Design of Memory/Model Based Collaborative Filtering Algorithms for Recommender Systems

Submitted in partial fulfilment of the requirements of the degree of
Bachelor of Technology

by

Radhika Sunil Jain (177249)
Srikavya (177257)
Ajmeera Sreekanth (177202)

Supervisor
Dr. Sanjaya Kumar Panda
Assistant Professor
NIT Warangal

Department of Computer Science and Engineering
NATIONAL INSTITUTE OF TECHNOLOGY WARANGAL
2020-21

APPROVAL SHEET

The project work entitled **"Design of Memory/Model Based Collaborative Filtering Algorithms for Recommender Systems"** by **Radhika Sunil Jain (177249), Srikavya (177257) and Ajmeera Sreekanth (177202)** is approved for the degree of Bachelor of Technology in Computer Science and Engineering.

Examiners

____

____

____

Supervisor

[illegible]

____

Dr. Sanjaya Kumar Panda
Assistant Professor

Chairman

____

Prof. P. Radha Krishna
Department of Computer Science and Engineering

Date:________________________

Place:________________________

DECLARATION

We declare that this written submission represents our ideas in our own words and where other's ideas or words have been included we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from where proper permission has not been taken when needed.

[illegible]

Radhika Sunil Jain - 177249

Srikavya

Srikavya - 177257

Sreekanth

Ajmeera Sreekanth - 177202
Date: 9th May, 2021

CERTIFICATE

This is to certify that the project work entitled **"Design of Memory/Model Based Collaborative Filtering Algorithms for Recommender Systems"** is a bonafide record of work carried out by **Radhika Sunil Jain (177249), Srikavya (177257) and Ajmeera Sreekanth (177202)**, submitted to the faculty of "Computer Science and Engineering Department", in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in "Computer Science and Engineering" at National Institute of Technology, Warangal during the academic year 2020-21.

[illegible]

Dr. Sanjaya Kumar Panda
Assistant Professor
Supervisor
Department of CSE
NIT Warangal

Prof. P. Radha Krishna
Head of the Department
Department of CSE
NIT Warangal

Abstract

Recommender Systems are used to recommend new items to users. Collaborative Filtering Approaches suffer from cold start problem in which it is difficult to recommend items to new users or for new items that do not have any ratings. Due to this the quality of ratings is reduced. Sparsity of the rating matrix is also a major concern that makes it difficult to find similar items. A variety of approaches have been proposed based on asking the users to manually rate some items. These approaches are not feasible.

This project aims to design an enhanced memory and model based algorithms to solve the cold start problem in Collaborative Filtering Algorithms and provide better recommendations. Our approach is involved in populating the rating matrix to provide some ratings to new users to solve the cold start problem at the same time removing the sparsity of the matrix. We have compared our results with a recent probabilistic matrix factorization based model.

List of Tables

Contents

Chapter 1

Introduction

1.1 Recommender Systems

A recommender system is a tool that is used to recommend items such as movies [1], music [2], travel [3, 4], social media [5, 6], books, web series, etc. to a user. A recommender system tries to find items that a user would be interested in. Recommender systems are useful for both the business and users by selecting a few items from a large choice of items that a user is likely to be interested in. This allows the user to quickly find items that he/she will be interested in and is beneficial for business by having the user consume more items. It can serve as a good marketing strategy. Recommender systems are used in various domains like e-commerce, movies, music, books to recommend items to users. Items are recommended to users using:

Collaborative filtering - Collaborative filtering [7] is a user/item based method. It is based on the assumption that users that have similar tastes will likely have similar tastes in the future as well. It finds users/items that have similar tastes as a user/item and recommends items that are rated highly by the similar users/items. Similarity metrics are used to find similar users and items are recommended based on these similar users. These recommender systems suffer from the cold start problem.

Content filtering - Content based filtering approach uses the knowledge known about a user to find items that have similar features to the items liked by a user. The knowledge about a user can be based on explicit information (previous ratings) or implicit information. It recommends items that are similar to the items that user has rated highly. This method does not consider the similarity between users and considers each user separately. Content based filtering is scalable and does not suffer from the cold start problem but is limited by the knowledge that the system has about a user. Domain knowledge about the features of items is required.

Collaborative filtering-based [8, 9, 10] recommender systems can be classified as:

• Memory-Based – It uses the K Nearest Neighbors (KNN) Algorithm to predict ratings using different similarity measures.

• Model-Based – It predicts ratings using a model. Matrix factorization approach using a model of latent features of users and items is a model-based approach.

1.2 K Nearest Neighbors (KNN) Algorithm

K nearest neighbors (KNN) algorithm is a type of supervised Machine Learning algorithm which can be used in classification and regression problems. It is a memory based algorithm. This K nearest neighbors (KNN) algorithm uses the similarity between users to find users that are similar to the user in interest based on the ratings given to items. The algorithm finds the top k nearest neighbours based on the chosen similarity metric. The determination of parameter k is a NP-hard problem. Similarity [11] between users u, v can be calculated using Mean Square Difference and cosine rule.

The KNN is a lazy learning approach that defers the learning process till the query (in this case recommendations for user) arrives. The KNN Algorithm is simple, easy to understand and gives good results but has sacalability issues for large datasets and suffers from the cold start problem. Finding a suitable similarity metric [12, 13] is key in this approach.

1.3 Matrix Factorization

Matrix factorization is a collaborative filtering algorithm used to recommend items to users. It is a model based algorithm based on dimensionality reduction. It factors the rating matrix into two feature vectors, one for users and one for items each containing k features. The resulting feature vectors are smaller than the given input rating matrix. The predicted rating matrix is calculated using these feature vectors. Matrix factorization has many variants - Probabilistic, Non-negative, etc.

