movie i, $= 0$ otherwise
$y(i,j)$	scalar; rating given by user j for movie i (if $r(i,j) = 1$ is defined)
$\mathbf{w}^{(j)}$	vector; feature vector of user j
$\mathbf{x}^{(i)}$	vector; feature vector of movie i
$b^{(i,j)}$	scalar; bias parameter for movie i and user j
$b_u^{(j)}$	scalar; bias parameter for user j
$b_m^{(i)}$	scalar; bias parameter for movie i
b_u	vector; bias parameters for user j
b_m	vector; bias parameters for movie i
n_u	number of users
n_m	number of movies
k	number of features
$\mathbf{X}$	matrix of vectors $\mathbf{x}^{(i)}$
$\mathbf{W}$	matrix of vectors $\mathbf{w}^{(j)}$
$\mathbf{B}$	matrix of bias parameters $b^{(i,j)}$
$\mathbf{R}$	matrix of elements $r(i,j)$
$\mathbf{Y}$	matrix of elements $y(i,j)$
$\hat{\mathbf{Y}}$	estimated matrix of $\mathbf{Y}$
$\mathbf{Ynorm}$	matrix of normalized elements of $y(i,j)$

3 Matrix Factorization

Matrix Factorization is a method used in Collaborative Filtering [1], a technique that automatically predicts an active user's interest by gathering rating information from other similar users or items. It aims to learn the latent factors that capture relationships between users and items (e.g., movies). In this approach, each latent factor represents a hidden characteristic that influences how users rate movies. For movies, each factor might correspond to characteristics such as genre, mood, or style, which contribute to the rating. For users, each factor represents their preference for the corresponding characteristics in the movies they rate highly.

Imagine we have a rating matrix Y with dimension n_m movies $\times$ n_u users where $y(i,j)$ is the rating given by user j for movie i. The latent factors are not the vectors $y(i)$ or $y(j)$ in the matrix Y; they are defined by two matrices, X and W, where:

- The matrix of characteristics of the movies with dimension $n_m \times k$ (movies $\times$ features) includes feature vectors for movies $x^{(i)}$.

- The matrix of preferences of the users with dimension $k \times n_u$ (features $\times$ users) includes feature vectors for users $w^{(j)}$.

The scalar product of X and W represents the collaboration between the features of movies and the features of users, capturing the latent factors that influence how users rate movies. This results in the approximated rating matrix $\hat{Y}$, with the expected value $E(\hat{Y}) = Y$, meaning that the predicted ratings should be as close as possible to the actual ratings.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}^{(0)} \\ \mathbf{x}^{(1)} \\ \vdots \\ \mathbf{x}^{(n_m-1)} \end{bmatrix} \cdot \mathbf{W} = \begin{bmatrix} \mathbf{w}^{(0)} & \mathbf{w}^{(1)} & \cdots & \mathbf{w}^{(n_u-1)} \end{bmatrix}$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \mathbf{x}^{(0)} \cdot \mathbf{w}^{(0)} & \cdots & \mathbf{x}^{(0)} \cdot \mathbf{w}^{(n_u-1)} \\ \mathbf{x}^{(1)} \cdot \mathbf{w}^{(0)} & \cdots & \mathbf{x}^{(1)} \cdot \mathbf{w}^{(n_u-1)} \\ \vdots & & \vdots \\ \mathbf{x}^{(n_m-1)} \cdot \mathbf{w}^{(0)} & \cdots & \mathbf{x}^{(n_m-1)} \cdot \mathbf{w}^{(n_u-1)} \end{bmatrix}$$

Note that we can also minimize the distance between the approximated rating matrix $\hat{Y}$ and the rating matrix Y without using the matrices X and W by randomly initializing $\hat{Y}$ to fit the model. However, this approach significantly increases computing time due to the vast number of values that need to be computed simultaneously as the matrix size grows. In contrast, the scalar product between X and W allows the model to converge more quickly.

Moreover, in practice, certain users may have extreme positive or negative opinions about a movie, even without having seen it, or some movies may be more highly rated due to the presence of popular actors. These are examples of bias. To improve the precision of the predictions, it is essential to incorporate a bias term [2], which leads to the refined expression:

$$\hat{Y} = X \cdot W + b_u^{(j)} + b_m^{(i)} + \mu$$

where $b_u^{(j)}$ is the user bias - observed deviations of user j, and $b_m^{(i)}$ is the movie bias - observed deviations of movie i. For example, movie i tends to receive $b_m^{(i)}$ more or fewer scores when compared to other movies, and user j tends to give an average of $b_u^{(j)}$ fewer or more scores for all their movies when compared to other users. μ is represents the average of all assigned ratings

$$\hat{\mathbf{Y}} = \begin{bmatrix} \mathbf{x}^{(0)} \cdot \mathbf{w}^{(0)} + b_u^{(0)} + b_m^{(0)} + \mu & \cdots & \mathbf{x}^{(0)} \cdot \mathbf{w}^{(n_u-1)} + b_u^{(n_u-1)} + b_m^{(0)} + \mu \\ \mathbf{x}^{(1)} \cdot \mathbf{w}^{(0)} + b_u^{(0)} + b_m^{(1)} + \mu & \cdots & \mathbf{x}^{(1)} \cdot \mathbf{w}^{(n_u-1)} + b_u^{(n_u-1)} + b_m^{(1)} + \mu \\ \vdots & & \vdots \\ \mathbf{x}^{(n_m-1)} \cdot \mathbf{w}^{(0)} + b_u^{(0)} + b_m^{(n_m-1)} + \mu & \cdots & \mathbf{x}^{(n_m-1)} \cdot \mathbf{w}^{(n_u-1)} + b_u^{(n_u-1)} + b_m^{(n_m-1)} + \mu \end{bmatrix}$$

