Design Document

Information Retrieval Assignment-3

Group member information:-

Shashank Pratap Singh	2019B4A70956H
Satvik Omar	2019B4A70933H
Utkarsh Tiwari	2019B1A71147H
Gaurav Kumar	2019B3A71324H

Introduction

As the World Wide Web continues to grow at an exponential rate, the size and the complexity of many websites grows along with it. For the users of these websites it becomes increasingly difficult and time consuming to find the information they are looking for. Recommender Systems provide personalized information by learning the user's interests from traces of interaction with that user.

This project addresses the advantages as well as limitations of current algorithms used to implement recommendation systems. It compares the different techniques on the basis of their errors using Root Mean Square Error, Precision on top K and Spearman Rank Correlation. The prediction time is also calculated for each case. This report provides a detailed summary of the project.

Dataset

In order to construct our Utility Matrix along with a test set of ratings, a dataset of 1,000,000 ratings from 6000 users on 3900 movies was taken from MovieLens. Every user has rated at least 20 movies hence We have a decent amount of history for each user. 80% of the 1 Million ratings were used to construct our Utility Matrix which then was used to train our models and help in predictions of future ratings. The remaining 20% of the ratings were used as a test set to evaluate the performance of our models.
Dataset link: https://grouplens.org/datasets/movielens/1m/

Data Preprocessing: We first preprocess the data to shuffle and split all the entries into training and testing data, in the ratio of 4:1. We then create matrices and normalize it, and finally, save the sparse matrices into the respective files/

Assumptions

1. Collaborative Filtering Neighborhood size : 150
2. SVD concepts (number of Eigen values) : 40
3. CUR number of rows and columns chosen : 160 (4*SVD concepts)
4. Distribution of testing and training data: 80% training 20% testing
5. K- precision is calculated over the top 100 ratings

Handling of strict and generous rates

We calculated the row average for each row, and subtracted it from each non-zero value in the row to normalize it, which helps us handle strict and generous rates

Techniques

1) Collaborative Filtering.

2) Collaborative Filtering with baseline approach.

3) Singular Value Decomposition.

4) Singular Value Decomposition with 90% retained Energy.

5) CUR Decomposition.

6) CUR Decomposition with 90% retained Energy.

Collaborative Filtering

Without Baseline:

Formula:

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

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

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

With Baseline:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} \cdot (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

baseline estimate for r_{xi}

$$b_{xi} = \mu + b_x + b_i$$

- μ = overall mean movie rating
- b_x = rating deviation of user x
  = (avg. rating of user x) $- \mu$
- b_i = rating deviation of movie i

The collaborative filtering method used is based on user-user collaborative filtering. It predicts the ratings of a user for a movie based on the ratings of other users who have rated the same movie and who have similar rating patterns with the user being predicted.

We first calculate the similarity between users using the dot product of their rating vectors. Then, we identify the 150 most similar users for each user and calculate the weighted average of their ratings for the movie being predicted. The weights are the similarity scores between the users. Finally, we return a collaborative matrix where the missing ratings are predicted based on the collaborative filtering method.

Additionally, we also have used a baseline approach, which predicts the ratings based on the average rating of the user and movie plus an overall mean rating. The baseline approach is combined with the collaborative filtering method to improve the accuracy of the predicted ratings.

Pros: (Non baseline)

1. No feature selection needed: Recommendations are based on user-user or item-item similarity.

Cons: (Non baseline)

1. Cold Start: Need enough users in the system to find a match.
2. Sparsity: The user/ratings matrix is sparse so it is hard to find users that have rated the same items.
3. First Rater: Cannot recommend an item that has not been previously rated and have a hard time rating new items.
4. Popularity bias: It tends to recommend popular items so it cannot recommend items to someone with unique taste.