The latent factors retain the dependencies and properties of the rating matrix. Bias can also be added to the predicted ratings to normalize different users that rate items differently. The Matrix Factorization Algorithm is simple, overcomes the scalability issue, is space efficient and gives better results than the KNN Algorithm but it suffers from the cold start problem.

1.4 Motivation

Collaborative recommender systems suffer from the cold start problem which arises due to new users or items that have few or no ratings. It is difficult to find similar users or items for such users and items. This is a major problem in recommender systems specially with large number of users. There are two categories of cold start problem - the New User problem that arises to due new users that have not rated any item, and the New Item problem that arises due to a new item that has not been rated by any users. In either case, it is difficult to find users/items that have similar tastes.

The cold start problem leads to spare dataset that gives poor recommendations and longer training times. A variety of solutions have been proposed to solve the cold start problem. One such solution is the Active Learning approach which involves asking the new users upon registering to rate a few items and select some items that they like. This approach is not reliable as some users might refuse to rate items. Another approach to solve the cold start problem is the use Content filtering methods as they do not suffer from the cold start problem.

Sparsity of the rating matrix is also a major concern in Collaborative Filtering approaches. It is due to very few ratings by most of the users in the rating matrix. This makes it difficult to find similar items as most of the users will not have rated common items. It also increases the computation cost.

Our approach is focused on trying to populate the rating matrix to remove sparsity and the cold start problem. We have tried to find suitable ratings to assign to the new users that gives good results. We do this by finding an appropriate number of popular items among the users.

1.5 Contribution

We have made the following contributions:

- Enhanced the K Nearest Neighbours algorithm to solve the cold start problem.
- Enhanced the the probabilistic matrix factorization model proposed in [14] to solve the cold start problem.

- Analysed and compared the performance of our algorithms on various metrics.
- Our enhanced algorithms also solve the sparsity problem in rating matrices.

Chapter 2

Related Work

Ruslan Salakhutdinov and Andriy Mnih [15] have proposed a Probabilistic Matrix Factorization approach that is an improvement to the traditional matrix factorization. The proposed algorithm is scalable linearly with the increase in the dataset and performs well on the large, sparse, and very imbalanced Netflix dataset.

Jesús Bobadilla, Fernando Ortega, and Antonio Hernando [7] have proposed an algorithm that introduces the concept of singularity in calculating the similarity between two users. It considers that each rating in the dataset does not have equal weightage. Items that have been rated positively by most users but negatively by some users contribute more to the similarity between users if they both fall in the group that has rated the item negatively. If all users have rated an item similarly (positively or negatively), that item does not contribute much to the similarity. Using this weighted similarity, similar users are found and ratings are assigned.

Hernando et al. [16] have proposed an algorithm to overcome the cold start problem in recommender systems. Cold start problem arises due to new users that do not have any ratings. In this paper, a probabilistic model is given through which the recommendations of these new users is inferred. The probabilistic model is based on "uncertainty rules" of users generally liking an item and a user liking an item based on the fact that it likes another item. These relations are represented as a graph. A new user is recommended items based on these "uncertainty rules".

Mohammad-Hossein Nadimi-Shahraki and Mozhde Bahadorpour [17] have analysed the methods to solve the cold start problem - how to make recommendations for a new user in collaborative filtering approach. To solve this problem "ask-to-rate" is popular method in which the new user is assigned some ratings by asking the user some questions about his/her likes and dislikes or asking the user to rate a few items. This method has many drawbacks as the user can refuse to answer the questions.

Panda et al. [12, 13] have proposed a modification of KNN that uses similarity count, a new metric to find the similarity between users. Similarity count between two users is the number of items that both users have rated. The k similar users to find the rating for an item by a user are taken based on the users that have rated the item in interest and have the highest similarity count with the user. It overcomes insufficient neighbour problem but does not handle cold start problem.

Panda et al. [18] have proposed an algorithm that normalizes the user ratings in 1-5 range and identifies different category of users - stringent, lenient and normal. The ratings are assigned based on normalization count. This method tests the recommendations on new metrics - total, average, minimum and maximum number of recommended items to a user.

Ortega, Hernando, Bobadilla, and Kang [19] have proposed a Group Recommender System model, in which matrix factorization is used to find group latent factors. The proposed algorithm has three methods: After Factorization (AF), Before Factorization (BF) and Weighted Before Factorization (WBF). In the After Factorization approach, the factors of groups of users are found after matrix factorization by merging the factors of the users in the group. In Before Factorization approach, a virtual user is built that represents the likes of all the users in the group. The factors are then found on the virtual users using folding-in method. In the Weighted Before Factorization approach, a different weight is given to the items rated by the virtual users. Then the BF approach is applied.

Hernando et al. [14] have proposed a modification to the matrix factorization algorithm that limits feature vector values to lie in range 0 to 1. It reduces the number of features k required thus reducing the space and computation cost. This method finds groups of users and the probability that a user like an item if it belongs to a particular group.

Antonio Hernando, Jesús Bobadilla, Fernando Ortega, Jorge Tejedor [20] have proposed a solution that introduces another metric "reliability" on a scale of 0 to 1 to recommend items to users. Every recommendation contains two values, the predicted rating of an item and the "reliability" that the rating is correct. The user determines to go ahead with the recommendation based on both values.

J. Bobadilla, F. Serradilla, A. Hernando [21] have proposed a weighted recommender system approach that give more weight to certain users. The recommendations by users that have scored higher in tests is given more weightage in an e-learning recommender system. The weightage that a user has with respect to a user is calculated using the relative "knowledge" between users. A new similarity metric "importance" is introduced that takes into account the weightage of a user.