Then, to learn the feature vectors $x^{(i)}$ and $w^{(j)}$ or the bias terms, the regularized squared error equation on the set of known ratings [2] is:

$$\min_{\mathbf{x},\mathbf{w},\mathbf{b_u},\mathbf{b_m}} \frac{1}{2} \sum_{(i,j)\in S} \left(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + \mathbf{b_u}^{(j)} + \mathbf{b_m}^{(i)} + \mu - y(i,j) \right)^2 + \underbrace{\frac{\lambda}{2} \left(||\mathbf{w}^{(j)}||^2 + ||\mathbf{x}^{(i)}||^2 + (\mathbf{b_u}^{(j)})^2 + (\mathbf{b_m}^{(i)})^2 \right)}_{\text{regularization}} \tag{1}$$

Where S is the set of all user-item pairs (i, j) with actual ratings.

After, we learn the parameters using Gradient Descent.

$$w^{(j)} = w^{(j)} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial w^{(j)}} \tag{2}$$

$$x^{(i)} = x^{(i)} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial x^{(i)}} \tag{3}$$

$$b_m^{(i)} = b_m^{(i)} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial b_m^{(i)}} \tag{4}$$

$$b_u^{(j)} = b_u^{(j)} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial b_u^{(j)}} \tag{5}$$

Where :

- $\frac{\partial \mathcal{L}}{\partial w^{(j)}} = \sum_{(i,j)\in S} \left(\left(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + b_u^{(j)} + b_m^{(i)} \right) - y(i,j) \right) \cdot \mathbf{x}^{(i)} + \lambda w^{(j)}$ represents the derived loss function associated with the parameter $w^{(j)}$.
- $\frac{\partial \mathcal{L}}{\partial x^{(i)}} = \sum_{(i,j)\in S} \left(\left(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + b_u^{(j)} + b_m^{(i)} \right) - y(i,j) \right) \cdot \mathbf{w}^{(j)} + \lambda x^{(i)}$ represents the derived loss function associated with the parameter $x^{(i)}$.
- $\frac{\partial \mathcal{L}}{\partial b_m^{(i)}} = \sum_{(i,j)\in S} \left(\left(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + b_u^{(j)} + b_m^{(i)} \right) - y(i,j) \right) + \lambda b_m^{(i)}$ represents the derived loss function associated with the parameter $b_m^{(i)}$.
- $\frac{\partial \mathcal{L}}{\partial b_u^{(j)}} = \sum_{(i,j)\in S} \left(\left(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + b_u^{(j)} + b_m^{(i)} \right) - y(i,j) \right) + \lambda b_u^{(j)}$ represents the derived loss function associated with the parameter $b_u^{(j)}$.
- α is a learning rate (adam-optimizer)
- $\mathcal{L}$ is the loss function 1.

4 Methodology

4.1 Dataset

In this project, two datasets are used: the ratings dataset and the list of movies dataset. These datasets were created by MovieLens on September 26, 2018, with data from 610 users and more than 100,000 ratings. They belong to the MovieLens Latest Datasets collection.

4.2 Pre-processing data

Matrix Y: This project processes the data from the ratings dataset to construct a matrix with dimensions of 9724 movies by 610 users, denoted as Y, representing user ratings j for each movie i. The matrix Y is populated with all available ratings, with any missing values (NAs) replaced by 0. It should be noted that, in most applications, Y is a sparse matrix since only a small fraction of its elements are non-zero [3]. In this project's dataset, about 1.7% of the elements are non-zero. Ratings within Y range from 0.5 to 5, with any values outside this range reset to 0, indicating an unassigned rating.

The objective is to predict the 0-value cells, which represent non-attributed ratings.

For the purpose of exploring the recommendation system, I included one additional user [Me] who has rated all the movies that I considered as my preferences in the movie dataset. Hence, the dimension of Y is now 9724 **x** 611.

Matrix R: This matrix is based on matrix Y, including 1 for ratings that have been assigned and 0 otherwise.

Matrix B: In practice, certain users may have extreme positive or negative opinions about a movie, even without having seen it, or some movies may be more highly rated due to the presence of popular actors. These are examples of bias. Incorporating bias into the model training process is crucial to accurately capture and explain the underlying characteristics.

The bias for user j is calculated as:

$$b_u^{(j)} = \bar{y}(j) - \mu$$

Where:
- μ represents the average of all assigned ratings.
- $\bar{y}(j)$ is the average rating given by user j, calculated as:

$$\bar{y}(j) = \frac{\sum_{i=0}^{n_m-1} y(i,j) \cdot r(i,j)}{\sum_{i=0}^{n_m-1} r(i,j)}$$

The bias for movie i is calculated as:

$$b_m^{(i)} = \bar{y}(i) - \mu$$

Where:
- μ represents the average of all assigned ratings.
- $\bar{y}(i)$ is the average rating of movie i

The bias for movie i given by user j is calculated as:

$$b^{(i,j)} = b_m^{(i)} + b_u^{(j)}$$

4.3 Initial Model Selection

It is important to note that in this context, the matrix $\hat{Y}$ no longer contains any cells with a value of 0, which represent non-attributed ratings, as these have been replaced by predicted ratings. Additionally, the predicted ratings may exceed the maximum value of 5 on the rating scale.

The objective is to minimize the distance between $\hat{Y}$ and Y. To achieve this, a loss function $\mathcal{L}$ is used to quantify this distance. This loss function is specifically designed to penalize the model parameters, thus facilitating an effective reduction in the overall error.

