IR Assignment 2
Design Document

Abhinav Kumar 2015B5A70674H
Yashdeep Thorat 2015B5A70675H
Shivam Bhagat 2015B5A70460H

Table for MovieLens 1M dataset:

Recommender System Technique	Root Mean Square Error (RMSE) (Train/Validation)		Precision on top K	Spearman Rank Correlation	Time taken for prediction (1 single prediction)
Collaborative	0.87478456		0.89342251	0.99999995411	$3.8078 \times 10^{-4}\,s$
Collaborative along with Baseline approach	0.83908953		0.89154682	0.99999995778	$4.3491 \times 10^{-4}\,s$
SVD	0.762832	1.037559	0.92393705	0.999999935	$8.6447 \times 10^{-4}\,s$
SVD with 90% retained energy	0.733616	1.037559	0.92458340	0.999999935	$8.5236 \times 10^{-4}\,s$
CUR	0.302670	1.363815	0.90190818	0.999999887	$1.24674 \times 10^{-5}\,s$
CUR with 90% retained energy	0.209455	1.037559	0.92458340	0.999999934	$1.27809 \times 10^{-5}\,s$

Implementation details:

Language used: **Python**

Packages used:

- numpy
- math
- scipy
- pandas
- time

1. Collaborative Filtering

Brief Description:

Item-Item collaborative filtering has been used. Predicted rating is calculated as weighted average of user ratings w.r.t item-item similarity scores. Centered cosine score (Pearson correlation) is used for computing similarity scores.

The formula for rating:

$$r_{xi} = \frac{\sum_{j \varepsilon N(i;x)} s_{ij} . r_{xj}}{\sum_{j \varepsilon N(i;x)} s_{ij}}$$

s_{ij}: similarity of items i and j
r_{xj}: rating of user u on item j
N(i;x): set items rated by x similar to i

The formula for the similarity score:

$$s_{ij} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Pros of collaborative filtering:

- Works for any kind of item: No feature selection needed.

Cons of collaborative filtering:

- The utility matrix is sparse: Hard to find users that have rated the same items.
- Need enough users in the system to find a match.
- Popularity bias: Cannot recommend items to someone with unique taste. Tends to recommend popular items.
- Cannot recommend an item that has not been previously rated.

Assumptions & some Parameters:

- Dataset name: MovieLens 1M
- Number of users: 6040
- Number of movies: 3952
- Neighborhood size taken: 15
- Rating range: 1 to 5
- Threshold rating for relevant movies: 3
- Dimension of utility matrix (userID vs movieID): (6040 x 3952)
- Dataset is split into rating_matrix (90%) & validation_matrix(10%)

Handling strict & generous raters and (missing ratings as 0) problem:

This is taken care by centered cosine similarity score. While calculating the cosine score, mean of non-zero ratings are subtracted from each non-zero rating for that movie. This ensures that all ratings are centered around 0 and the rating 0 is not taken as “negative” but as average for that movie.
Also, the denominator of the formula for score contains standard deviation of the ratings of the movie, this normalizes the score and handles strict & generous raters.

Running time details:

We have precomputed and cached (stored) a matrix which contains similarity scores of all possible (userID, movieID) pairs. Due to this, the overall time has dropped significantly.

<u>Before caching:</u>
Running time for all predictions: 60.5 minutes
Running time for all predictions + error calculation: 60.5 minutes

<u>After caching:</u>
Running time for all predictions: 3.7756 seconds
Running time for all predictions + error calculation: 3.8634 seconds

---

2. Collaborative Filtering + Baseline

Brief Description:

In addition to the above approach, a global baseline estimate is also added to the local estimate to predict the rating. To avoid counting twice, the same is subtracted from each rating while computing the local estimate.

Formula for rating:

$$r_{xi} = b_{xi} + \frac{\sum_{j \varepsilon N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \varepsilon N(i;x)} s_{ij}} \qquad b_{xi} = \mu + b_x + b_i$$

μ = overall mean movie rating
b_x = rating deviation of user x
b_i = rating deviation of movie i

Pros of collaborative filtering + baseline:

- Works for any kind of item: No feature selection needed.
- Takes into account the general behavior of the target user.
- Gives less RMSE than normal Collaborative.
- Can recommend an item that has not been previously rated.

Cons of collaborative filtering + baseline:

- Sparsity: The utility matrix is sparse: Hard to find users that have rated the same items.
- Cold Start: Need enough users in the system to find a match.

Assumptions & some Parameters:

<Same as normal Collaborative>

Handling strict & generous raters and (missing ratings as 0) problem:

<Same as normal Collaborative>

Running time details:

We have precomputed and cached (stored) a matrix which contains similarity scores of all possible (userID, movieID) pairs. Due to this, the overall time has dropped significantly.

<u>Before caching:</u>
Running time for all predictions: 61 minutes
Running time for all predictions + error calculation: 61 minutes

After caching:
Running time for all predictions: 4.3759 seconds
Running time for all predictions + error calculation: 4.4218 seconds

Comparison of normal Collaborative and Collaborative + Baseline:

- The RMSE error of Collaborative + baseline is LOWER than that of the normal Collaborative method.(*Reason*: The former takes into account global baseline effect too)
- Spearman correlation is almost the same for both methods. (very close to 1) (*Reason*: both actual & predicted ratings behave like similar monotonic functions).
- Precision at top K is HIGHER for normal Collaborative than that of Collaborative + baseline method. *(Reason*: Unlike RMSE, this method doesn't penalize lower ratings)
- Time taken for prediction & error calculation is HIGHER for Collaborative + baseline than that of the normal Collaborative method. (*Reason*: More time needed for baseline calculation)

---

3. Singular Valued Decomposition

Brief Description:
Singular Value Decomposition (SVD) is a method of dimensionality reduction. Projecting the data into lower dimension helps us to see some similarity and pattern in data, by finding the global pattern and correlation between them.