Chapter 3

Problem Statement

3.1 Problem Model and Formulation

Given - m number of users, n number of items and a matrix M of size $m \times n$ that gives ratings of n items given by m users. Rating is given from 1-5 with 1 being the lowest rating and 5 being highest rating. A user might not have rated some items. If a user u has not rated item i then $M_{u,i}$ will be empty or zero. Given a matrix M of ratings we want to predict the ratings of items that have not been rated by the users and recommend items to the users.

The main objective is to minimize MAE, CMAE, MSE and RMSE.

- Mean Absolute Error,

$$\text{MAE} = \frac{\sum_{r_{u,i} \neq 0} |p_{u,i} - r_{u,i}|}{|r_{u,i}| r_{u,i} \neq 0|} \tag{3.1}$$

- Constrained Mean Absolute Error,

$$\text{CMAE} = \frac{\sum_{r_{u,i} \neq 0, r_{u,i} \geq 4 \vee p_{u,i} \geq 4} |p_{u,i} - r_{u,i}|}{|r_{u,i}| r_{u,i} \neq 0, r_{u,i} \geq 4 \vee p_{u,i} \geq 4|} \tag{3.2}$$

- Mean Square Error,

$$\text{MSE} = \frac{\sum_{r_{u,i} \neq 0} (p_{u,i} - r_{u,i})^2}{|r_{u,i}| r_{u,i} \neq 0|} \tag{3.3}$$

- Root Mean Square Error,

$$\text{RMSE} = \sqrt{\frac{\sum_{r_{u,i} \neq 0} (p_{u,i} - r_{u,i})^2}{|r_{u,i}| r_{u,i} \neq 0|}} \tag{3.4}$$

where,
$r_{u,i}$ is rating of item i for user u
$p_{u,i}$ is predicted rating of item i for user u

We also want to maximize F-score, Precision and Recall.

$$\text{F-score} = 2 \times \frac{\text{Precision.Recall}}{\text{Precision} + \text{Recall}} \tag{3.5}$$

$$\text{Precision} = \frac{|TP|}{|TP| + |FP|} \tag{3.6}$$

$$\text{Recall} = \frac{|TP|}{|TP| + |FN|} \tag{3.7}$$

TP=True Positive, FP=False Positive, FN=False Negative

Chapter 4

Proposed Algorithms

4.1 Existing Algorithms

4.1.1 K Nearest Neighbours Algorithm

The K Nearest Neighbors (KNN) Algorithm is a well-known memory based collaborative filtering algorithm used to solve the recommender system problem. It calculates the similarity value for each pair of users. Similarity can be calculated using Mean Square Difference or Cosine (Dot product) of ratings of two users. It then selects k users that have the highest similarity for a user u. The item rating for an unrated item i of user u is given as the average of the ratings of item i for the k nearest neighbors.

Issues with KNN are as follows:

1. Unknown Rating Problem - cannot find similarity between two users if they have not rated any common items

2. Insufficient Neighbor Problem

3. Cold Start Problem - a new user will not have any ratings

4. Scalability Problem - not scalable for large datasets

4.1.2 Matrix Factorization Algorithm

The Matrix Factorization Algorithm is a model based collaborative filtering approach. It factors the $m \times n$ matrix M of users and ratings into two feature vectors- one for users P (of size $m \times k$) and one for items Q (of size $n \times k$)

each containing k features. Using these two vectors the entire rating matrix M can be filled. Predicted rating matrix is given by $P \times Q^T$.The feature vectors are trained using gradient descent method. Feature vector values are updated using LMS update rule.

It reduces the space required for computation as the factor vectors are much smaller than the rating matrix M for large values of m and n. Dimensionality reduction using Singular Value Decomposition (SVD) handles sparseness problem of rating matrix and the scalability issue for large datasets.

Issues with Matrix Factorization are as follows:

1. Loss of information in dimensionality reduction
2. Overfitting of the factor vectors
3. SVD increases computation cost

4.2 Improved Algorithms

4.2.1 Probabilistic Matrix Factorization

In [14] A. Hernando et al. have proposed a novel collaborative filtering technique for predicting the ratings of users in recommender systems. In this technique the rating matrix is factorized into two non-negative matrices, one for users and one for items, with values in the range 0 to 1 that have an understandable probabilistic meaning. Using these factorized matrices, the predicted ratings of users are determined, the users can be divided into groups with similar tastes and the recommendations can be justified and understood.

We used the probabilistic matrix factorization model proposed by A. Hernando et al. and extended it to solve the cold start problem in sparse matrices.

In Memory based approaches the predictions follow the proposition - For a user u that is predicted by the recommender system to like the item i, there must be some other users in the dataset with similar taste as u who have rated the item i positively.

Memory based approaches give good quality of recommendations according to evaluation metrics, for a good similarity function to find the similarity between users. The drawback of these is algorithms is that they are not scalable for large datasets.

To overcome scalability issue, matrix factorization is used but it is not suitably designed for a rating matrix as input, the factorized matrices do not follow the proposition and the MAE evaluation metric indicates that matrix factorization does not give good predictions about the ratings of users.

A novel technique for factoring the input rating matrix into two non-negative matrices while preserving the advantages of the traditional matrix factorization technique is proposed:

- Similar to traditional matrix factorization, K latent factors are considered to explain the ratings that users make. Two vectors are considered, a K dimensional vector $a_u = (a_{u,1}, ..., a_{u,K})$ for each user u and a K dimensional vector $b_i = (b_{i,1}, ..., b_{i,K})$ for each item i.

- Since it is a model based approach, it is scalable for large datasets.

- It gives accurate predictions of ratings and gives good recommendations.

- The parameter K denotes the number of groups of users in the rating matrix.

- The value of the parameter K is lower in this model compared to traditional matrix factorization approach to get more accurate predictions in the model. Consequently, the dimension of vectors a_u and b_i is lower in this model compared to traditional matrix factorization, which leads to more efficient memory usage and lesser computation cost.