The loss function $\mathcal{L}$ is calculated as presented in [2], but specifically for this project:

$$\begin{aligned}\mathcal{L}(\mathbf{x^{(i)}}, \mathbf{w^{(j)}}, \mathbf{b_u}^{(j)}, \mathbf{b_m}^{(i)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + \mathbf{b_u}^{(j)} + \mathbf{b_m}^{(i)} + \mu - y(i,j)\right)^2 \\ + \underbrace{\frac{\lambda}{2}\left(||\mathbf{w}^{(j)}||^2 + ||\mathbf{x}^{(i)}||^2 + (\mathbf{b_u}^{(j)})^2 + (\mathbf{b_m}^{(i)})^2\right)}_{\text{regularization}}\end{aligned} \tag{6}$$

Where:

- The notation $(i, j) : r(i, j) = 1$ represents the set of observed ratings, indicating that the corresponding cell in the rating matrix R has a value of 1. It is important to exclude cells where $r(i, j) = 0$ to prevent the model from adjusting parameters to fit predicted ratings near undefined values (NAs).

4.4 Model Adjustment

After executing the initial model, it was observed that the resulting cost value was relatively high. Therefore, adjustments to the bias are intended to determine whether this modification can reduce the total cost. Specifically, the model adjustment contains only one bias term, which is a combination of two bias terms, $b_u^{(j)}$ and $b_m^{(i)}$.

Basically, we have the bias matrix B, which includes the bias $b^{(i,j)}$ of user j for movie i. This bias is defined as:

$$b^{(i,j)} = b_m^{(i)} + b_u^{(j)}$$

Where $b_m^{(i)}$ represents the bias of movie i, and $b_u^{(j)}$ represents the bias of user j.

The predicted rating matrix is now defined as:

$$\hat{\mathbf{Y}} = X \cdot W + B$$

$$\hat{\mathbf{Y}} = \begin{bmatrix} \mathbf{x}^{(0)} \cdot \mathbf{w}^{(0)} + b^{(0,0)} + \mu & \cdots & \mathbf{x}^{(0)} \cdot \mathbf{w}^{(n_u-1)} + b^{(0,n_u-1)} + \mu \\ \mathbf{x}^{(1)} \cdot \mathbf{w}^{(0)} + b^{(1,0)} + \mu & \cdots & \mathbf{x}^{(1)} \cdot \mathbf{w}^{(n_u-1)} + b^{(1,n_u-1)} + \mu \\ \vdots & & \vdots \\ \mathbf{x}^{(n_m-1)} \cdot \mathbf{w}^{(0)} + b^{(n_m-1,0)} + \mu & \cdots & \mathbf{x}^{(n_m-1)} \cdot \mathbf{w}^{(n_u-1)} + b^{(n_m-1,n_u-1)} + \mu \end{bmatrix}$$

Then, the squared error equation for this adjustment is as follows:

$$\mathcal{L}(\mathbf{x^{(i)}}, \mathbf{w^{(j)}}, \mathbf{b}^{(i,j)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + \mathbf{b}^{(i,j)} + \mu - y(i,j) \right)^2 + \underbrace{\frac{\lambda}{2} \left(||\mathbf{w}^{(j)}||^2 + ||\mathbf{x}^{(i)}||^2 + (\mathbf{b}^{(i,j)})^2 \right)}_{\text{regularization}} \tag{7}$$

4.4.1 Further Adjustments

First of all, I decided not to include μ (the fixed factor in the model) in the prediction term, as shown in equation 1. When μ is included, the average of all assigned ratings, μ, dominates the predicted ratings, causing the bias term and the scalar product of X and W to take a secondary role. This may prevent the model from effectively learning the interactions and biases, leading to slower convergence. I want the model to focus more on the interactions between users and movies, so I removed μ to allow the model to better capture the latent factors. Then,

$$\hat{Y} = X \cdot W + b_u^{(j)} + b_m^{(i)}$$

Now, the system no longer includes the fixed term μ, which previously played an important role in predicting ratings. Instead, the system will learn from the training parameters X, W,

and the biases. One key point is that I do not want the model to fit the rating matrix Y, but rather the normalized rating matrix Y_{norm}, with the goal of focusing on relative factors rather than absolute ones. For example, due to the popularity of certain films, some ratings may be higher than average. Specifically, we aim to normalize the actual ratings where $r(i,j) = 1$.

The normalized matrix Y is calculated as:

$$Y_{\text{norm}}(i,j) = (y(i,j) - \bar{y}(i)) \cdot r(i,j)$$

Where:
- $\bar{y}(i)$ is the average rating of movie i.

$$\bar{y}(i) = \frac{\sum_{j=0}^{n_u-1} y(i,j) \cdot r(i,j)}{\sum_{j=0}^{n_u-1} r(i,j)}$$

The objective is to minimize the distance between $\hat{Y}$ and Y_{norm}. To achieve this, a loss function $\mathcal{L}$ is used to quantify this distance. This loss function is specifically designed to penalize the model parameters, thus facilitating an effective reduction in the overall error.

$$\begin{aligned}\mathcal{L}(\mathbf{x^{(i)}}, \mathbf{w^{(j)}}, \mathbf{b_u^{(j)}}, \mathbf{b_m^{(i)}}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + \mathbf{b_u}^{(j)} + \mathbf{b_m}^{(i)} - y_{\text{norm}}(i,j)\right)^2 \\ + \underbrace{\frac{\lambda}{2}\left(||\mathbf{w}^{(j)}||^2 + ||\mathbf{x}^{(i)}||^2 + (\mathbf{b_u}^{(j)})^2 + (\mathbf{b_m}^{(i)})^2\right)}_{\text{regularization}} \end{aligned} \tag{8}$$

Where:
- The normalized rating in the matrix **Ynorm** is denoted as $y_{\text{norm}}(i,j)$

This further adjusment would be called **Approach 1**.