The decomposition of the actual matrix is given by following formulae:

$$A_{n\times m} = U_{n\times r}\, \Sigma_{r\times r}\, V^{T}_{m\times r}$$

where;

$$A^T A = V\, \Sigma^2\, V^T$$
$$AA^T = U\, \Sigma^2\, U^T$$
$$UU^T = I$$
$$V V^T = I$$

Here the main idea is to project the data into concept space which kind of represent the **"Eigen/ideal movies" and "Eigen/ideal user"** which holds the information of common pattern found in user and movies. Thus this method **makes use of both user-user relationships along with the item-item relationship for projection.**

1. Mode1: Using 100% Energy
In this mode, we kept all the eigenvalues of the decomposition in the Sigma matrix.

Pros of This Mode:

- This decomposition thus is able to reconstruct the original rating/utility matrix with the error comparable to machine precision error.

Cons of This Mode:

- Since we preserve all the eigenvalue, thus the highest variance in the projection to the concept space, the prediction was susceptible to noise.

2. Mode2: Using 90% Energy/Variance preservation
In this mode, we have removed some of the lower eigenvalues such that we preserve 90% of the variance in the projected space.

Pros of this Mode:

- This removal of lower energy/variance helps to smoothen the prediction by removing the noise in the data which could be due to missing values.

Cons of this Mode:

- This mode loses some information since we remove some of the eigenvalues.
- Thus the reconstruction error increases in this case.

Assumptions and Other parameters:

1. **Training and Validation Split:** We have done a 90%-10% training and validation split on the actual rating matrix to test our prediction on unseen data. 90 percent of the data was only kept in the rating matrix while decomposing and we tested the prediction on the rest of the 10% data after reconstruction.
2. **Unrated movies:** We have taken into consideration of unrated movies by subtracting the mean rating of all users in their respective rating. Thus the unrated moves are not seen as negatives now.

Handling strict & generous raters

We have taken into consideration the strict and generous raters by normalizing the rating matrix by dividing the variance of rating of each user in their corresponding rating.

Running time details:

We have precomputed the SVD decomposition and saved them beforehand so that we don't compute the decomposition at the runtime.
Before caching:
Running time for all predictions in 100% energy: 7.5 minutes
Running time for all predictions in 90% energy: 7.2 minutes

After caching:
Running time for all predictions in 100% energy: 8.6472 seconds
Running time for all predictions in 90% energy: 8.5261 seconds

4. <u>CUR Decomposition:</u>

The problem encountered in SVD decomposition is that U and V matrix is larger in size and not sparse and Sigma matrix being smaller in size, is very sparse. So, this comes as a challenge in storing the decomposition.
So, we use now the CUR decomposition where the C and R matrix is large but sparse whereas U matrix in small but dense. The decomposition looks like:

$$A = C\ U\ R$$

Here C, the matrix is the made by randomly sampling the columns of the actual data matrix and further normalizing it by its probability. And we find U matrix by finding the pseudo-inverse of the SVD decomposition of the intersection matrix of C and R.

Discussion about Modes:
The interpretation of two modes here 100% energy and 90% energy have the same intuition in this context also. (Same as SVD)

Assumptions and Other parameters:

1. **The number of Rows and Columns:** We experimented with the number of rows and columns to be sampled from the original matrix and found that keeping **90% of rows and columns give us good tradeoff between reconstruction and prediction errors.**

Pros of this model:

1. Now since the C and R matrix contains, the rows and columns of the original matrix, they are sparse and less expensive on memory consumption.
2. Also, now the concept space is more interpretable since we can think of the person in terms of kind of movies he likes or dislikes.
3. Similarly, the movies can be thought in terms of the kind of person who watches them.

Cons of this model:

1. Since we are randomly choosing the rows and columns, we are losing some information about our actual rating matrix, then we will tend to have higher reconstruction error.

Handling strict & generous raters

The strict and generous raters were handled in the same manner as done in SVD.

Running time details:

We have precomputed the CUR decomposition and saved them beforehand so that we don't compute the decomposition at the runtime.

Before caching:

Running time for all predictions in 100% energy: 10.5 minutes

Running time for all predictions in 90% energy: 10.3 minutes

After caching:

Running time for all predictions in 100% energy: 0.1247 seconds

Running time for all predictions in 90% energy: 0.1278 seconds

Comparison of Collaborative and SVD & CUR approach:

- The RMSE error of Collaborative approach is LOWER than that of the SVD/CUR method. (*Reason*: The former takes into account global baseline effect too)
- Spearman correlation is almost the same for both methods. (very close to 1) (*Reason*: both actual & predicted ratings behave like similar monotonic functions).
- Precision at top K is HIGHER for SVD/CUR than that of Collaborative approach. *(Reason*: Unlike RMSE, this method doesn't penalize lower ratings)
- Time taken for prediction & error calculation is HIGHER for Collaborative approach and SVD than that of the CUR approach. (*Reason*: More time needed for baseline calculation)