- The vector a_u is very sparse: only a few components of a_u have high values, while most of the components have values that are almost 0 (which are taken as 0 for practical purposes). Hence, the the predicted rating matrix $p_{u,i} = a_u b_i$ is computed quickly, since only a few components of the summation are used in calculating the predicted ratings.

Model- This algorithm finds groups of users that share the same tastes and predicts the probability of a user liking an item.

Input - The input of the algorithm is a matrix of ratings M, where,

- N: Number of users in the input matrix M.

- M: Number of items in the input matrix M.

- $r_{u,i}$: Rating given by user u to the item i. To generalize the algorithm for rating matrices that have different scales of ratings, the normalized ratings $r^*_{u,i}$ which lies between 0 to 1 is considered. This is done using min-max normalization.

Parameters - The following parameters are considered:

- $K \in N$. It is the number of latent factors the data is factored into by the algorithm. It denotes the number of groups of users in the dataset.
- $\alpha \in (0, 1)$. This parameter is an indication of having more than one group of users that have the same taste as a user. If α is almost 0, it indicates that most of the users belong to only one group. A high α value indicates that a user may belong to more than one group and more than one group has similar tastes as the user.
- $\beta > 1$. This parameter is an indication of the amount of evidence that the algorithm requires to find the probability of a user u liking the item i. If the value of β is high, it indicates that the algorithm needs more evidence to find the probability of a user u liking the item i.

Output - From the input and parameters, the algorithm outputs two matrices:

- The $N \times K$ matrix $(a_{u,k})$ for users. This matrix gives the probability of a user u belonging to a group k. Thus, by the rules of probability for each user u, we have: $\sum_{k=1}^{K} a_{u,k} = 1$
- The $K \times M$ matrix $(b_{k,i})$ for items. This matrix gives the probability of users belonging to a group k liking the item i.
- Probability that the user u likes the item i. This value $p_{u,i}$ is calculated using the following equation:

$$p_{u,i} = \sum_{k=1}^{K} a_{u,k} . b_{k,i} \tag{4.1}$$

- Expected rating of the user u for the item i, $p_{u,i}$ is transformed to obtain the non-normalized rating matrix. When the recommender system uses 1, . . . , 5 as the rating scale, this is obtained according to the following expression:

$$q_{u,i} = \begin{cases} 1 & \text{if } 0 \leq p_{u,i} < 0.2 \\ 2 & \text{if } 0.2 \leq p_{u,i} < 0.4 \\ 3 & \text{if } 0.4 \leq p_{u,i} < 0.6 \\ 4 & \text{if } 0.6 \leq p_{u,i} < 0.8 \\ 5 & \text{if } 0.8 \leq p_{u,i} \leq 1 \end{cases} \tag{4.2}$$

The predicted ratings given by a user u to an item i is simulated by the following procedure [22]:

- For each user u, a K dimensional random vector variable ϕ_u is sampled from the Dirichlet distribution:
  $\phi_u \sim Dir(\alpha, ..., \alpha)$
  The vector $(\phi_{u,1}, ..., \phi_{u,K})$ indicates the probability of a user belonging to each of the K possible groups.

- For every item i and every factor k, a random variable $\kappa_{i,k}$ is sampled from the Beta distribution (which can take values from 0 to 1):
  $\kappa_{i,k} \sim Beta(\beta, \beta)$
  The value $\kappa_{i,k}$ indicates the probability of a user that belongs group k liking the item i.

- For every item i rated by a user u such that $r_{u,i} \neq 0$, the following random variables are sampled:

  - The random variable $z_{u,i}$ from the categorical distribution (which can take values from 1, . . . , K).
    $z_{u,i} \sim Cat(\phi_u)$
    A value k in $z_{u,i}$ indicates that the user u rates the item i as if the user belongs to the group k
  - The random variable $\rho_{u,i}$ from the Binomial distribution (which can take values from 0 to R). In this case R is set to 4.
    $\rho_{u,i} \sim Bin(R, \kappa_{i,z_{u,i}})$
    The normalized rating is obtained from $\rho_{u,i}$ using the following expression:
    $r^*_{u,i} = \frac{\rho_{u,i}}{R}$

The conditional probability distribution of the unknown random variables is then determined. The real posterior distribution $p(\phi_u, \kappa_{i,k}, z_{u,i}|\rho_{u,i})$ is determined by a distribution $q(\phi_u, \kappa_{i,k}, z_{u,i})$.

$$q(\phi, \kappa_{i,k}, z_{u,i}) = \prod_{u=1}^{N} q_{\phi_u}(\phi_u) \prod_{i=1}^{M} \prod_{k=1}^{K} q_{\kappa_{i,k}}(\kappa_{i,k}) \prod_{r_{u,i} \neq 0} q_{z_{u,i}}(z_{u,i}) \tag{4.3}$$

where:

- The distribution $q_{\phi_u}(\phi_u)$, $q_{k_{i,k}}(k_{i,k})$ and $q_{z_{u,i}}(z_{u,i})$ have the following distributions:

  - The approximated conditional probability distribution of ϕ_u knowing the rating matrix M, also follows a Dirichlet distribution

$$q_{\phi_u}(\phi_u) \sim Dir(\gamma_{u,1}, ...\gamma_{u,k}) \tag{4.4}$$

where $\gamma_{u,1}, ...\gamma_{u,k}$ are parameters that need to be learned, as shown later.