Approach 1: This approach involves four training parameters: X, W, b_u and b_f. Firstly, the model "partially" trains these parameters, meaning only X and W are randomly initialized, while b_u and b_f are initialized using pre-computed values from 4.2. I believe the pre-computed biases could help the model converge faster. I refer to this as approach **1.A**. Secondly, the model "fully" trains these four parameters, meaning they are randomly initialized; I refer to this as approach **1.B**.

Approach 2: This approach involves three training parameters: X, W, and B. Firstly, the model "partially" trains these parameters, meaning only X and W are randomly initialized, while B is initialized using pre-computed values from 4.2. I believe that the pre-computed biases could help the model converge faster. I refer to this as approach **2.A**. Secondly, the model "fully" trains these three parameters, meaning they are randomly initialized; I refer to this as approach **2.B**.

Remember that B is the biais matrix where $b^{(i,j)}$ is the combination of $b_m^{(i)}$ and $b_u^{(j)}$. Therefore, the loss function is calculated as follows:

$$\mathcal{L}(\mathbf{w}^{(j)}, \mathbf{x}^{(i)}, \mathbf{b}^{(i,j)}) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(\mathbf{x}^{(i)} \cdot \mathbf{w}^{(j)} + \mathbf{b}^{(i,j)} - y_{\text{norm}}(i,j) \right)^2 + \underbrace{\frac{\lambda}{2} \left(||\mathbf{w}^{(j)}||^2 + ||\mathbf{x}^{(i)}||^2 + (\mathbf{b}^{(i,j)})^2 \right)}_{\text{regularization}} \tag{9}$$

The gradient descent updates are now defined by Equations 3 and 4, with an additional adjustment equation for the parameter $\mathbf{b}^{(i,j)}$ calculated as follows:

$$b^{(i,j)} = b^{(i,j)} - \alpha \cdot \frac{\partial \mathcal{L}}{\partial b^{(i,j)}}$$

Where:
- $\mathcal{L}$ is the loss Equation 9

4.5 Predicted Ratings

Predicted ratings matrix for the initial model and model adjustment is simply the matrix $\hat{Y}$,

$$\textbf{Predicted Ratings Matrix} = \widehat{Y}$$

which corresponds to each model. However, this is not the same for the further adjustment model.

For the model with additional adjustments 4.4.1, after obtaining the matrix $\widehat{Y}$, we need to return to the original form of ratings by adding back the average rating for each film, which was subtracted for normalization purposes 4.4.1. There is a minor issue with new users who have not provided any ratings. As a result, these users are not included in the normalization process, and their predicted ratings for movie i are replaced with the average rating of movie i. Consequently, the predicted ratings matrix is calculated as follows:

$$\textbf{Predicted Ratings Matrix} = \widehat{Y} + \overline{Y_i}$$

Where:

- $\widehat{Y}$: represents the estimated rating matrix of Y. The method for calculating $\widehat{Y}$ depends on the specific considerations taken into account. This matrix contains the predicted ratings $\hat{y}(i,j)$ for movie i given by user j.
- $\overline{Y_i}$ represents the matrix of average movie ratings, where $\overline{y_i}$ denotes the average rating for movie i.

4.6 Principal Component Analysis - PCA

The objective is not to use the entire dataset for analysis but to focus on its most important components - its principal components. By reducing the size of a large matrix, we face a trade-off: the model may lose a small degree of accuracy, but the computation time is significantly reduced, allowing us to process larger datasets more efficiently. Moreover, after reduction, the matrix becomes denser because the essential information is concentrated in a few compact vectors rather than being spread out sparsely with a lot of 0 values in the matrix.

In this project, the matrix has a size of 9724 movies x 611 users. Here, I treat the 611 users as features for each movie. These features represent the "preferences" or "tendencies" of how the movie is rated. So, the movie feature matrix is represented by a 611 x 611 matrix.

Step1: Standardize the data to have an equivalent value scale. Standardization is done by dividing a feature data value by the average value of all feature data sothat the data will be centered at 0.

Step2: Calculate the covariance matrix to determine the correlation of each feature in the data

Step3: Calculate the eigenvector and eigenvalues from the covariance matrix, then sort from the largest value.

Step 4: The goal is to select the principal components by considering the eigenvectors corresponding to the largest or most significant eigenvalues.

In this project, the initial model and the main model adjustments are designed to fit closely to the rating matrix Y, while the additional adjustments in Approach 1 and Approach 2 aim to fit the model to the normalized rating matrix Y_{norm}. Thus, PCA needs to be applied to three matrices: Y, Y_{norm}, and B.

It is important to note that PCA cannot be applied to the additional adjustment in Approach 1.A. This is because, in this approach, the biases for users $b_u^{(j)}$ and for movies $b_m^{(i)}$ are precomputed in order to accelerate convergence. The issue arises from the fact that these two bias matrices are of dimensions 1x611 and 9724x1, respectively, which makes them incompatible for PCA.

The principal component matrix $P1$ for the rating matrix Y has dimensions $611 \times t$, where 611 represents the number of features (users) and t is the number of principal components. In contrast, the principal component matrices $P2$ and $P3$ applied to the normalized rating matrix Y_{norm} and to the matrix B (used in the additional adjustments), respectively, each have dimensions $611 \times v$, where v denotes the number of principal components.