- The approximated conditional probability distribution of $\kappa_{i,k}$ knowing the rating matrix M also follows a Beta distribution:
$$q_{\kappa_{i,k}}(\kappa_{i,k}) \sim Beta(\epsilon^{+}_{i,k}, \epsilon^{-}_{i,k}) \tag{4.5}$$
where $\epsilon^{+}_{i,k}, \epsilon^{-}_{i,k}$ are parameters that need to be learned, as shown later.
- The conditional probability distribution of $z_{u,i}$ knowing the rating matrix M also follows a categorical distribution:
$$q_{z_{u,i}}(z_{u,i}) \sim Cat(\lambda_{u,i,1}, ..., \lambda_{u,i,k}) \tag{4.6}$$
where $\lambda_{u,i,k}$ are parameters that need to be learned, as shown later. Since $q_{z_{u,i}}(z_{u,i})$ is sampled from a Categorical distribution, these parameters follow the condition:
$\lambda_{u,i,1} + ... + \lambda_{u,i,k} = 1$
- The parameters given above $\gamma_{u,k}, \epsilon^{+}_{i,k}, \epsilon^{-}_{i,k}, \lambda_{u,i,k}$ must follow the conditions given below:
$$\gamma_{u,k} = \alpha + \sum_{\{i|r_{u,i}\neq 0\}} \lambda_{u,i,k} \tag{4.7}$$
$$\epsilon^{+}_{i,k} = \beta + \sum_{\{u|r_{u,i}\neq 0\}} \lambda_{u,i,k}.r^{+}_{u,i} \tag{4.8}$$
$$\epsilon^{-}_{i,k} = \beta + \sum_{\{u|r_{u,i}\neq 0\}} \lambda_{u,i,k}.r^{-}_{u,i} \tag{4.9}$$
$$\lambda^{'}_{u,i,k} = exp(\psi(\gamma_{u,k}) + r^{+}_{u,i}.\psi(\epsilon^{+}_{i,k}) + r^{-}_{u,i}.\psi(\epsilon^{-}_{i,k}) - R.\psi(\epsilon^{+}_{i,k} + \epsilon^{-}_{i,k}))$$
$$\lambda_{u,i,k} = \frac{\lambda^{'}_{u,i,k}}{\lambda^{'}_{u,i,1} + ... + \lambda^{'}_{u,i,K}} \tag{4.10}$$
where,
  - ∗ ψ is the digamma function
  - ∗ $r^{+}_{u,i} = \rho_{u,i} = R.r^{*}_{u,i}$
  - ∗ $r^{-}_{u,i} = R - \rho_{u,i} = R.(1 - r^{*}_{u,i})$

Using these parameters, we find the feature vectors and predicted ratings as:

- The probability $a_{u,k}$ of a user u belonging to the group k -

$$a_{u,k} = \frac{\gamma_{u,k}}{\sum_{j=1 \ldots K} \gamma_{u,j}} \tag{4.11}$$

- The probability $b_{k,i}$ of users in group k liking the item i -

$$b_{k,i} = \frac{\epsilon^{+}_{i,k}}{\epsilon^{+}_{i,k} + \epsilon^{-}_{i,k}} \tag{4.12}$$

- The probability of a user u liking the item i is given by -

$$p_{u,i} = \sum_{k=1}^{K} a_{u,k} b_{k,i} \tag{4.13}$$

To solve the cold start problem which arises due to new users or new items that do not have any ratings, a variety of approaches have been proposed [16, 17] based on asking the users to manually rate some items that are not feasible.

Our approach is based on the popularity based approach [16] that tries to populate the rating matrix instead of manually asking for ratings.

The training algorithm is given by -

- Input: $M, \alpha, \beta, ut, it, epochs$
- Output: $a_{u,k}, b_{k,i}, p_{u,i}, q_{u,i}$
- Steps:
  - Find the *it* most popular items from M based on the number of users that have rated the item.
  - Find *ut* users from M that have the least number of ratings.
  - Assign the average of the ratings of these *it* items to the *ut* users provided that they have not rated the item previously.
  - Initialize $\gamma_{u,k}$ by assigning random values
  - Initialize $\epsilon^{+}_{i,k}$ by assigning random values
  - Initialize $\epsilon^{-}_{i,k}$ by assigning random values
  - For each *epoch* in number of *epochs*:
    - For every user u in N:

    * For every item i that has been rated by the user u in the training dataset:
    * For every factor k: update $\lambda_{u,i,k}$ using Equation 4.10
    * For every user u:
    * For every factor k: update $\gamma_{u,k}$ using Equation 4.7
    * For every item i that has been rated by the user u in the training dataset:
    * For every factor k: update $\epsilon^{+}_{i,k}$ using Equation 4.8
    * For every factor k: update $\epsilon^{-}_{i,k}$ using Equation 4.9
  - Output $a_{u,k}, b_{k,i}, p_{u,i}, q_{u,i}$ using Equations 4.11, 4.12, 4.13, 4.2

4.2.2 KNN Modification

To solve the cold start problem which arises due to new users or new items that do not have any ratings we use a popularity based approach [16] that tries to populate the rating matrix instead of manually asking for ratings.

Algorithm:

- Input: M, ut, it
- Find the it most popular items from M based on the number of users that have rated the item.
- Find ut users from M that have the least number of ratings.
- Assign the average of the ratings of these it items to the ut users provided that they have not rated the item previously.
- Apply KNN Algorithm with Cosine Similarity on training data
- Output predictions for testing data

Chapter 5

Evaluation

5.1 Experimental Setup

The experiments were carried out using MovieLens 100k dataset and MovieLens 1M dataset [23]. The ratings file consists of – UserID of a user, MovieID of the movie rated and rating from (1-5). The movies file consists of – MovieID, corresponding movie name and its genres. The MovieLens 100k dataset contains 1,00,000 ratings (1-5) from 943 users on 1,682 items. Each user has rated at least 20 movies. The dimensions of the user-item rating matrix are 943 users x 1682 movies, which amounts to 1,586,126 ratings, of which only 100,000 ratings are given. This gives an idea about the degree of sparsity of the user-item rating matrix. For this dataset, we get:

- Number of users that have rated less than 10 items = 0
- Number of users that have rated less than 20 items = 14
- Number of users that have rated less than 30 items = 112
- Number of movies with more than 30 ratings = 882
- Number of movies with more than 40 ratings = 638

The MovieLens 1M dataset is collected over various time periods. This dataset contains 1M+ ratings from 6,000 users on 4,000 movies. The ratings are in the range (1-5).

We have evaluated our KNN modification for both MovieLens 100k and MovieLens 1M datasets. For our Probabilistic Matrix Factorization approach, we have used MovieLens 100k dataset to evaluate our algorithm.

5.2 Parameter Setup

5.2.1 For KNN Modification

We evaluated our model for different values of ut and it and concluded from our experiments the best values for these parameters.

Table 5.1: KNN Modification Parameters for MovieLens 100k and 1M Datasets

Parameter	**Value**
k	5
ut	30
it	10
Similarity	Cosine

5.2.2 For Probabilistic Matrix Factorization

We have used the parameter values specified in [14]. We use various ut and it values to determine the best values for these parameters.

Table 5.2: Probabilistic Matrix Factorization Parameters for MovieLens 100k Dataset

Parameter	**Value**
K	6
α	0.8
β	5
R	4
Epochs	20

5.3 Evaluation Metrics

The algorithms were evaluated for the metrics mentioned in Section 3.1 on testing data after training the model for given rating $r_{u,i}$ and finding the predicted rating $p_{u,i}$.

The objective is to minimize MAE, CMAE, MSE, RMSE and to maximise Precision, Recall and F-score.

For KNN, Surprise package [24] is used to evaluate MAE, MSE, RMSE, Precision and Recall. The time taken to fit the training data (Fit Time) and Test Time are also considered. n-fold cross validation method is used to evaluate the algorithm. Cross-validation is process that is used to evaluate a machine learning model on a small dataset. The given dataset is split into n different groups. In each iteration, one group is taken as the test data while the remaining groups are taken as the training data. The average of the results of all iterations is then considered. We use n-fold cross validation method as it is easy to implement and understand, and gives results that are less biased compared to other methods as testing is done for different groups. The steps involved in n-fold cross validation are as follows:

- Randomly shuffle the dataset
- Divide the given dataset into n groups of same size
- For each group g in the dataset:
  - Take group g as the testing data
  - Take the remaining n - 1 groups as the training data
  - Train the model on the training data
  - Use the testing data to evaluate the model on the given metrics
  - Store the results of evaluating the model on the given testing data
- Find the Mean and Standard Deviation (Std) of the results from all n iterations

For Probabilistic Matrix Factorization, we have evaluated our algorithm on MAE, CMAE and RMSE. We have also considered the time taken by each iteration in the training process and compared the results by the algorithm in [14].

Chapter 6

Results and Discussion

6.1 KNN Results for MovieLens 100k dataset

Results for running the KNN algorithm are shown in Table 6.1 for MAE, MSE, RMSE, Precision and Recall. Here, the similarity metric is cosine similarity. The value of k (no. of recommendations) is 5. 5-fold cross validation is used to evaluate the model.

Table 6.1: KNN using 5-fold Cross Validation for 100k Dataset

Metric	Fold1	Fold2	Fold3	Fold4	Fold5	Mean	Std
MAE	0.7468	0.7519	0.7476	0.7478	0.7487	0.7485	0.0018
MSE	0.9434	0.9507	0.9449	0.9367	0.9401	0.9432	0.0047
RMSE	0.9713	0.9751	0.9720	0.9678	0.9696	0.9712	0.0024
Precision	0.6502	0.6542	0.6608	0.6582	0.6609	0.6569	0.0041
Recall	0.2532	0.2595	0.2593	0.2669	0.2761	0.2630	0.0078
Fit Time	0.37	0.38	0.37	0.37	0.38	0.37	0.00
Test Time	1.76	1.73	1.73	1.72	1.84	1.76	0.04

Table 6.2 shows the results of running the proposed improvement to KNN given in subsection 4.2.2 on the MovieLens 100k dataset. Model is evaluated for for MAE, MSE, RMSE, Precision and Recall. Here, the similarity metric is cosine similarity. The value of k (no. of recommendations) is 5. 5-fold cross validation is used to evaluate the model. The user threshold ut - the number of users with least ratings considered, is taken as 30 and the item threshold it - the number of most popular items considered, is taken as 10.

Table 6.2: KNN Improvement using 5-fold Cross Validation with $ut = 30, it = 10$ for 100k Dataset

Metric	**Fold1**	**Fold2**	**Fold3**	**Fold4**	**Fold5**	**Mean**	**Std**
MAE	0.7417	0.7365	0.7373	0.7430	0.7429	0.7403	0.0028
MSE	0.9445	0.9233	0.9299	0.9286	0.9382	0.9329	0.0075
RMSE	0.9718	0.9609	0.9643	0.9636	0.9686	0.9659	0.0039
Precision	0.7194	0.7106	0.7344	0.7197	0.6979	0.7164	0.0119
Recall	0.3079	0.3092	0.3026	0.2989	0.2962	0.3030	0.0050
Fit Time	0.36	0.38	0.40	0.40	0.40	0.39	0.02
Test Time	1.79	1.80	1.90	1.85	1.82	1.83	0.04

From Table 6.1 and Table 6.2 it is inferred that the MAE, MSE, RMSE show slight improvement in our algorithm compared to KNN while the Precision and Recall metrics show a significant improvement. The Fit Time and Test Time remain almost same.

6.2 KNN Results for MovieLens 1M dataset

In section 6.1 it is observed that our algorithm shows an improvement over KNN for the MovieLens 100k dataset. But in this section we observe the results of running the algorithms on the much larger MovieLens 1M dataset.