Step 5: Compute the new dimension matrices:

$$\mathbf{New\,Y} = Y \cdot P1, \quad \mathbf{New\,Y_{norm}} = Y_{norm} \cdot P2, \quad \mathbf{New\,B} = B \cdot P3.$$

Here, $\mathbf{New\,Y}$ has dimensions $9724 \times t$, while both $\mathbf{New\,Y_{norm}}$ and $\mathbf{New\,B}$ have dimensions $9724 \times v$.

Step 6: For the initial model and its main adjustment, I fit the model to minimize the difference:

$$\widehat{\mathbf{New}Y} - \mathbf{New\,Y}.$$

Similarly, for the additional adjustments (following Approach 1 and Approach 2), I fit the model to minimize:

$$\widehat{\mathbf{New}Y_{\mathbf{norm}}} - \mathbf{New\,Y_{norm}}.$$

Step 7: Project the approximated rating matrices, obtained from dimensionality reduction, back to the original dimensional space (Reconstruction):

$$\hat{Y}_{init} = \widehat{\mathbf{New}Y} \cdot P1^T, \quad \hat{Y}_{add} = \widehat{\mathbf{New}Y}_{norm} \cdot P2^T.$$

Although $\hat{Y}_{init}$ and $\hat{Y}_{add}$ share the same dimensions, they differ in notation because $\hat{Y}_{add}$ is not the final prediction matrix.

Step 8: The prediction matrix for the initial model (and its main adjustments) is given by $\hat{Y}_{init}$,

$$\textbf{Predicted Ratings Matrix} = \hat{Y}_{init},$$

while the final prediction for the Model with Additional Adjustments is

$$\textbf{Predicted Ratings Matrix} = \hat{Y}_{add} + \overline{Y_i},$$

where $\overline{Y_i}$ represents the matrix of average movie ratings 4.5.

5 Results

5.1 Evaluation Metrics

Evaluation metrics for recommendation on this project contain :

1. RMSE [4]: this indicator is particularly useful because it provides a single value to indicate how well the model's predictions match the observed data. The lower the RMSE, the closer the predictions are to the true values, and the better the model performs. However, RMSE is sensitive to outliers due to the squaring of errors, meaning that large errors disproportionately affect the final score.

$$RMSE = \sqrt{\frac{1}{S} \sum_{(i,j) \in S} (\hat{y}(i,j) - y(i,j))^2} \tag{10}$$

Where:

- S is the set of all user-item pairs (i, j) with actual ratings.

We can also calculate the Coefficient of Variation of RMSE to evaluate the variation of the data relative to the mean of the ratings.

$$CV(RMSE) = \frac{\text{RMSE}}{\mu} \cdot 100 \tag{11}$$

Where:

- μ represents the average of all assigned ratings.

2. MAE [4]: this indicator provides an intuitive interpretation of the model's error, representing the average amount the predictions differ from the true values in the same unit as the data. A smaller MAE indicates better predictive accuracy. Since MAE does not square the errors, large deviations (outliers) do not have as much impact as they do on RMSE. This makes MAE more robust in the presence of extreme values. MAE is a simpler and more robust metric, focusing on the absolute magnitude of errors. It treats all deviations equally.

$$MAE = \frac{1}{S} \sum_{(i,j) \in S} |\hat{y}(i,j) - y(i,j)| \tag{12}$$

The Coefficient of Variation for MAE is also considered to measure the relative variation of the absolute errors with respect to the mean of the observed values, providing a normalized measure of the prediction errors consistency.

$$CV(MAE) = \frac{\text{MAE}}{\mu} \cdot 100 \tag{13}$$

3. Pearson product-moment Correlation Coefficient [4, 5]: this indicator is used to determine how strongly two variables are correlated. It is sensitive to outliers, as extreme values can significantly distort the correlation.

$$PCC = \frac{\sum_{(i,j)\in S} \left(\hat{y}(i,j) - \overline{\hat{y}}\right)\left(y(i,j) - \bar{y}\right)}{\sqrt{\sum_{(i,j)\in S} \left(\hat{y}(i,j) - \overline{\hat{y}}\right)^2} \sqrt{\sum_{(i,j)\in S} \left(y(i,j) - \bar{y}\right)^2}} \tag{14}$$

Where:

- $\overline{\hat{y}}$ is the mean of all predicted ratings $\hat{y}(i,j)$, corresponding to actual ratings $y(i,j)$ that are non-zero.
- $\bar{y}$ is the mean of all actual ratings $y(i,j)$.

5.2 RMSE and MAE

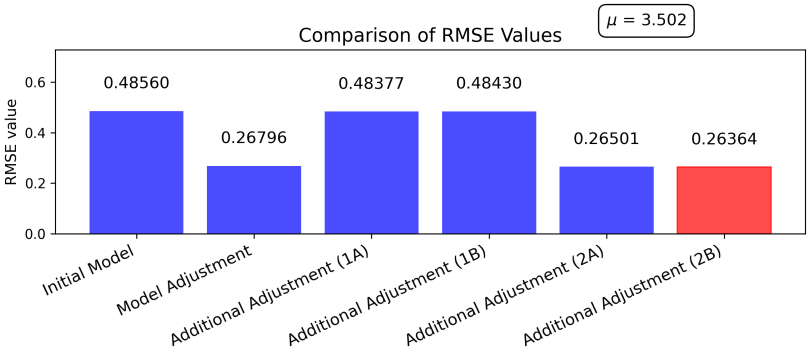

Fig. 1: Comparison of RMSE Values

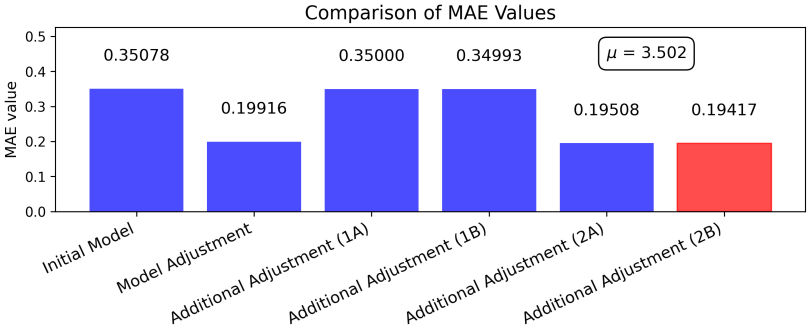

Fig. 2: Comparison of MAE Values

Obviously in Fig. 1 and Fig. 2 , three models – the main model adjustment and the two scenarios of the additional adjustment model in Approach 2 – yield the lowest RMSE and MAE. These three models train only three parameters, namely, X, W, and B, compared to the

four parameters used in the other three models. Additionally, within each model group—those with three parameters and those with four parameters—the model that incorporates the fixed term μ (i.e., the main model adjustment in the three-parameter group and the initial model in the four-parameter group) exhibits the highest RMSE and MAE in its group. As expected, the models with further adjustments have proven to be more effective. We can also conclude that Approach **2.B** of the additional adjustment model has the best performance, with fewer outliers in its predictions and the lowest magnitude of error. With an MAE of just 0.19, the predictions are very close to the actual ratings.

For instance, if the model predicts a rating of 4, the actual rating is generally within the range of approximately 3.8 to 4.2. This small error margin indicates high precision, meaning the model consistently produces predictions that are nearly identical to the true values. Additionally, by training only three parameters, the model is simpler and less prone to overfitting, which further contributes to its robust performance. Overall, these results suggest that the simpler design of these models especially Approach **2.B** is highly effective in capturing the underlying patterns in the data while minimizing prediction errors.

Moreover, it's important to note that in both Approach 1 and Approach 2, there are two categories of parameters: fully trained parameters (Approach **1.B**, **2.B**) and partially trained parameters (Approach **1.A**, **2.A**). The category with fully trained parameters is more effective than the partially trained one, although the performance difference is not significant. We do not have enough evidence to conclude that partially trained parameters help the model find the optimal direction and converge more quickly.

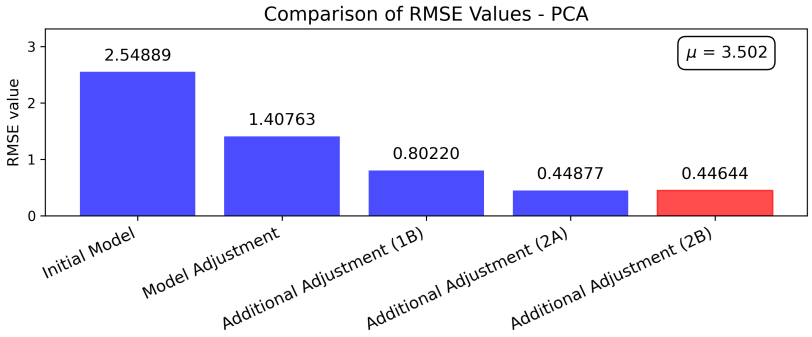

Fig. 3: Comparison of RMSE Values - PCA

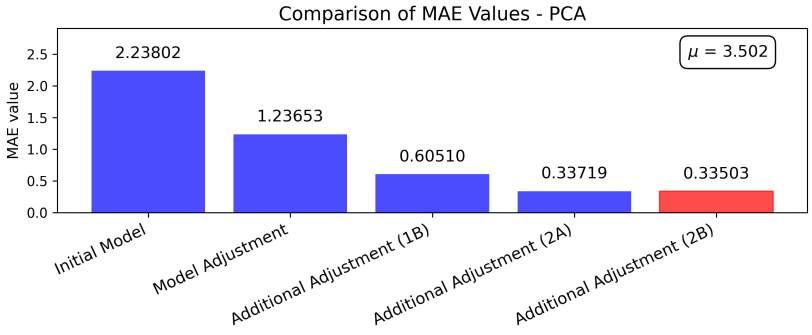

Fig. 4: Comparison of MAE Values - PCA

When I apply PCA to the model, the RMSE and MAE increase as shown in the two barplots above Fig. 3, Fig. 4, which is expected.

Additionally, we can observe that the initial model and its adjusted version have significantly higher RMSE and MAE than the other models. One possible explanation for this is that these two models, without PCA, include the fixed term μ (the average rating of all assigned ratings). However, when applying PCA, the fixed term μ is no longer explicitly considered, because in the reduced space, all information is compressed, making it unclear which ratings are the actual assigned ones. This leads to increased difficulty in optimizing the model.

Loss of Approach 2.A in PCA space	**Loss of Approach 2.B in PCA space**
185384.482	17657.133

Tab. 1: Comparison of Loss between Approach 2.A and 2.B (PCA)

Another observation when applying PCA to the two most performant models (2.A and 2.B) is that the loss value for Approach 2.A is significantly higher than that of Approach 2.B (Tab. 1), which is considered noticeably less accurate. When I reconstruct the original dimensional space, the RMSE and MAE of Approach 2.B remain lower than those of Approach 2.A, although the gap is small and not as large as before returning to the original space. This phenomenon can be explained by the fact that Model 2.A learns the normalized rating matrix $\mathbf{New}Y_{\text{norm}}$ "less well"in the PCA space, but once reconstructed in the original space, it performs better.