Results for running the KNN algorithm on MovieLens 1M dataset are shown in Table 6.3 for MAE, MSE, RMSE, Precision and Recall. Here, the similarity metric is cosine similarity [25]. The value of k (no. of recommendations) is 5. 5-fold cross validation is used to evaluate the model.

Table 6.3: KNN using 5-fold Cross Validation for 1M Dataset

Metric	**Fold1**	**Fold2**	**Fold3**	**Fold4**	**Fold5**	**Mean**	**Std**
MAE	0.7322	0.7304	0.7306	0.7300	0.7314	0.7309	0.0008
MSE	0.9054	0.9053	0.9037	0.9029	0.9065	0.9048	0.0013
RMSE	0.9515	0.9515	0.9506	0.9502	0.9521	0.9512	0.0007
Precision	0.6571	0.6512	0.6531	0.6459	0.6445	0.6504	0.0046
Recall	0.2535	0.2516	0.2478	0.2480	0.2445	0.2491	0.0031
Fit Time	71.89	70.92	72.21	70.81	71.95	71.56	0.57
Test Time	142.92	145.28	145.00	143.01	140.87	143.42	1.60

Table 6.4 shows the results of running the proposed improvement to KNN given in subsection 4.2.2 on the MovieLens 1M dataset. Model is evaluated for for MAE, MSE, RMSE, Precision and Recall. Here, the similarity metric is cosine similarity. The value of k (no. of recommendations) is 5. 5-fold cross validation is used to evaluate the model. The user threshold ut - the number of users with least ratings considered, is taken as 30 and the item threshold it - the number of most popular items considered, is taken as 10.

Table 6.4: KNN Improvement using 5-fold Cross Validation with $ut = 30, it = 10$ for 1M Dataset

Metric	**Fold1**	**Fold2**	**Fold3**	**Fold4**	**Fold5**	**Mean**	**Std**
MAE	0.7226	0.7231	0.7207	0.7216	0.7228	0.7222	0.0009
MSE	0.8920	0.8940	0.8920	0.8949	0.8949	0.8936	0.0013
RMSE	0.9444	0.9455	0.9445	0.9460	0.9460	0.9453	0.0007
Precision	0.6746	0.6695	0.6721	0.6643	0.6632	0.6687	0.0044
Recall	0.2652	0.2578	0.2623	0.2559	0.2594	0.2601	0.0033
Fit Time	82.00	87.10	83.48	82.16	83.50	83.65	1.84
Test Time	179.01	170.82	179.64	178.05	176.50	176.80	3.17

From Table 6.3 and Table 6.4 it is inferred that the MAE, MSE, RMSE show slight improvement in our algorithm compared to KNN while the Precision and Recall metrics show a significant improvement. The Fit Time and Test Time remain almost same.

6.3 Probabilistic Matrix Factorization

Table 6.5 shows the comparison between the probabilistic matrix factorization method mentioned in [14] and our proposed improvement mentioned in subsection 4.2.1 using different threshold values (ut and it). The values of the parameters [14] taken are $\alpha = 0.8$, $\beta = 5$, $R = 4$ and $K = 6$.

Table 6.5: Comparison of Probabilistic Matrix Factorization (PMF)

Algorithm	**MAE**	**CMAE**	**RMSE**	**Average Time Taken Per Epoch**
PMF	0.717868	0.713281	0.984844	139.74789 s
PMF ut=30, it=10	0.711661	0.709881	0.977580	40.535576 s

Table 6.5 continued from previous page

Algorithm	**MAE**	**CMAE**	**RMSE**	**Average Time Taken Per Epoch**
PMF *ut*=**40,** *it*=**10**	0.725594	0.729429	0.998778	39.024904 s
PMF *ut*=**30,** *it*=**20**	0.726983	0.727860	1.002835	39.844838 s
PMF *ut*=**40,** *it*=**20**	0.744143	0.743919	1.029879	39.509199 s

where,
ut is the number of users with least ratings considered
it is the number of most popular items considered

From Table 6.5 it can be inferred that our proposed improvement leads to a significant improvement in the average training time taken per epoch. The algorithm performs best for parameters $ut = 30$, $it = 10$. For these values of parameters, the algorithm shows slight improvement in terms of MAE, CMAE and RMSE.

Our algorithm gives better average training time taken per epoch as we have populated the sparse training matrix initially before fitting the model. A sparse matrix with very few ratings takes a longer time to train. By removing the sparsity, our algorithm gives better average time per epoch. Our algorithm removes the user cold start issue by giving an initial rating to users with few ratings. This improves the MAE, CMAE and RMSE values.

Chapter 7

Conclusions and Future work

7.1 Conclusion

In this project we have presented a new matrix factorization algorithm for recommending items to users to solve the cold start problem. In our approach we have populated the training data by assigning the average of popular movies to sparse users. We have also proposed a modified KNN algorithm that tries to solve the cold start problem. Experimental results prove that presented matrix factorization algorithm gives training time far better than existing algorithms while giving slightly better results for MAE, CMAE and RMSE compared to existing algorithms. The proposed KNN algorithm gives better MAE, MSE, RMSE, Precision and Recall results compared to existing KNN algorithms for chosen threshold parameters.

7.2 Future Work

There are several aspects where we can still do some research.

1. The discussed algorithms were tested for various parameter values and we concluded the best values from it. To make the process efficient, we can find an efficient algorithm to find a suitable user and item threshold for any dataset instead of relying on hit and trial method.

2. The algorithms were tested on the MovieLens dataset.To generalize, we can evaluate the algorithms for different datasets - e-commerce, books, music, etc to check the efficiency of our algorithms for different domains.

3. In the discussed algorithms, equal weightage is given to each user and item. In reality every user and item will not be of equal importance. We can extend our algorithms by giving more weightage to some users and items in finding the predicted ratings.