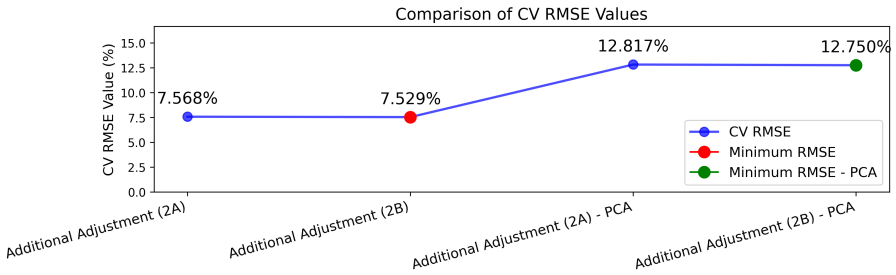

Fig. 5: Comparison of CV(RMSE) Values

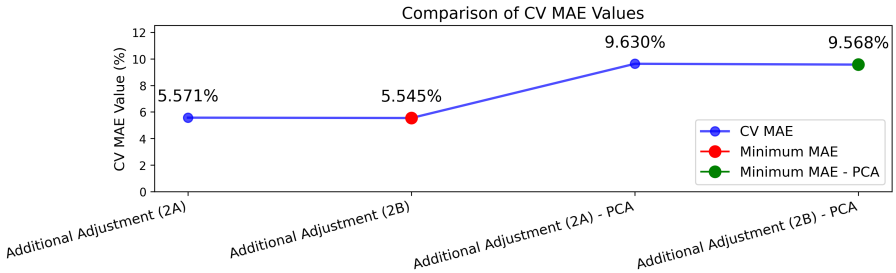

Fig. 6: Comparison of CV(MAE) Values

According to Fig. 5 and Fig. 6, we will take a look at 2 other indicators: CV(RMSE) and CV(MAE). Obviously, it was predicted that the errors of the non-PCA models will be smaller. However, it is noteworthy that the CV(RMSE) and CV(MAE) values of the non-PCA models are lower by 5% compared to the values of the models with PCA. his 5% difference can be explained by the fact that, in the earlier part of this project, I only selected the eigenvectors that explain 95% of the total variance to construct the principal component matrix.

We can clearly confirm that the model without PCA has stable and less fluctuating error.

5.3 Pearson product-moment Correlation Coefficient - PCC

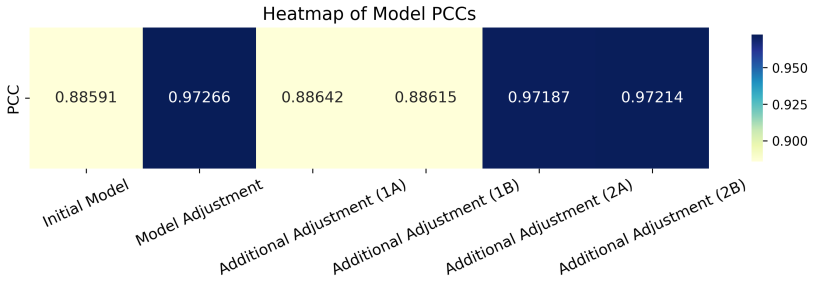

Fig. 7: Heatmap of model PCCs

The heatmap (Fig. 7) shows that the main model adjustment and the additional adjustment in Approach 2 (3-parameters group) both achieve very high PCC values (around 0.97), which means they have a strong correlation with the actual ratings. In contrast, the initial model and the additional adjustments in Approach 1 (4-parameters group) yield lower PCC values (around 0.88), indicating less effective performance. Overall, this suggests that Approach 2 is more effective at improving prediction accuracy.

5.4 Computation Time

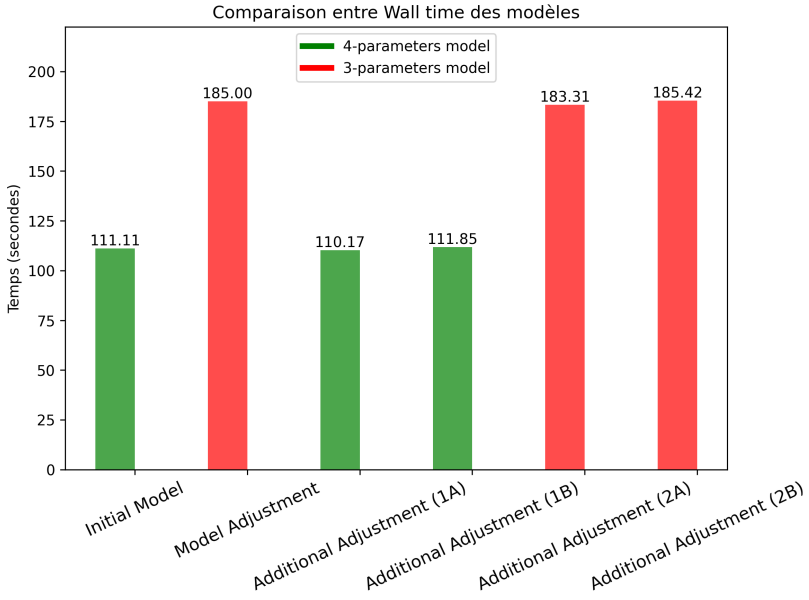

Fig. 8: Comparison of the Wall times of the models

Obviously, we can observe that the four-parameter models have a faster computation time compared to the three-parameter models. One possible explanation for this is that the three-parameter models train an entire bias matrix B, whereas the four-parameter models only train 2 biases vectors, which is a simpler structure. However, despite the faster computation time of the four-parameter models, the error rate of the three-parameter models is better, as shown in the results above. This suggests that while the four-parameter models are more efficient in terms of computation, the three-parameter models might provide more accurate predictions.
When applying PCA, the computation time decreases as expected, since PCA reduces the dimensionality of the data, leading to less computational complexity.