Bibliography

[1] Jinha Kim Jinoh Oh, Sungchul Kim and Hwanjo Yu. When to recommend: A new issue on tv show recommendation. *Information Sciences*, 280:261–274, 2014.

[2] Kazunori Komatani Tetsuya Ogata Kazuyoshi Yoshii, Masataka Goto and Hiroshi G. Okuno. An efficient hybrid music recommender system using an incrementally trainable probabilistic generative model. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):435–447, 2008.

[3] Yu-Ling Hsueh Ren-Hung Hwang and Yu-Ting Chen. An effective taxi recommender system based on a spatio-temporal factor analysis model. *Information Sciences*, 314:28–40, 2015.

[4] Francesco Ricci. Travel recommender systems. *IEEE Intelligent Systems*, 17:55–57, 2002.

[5] Jeonhyung Kang and Kristina Lerman. La-ctr: A limited attention collaborative topic regression for social media. *Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013,*, 11, 2013.

[6] Xinjun Guan Lei Li Chen Lin, Runquan Xie and Tao Li. Personalized news recommendation via implicit social experts. *Information Sciences*, 254:1–18, 2014.

[7] Fernando Ortega Jesús Bobadilla and Antonio Hernando. A collaborative filtering similarity measure based on singularities. *Information Processing and Management*, 48(2):204–217, 2012.

[8] Fernando Ortega Jesús Bobadilla and Antonio Hernando. A new collaborative filtering metric that improves the behavior of recommender systems. *Knowledge-Based Systems*, 23(6):520–528, 2010.

[9] Fernando Ortega Jesús Bobadilla and Antonio Hernando. Improving collaborative filtering recommender system results and performance using genetic algorithms. *Knowledge-Based Systems*, 24(8):1310–1316, 2011.

[10] Fernando Ortega Jesus Bernal Jesus Bobadilla, Antonio Hernando. A framework for collaborative filtering recommender systems. *Expert Systems with Applications*, 38:14609–14623, 2011.

[11] Fernando Ortega Jesús Bobadilla and Antonio Hernando. A similarity metric designed to speed up, using hardware, the recommender systems k-nearest neighbors algorithm. *Knowledge-Based Systems*, 51:27–34, 2013.

[12] Sanjib Kumar Nayak and Sanjaya Kumar Panda. A user-oriented collaborative filtering algorithm for recommender systems. *5th IEEE International Conference on Parallel, Distributed and Grid Computing*, 2018.

[13] Manas Ranjan Senapati Sanjaya Kumar Panda and Pradip Kumar Sahu. An item-oriented collaborative filtering algorithm for recommender systems. *The Institute of Engineers*, 2019.

[14] Fernando Ortega Antonio Hernando, Jesús Bobadilla. A non negative matrix factorization for collaborative filtering recommender systems based on a bayesian probabilistic model. *Knowledge-Based Systems*, pages 15–43, 2016.

[15] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. NIPS, 2007.

[16] Fernando Ortega Abraham Gutierrez Antonio Hernando, Jesús Bobadilla. A probabilistic model for recommending to new cold-start non-registered users. *Information Sciences*, 2016.

[17] Mohammad-Hossein Nadimi-Shahraki and Mozhde Bahadorpour. Cold-start problem in collaborative recommender systems: Efficient methods based on ask-to-rate technique. *Journal of Computing and Information Technology*, 2:105–113, 2014.

[18] Munesh Singh Sanjaya Kumar Panda, Sourav Kumar Bhoi. A collaborative filtering recommendation algorithm based on normalization approach. *Journal of Ambient Intelligence and Humanized Computing*, 2020.

[19] Jesus Bobadilla Fernando Ortega, Antonio Hernando and Jeon Hyung Kang. Recommending items to group of users using matrix factorization based collaborative filtering. *Information Sciences*, 345:313–324, 2016.

[20] Fernando Ortega Jorge Tejedor Antonio Hernando, Jesús Bobadilla. Incorporating reliability measurements into the predictions of a recommender system. *Information Sciences*, 218:1–16, 2013.

[21] A. Hernando J. Bobadilla, F. Serradilla. Collaborative filtering adapted to recommender systems of e-learning. *Knowledge-Based Systems*, 22:261–265, 2009.

[22] Antonio Hernando Esteban Remigio Hurtado Jesus Bobadilla, Rodolfo Bojorque. Recommender systems clustering using bayesian non negative matrix factorization. *IEEE Access*, 6:3549–3564, 2018.

[23] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), 2015.

[24] Nicolas Hug. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174, 2020.

[25] Tanisha Tripathi Tushar Narula Gaurav Srivastav Ramni Harbir Singh, Sargam Maurya. Movie recommendation system using cosine similarity and knn. *International Journal of Engineering and Advanced Technology (IJEAT)*, 9(5):2249–8958, 2020.

Acknowledgement

Its our privilege and honor to express our gratitude to all the respected personalities who have guided us, inspired us and supported us in the smooth and successful completion of our project. We thank our beloved faculty supervisor Dr. Sanjay Kumar Panda, Assistant Professor, Department of Computer Science and Engineering (CSE), National Institute of Technology, Warangal, for his persistent supervision, guidance, suggestions and encouragement during this project. He has motivated us during the low times and gave us courage to move ahead positively.

We are grateful to Prof. P. Radha Krishna, Head of the Department, Computer Science and Engineering, National Institute of Technology, Warangal for his moral support to carry out this project. We are also thankful to the Project Evaluation Committee, for their strenuous efforts to evaluate our projects. We wish to thank all the staff members in the department for their kind cooperation and support given throughout our project work. We are thankful to all of our colleague friends who have given valuable suggestions and help in all stages of the development of the project.

Radhika Sunil Jain (177249)
Srikavya (177257)
Ajmeera Sreekanth (177202)