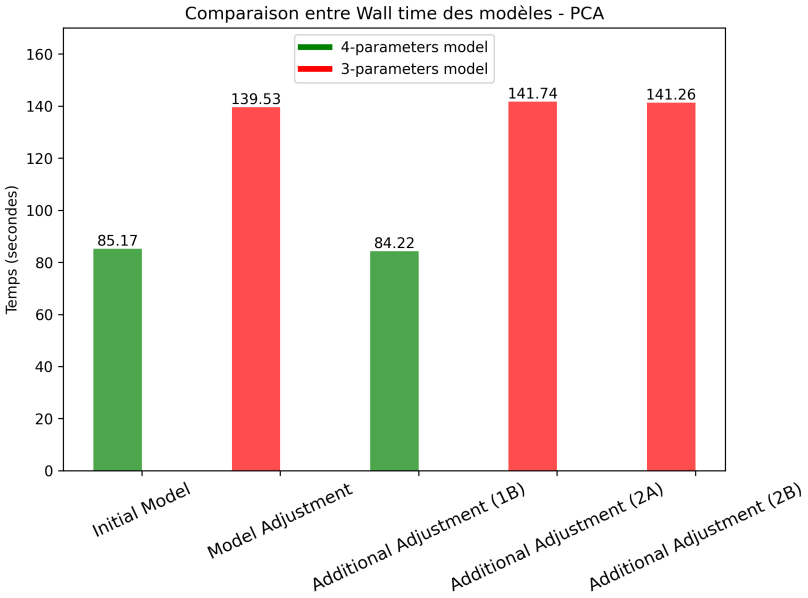

Fig. 9: Comparison of the Wall times of the models - PCA

Nevertheless, when considering this metric, it is important to view it from a relative perspective, as it is highly dependent on the available resources and the specific circumstances of each case. For instance, factors such as system load, hardware limitations (e.g., restricted RAM or processing power), and external processes can significantly affect the wall time.

6 Discussion

1. The problem of splitting data for training : the model learns latent features from user-item interaction data. However, if certain users or items in the probe/test set did not appear in the training set, the model has no prior knowledge (i.e., latent factors) about them. This lack of prior information leads to poor performance in predicting preferences, which is a classic manifestation of the cold start problem. For example, the data set comprises 10 users and 50 movies. Splitting it into a training set (7 users, 35 movies) and a probe set (3 users, 15 movies) introduces significant challenges. The model trained on user-item pairs in the training set cannot predict interactions for user-item pairs in the probe set due to the absence of prior exposure.
Moreover, another reason why we cannot use the model trained in the training set to predict the probe set is because the formula to find the approximate matrix of Y: $\hat{Y} = X \cdot W + B$, where W, X, and B are randomized to minimize the loss function.

2. A key issue arises when applying PCA: when we reduce the dimensionality of the ratings matrix and fit the model in the lower-dimensional space, the actual ratings no longer exist in their original form. Instead, the reduced-dimensional ratings matrix serves as an approximation of the original matrix. As a result, when computing the loss function in the reduced space, we unintentionally fit the model using all ratings, including the non-assigned ones. After reconstructing the original space, this may lead to an increase in both RMSE and MAE.
Therefore, using PCA to reduce the dimensionality of the matrices in this case is not ideal, as it results in the loss of information about actual ratings in the PCA space. Instead, a more favorable approach is to apply matrix factorization models directly, ensuring that the loss function is computed only on observed ratings. According to Koren, Bell, & Volinsky (2009) [2], matrix factorization methods in recommender systems are widely used and effective in practice

because they explicitly optimize the model based only on actual ratings.

7 Conclusion

To conclude, this report presents a personal side project aimed at exploring how recommender systems work with Matrix Factorization. This is not a formal research study but rather a set of observations and adjustments to existing models in an attempt to enhance their performance.

Among all the models evaluated in this project, Approach 2.B achieved the best results, with the lowest RMSE and MAE, as well as the highest PCC, indicating a strong correlation between the predicted and actual ratings. However, although Approach 2.A requires an additional step to pre-calculate the biases, its improvement over Approach 2.B remains marginal. In fact, despite this extra step, Approach 2.A does not outperform the model using random biases.

Furthermore, I attempted to apply PCA to optimize the loss function in a reduced-dimensional space to accelerate convergence, then reconstruct it back to the original space. However, the loss of information regarding actual ratings during dimensionality reduction made it challenging for the model to converge properly and led to optimization in an incorrect direction.

Therefore, it is preferable to apply matrix factorization directly in the original dimension, where the actual ratings are explicitly preserved. However, as the dataset grows larger and sparser, computational cost, time, and prediction quality become significant challenges. Some direct matrix factorization methods, such as SVD, can be applied. Alternatively, dimensionality reduction techniques like t-SNE or Autoencoders may help mitigate these computational constraints while preserving essential information.

References

[1] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web: methods and strategies of web personalization*, pages 291–324. Springer, 2007.

[2] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[3] L. Lü, M. Medo, et al. Recommender systems. *Physics Reports*, 519(1):20–21, 2012.

[4] L. Lü, M. Medo, et al. Recommender systems. *Physics Reports*, 519(1):10–11, 2012.

[5] J.L. Rodgers and W.A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 42:59–66, 1988.