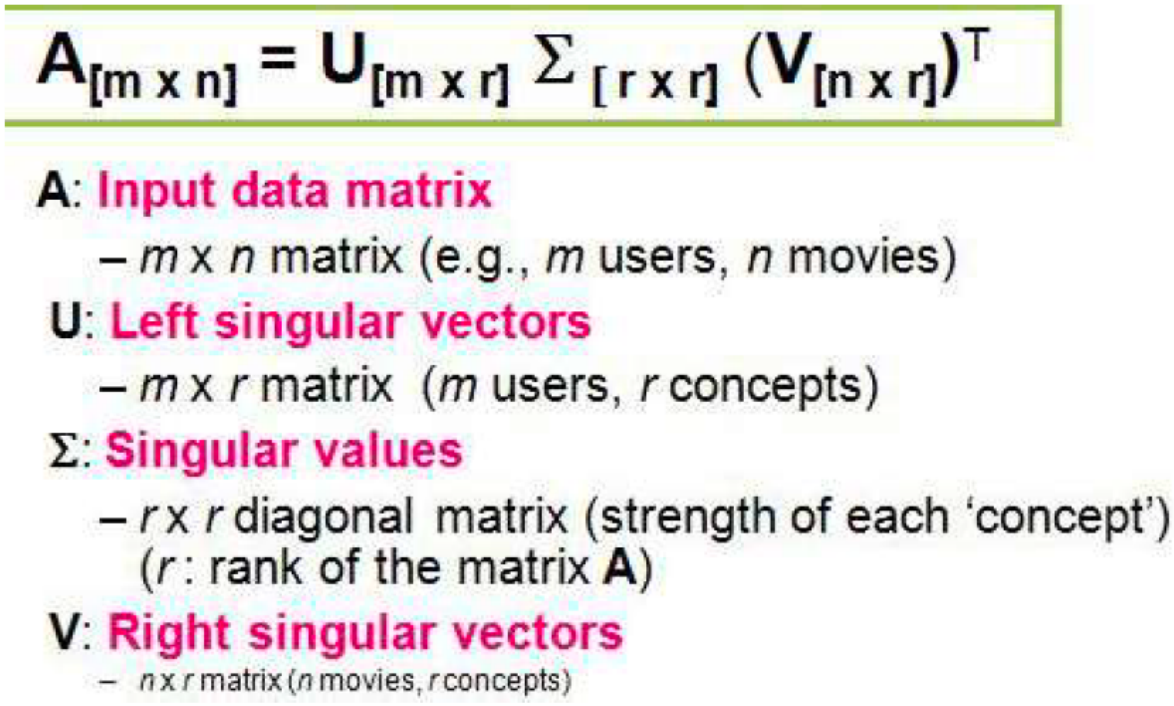
Pros: (Baseline)

1. No feature selection needed: Recommendations are based on user-user or item-item similarity.
2. Automatically takes care of Strict and lenient raters.

Cons: (Baseline)

1. Sparsity: The user/ratings matrix is sparse so it is hard to find users that have rated the same items.
2. First Rater: Cannot recommend an item that has not been previously rated and have a hard time rating new items.
3. Popularity bias: It tends to recommend popular items so it cannot recommend items to someone with unique taste.

SVD



We implemented a method for performing Singular Value Decomposition (SVD) on a given sparse matrix. SVD is a commonly used method in recommender systems to create user-item matrices by decomposing a large matrix into smaller matrices that represent user and item factors.

The program uses the `scipy.sparse` library to handle sparse matrices efficiently.

We first perform the actual SVD decomposition by calculating the eigenvalues and eigenvectors of the matrix using the `scipy.sparse.linalg.eigsh` function. We then calculate the singular values and the left and right singular vectors. and then calculate the number of singular values that must be retained to retain the desired energy percentage. Then, we return the truncated left and right singular vectors and the retained singular values. Finally, we calculate the dot product of the three matrices to obtain the final SVD matrix. We also allow specifying the desired energy percentage to retain.

Pros:

1. Discovers hidden correlations/topics. For example, words that occur commonly together.

2. Removes redundant and noisy features. Features that are highly correlated to some other feature can be removed by reducing the dimensions of the sigma matrix.
3. Easier storage and processing of data.

Cons:

1. Computational cost is cubic time in the size of data to compute.
2. The results are dense vectors hence could take lots of space and be hard to interpret.

CUR

- **Sampling columns (similarly for rows):**

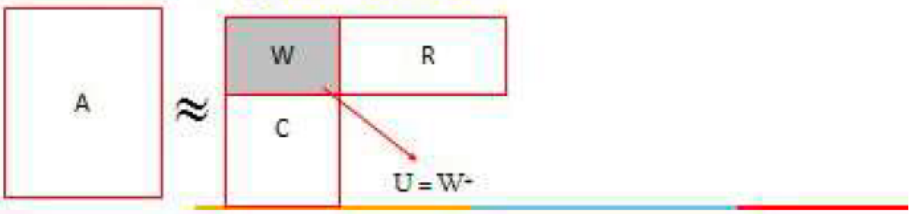
Total length of all the columns

Input: matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, sample size c
Output: $\mathbf{C}_d \in \mathbb{R}^{m \times c}$

1. for $x = 1 : n$ [column distribution]
2. $P(x) = \sum_i \mathbf{A}(i,x)^2 / \boxed{\sum_{i,j} \mathbf{A}(i,j)^2}$
3. for $i = 1 : c$ [sample columns]
4. Pick $j \in 1 : n$ based on distribution $P(j)$
5. Compute $\mathbf{C}_d(:,i) = \mathbf{A}(:,j) / \sqrt{cP(j)}$

Note this is a randomized algorithm, same column can be sampled more than once

- Let **W** be the "intersection" of sampled columns **C** and rows **R**
  - Let SVD of **W** = **X Z Y**^{T}
- **Then: U** = **W**$^+$ = **Y Z**$^+$ **X**^{T}
  - **Z**$^+$: **reciprocals of non-zero singular values:** $\mathbf{Z}^+_{ii} = 1/\mathbf{Z}_{ii}$
  - W$^+$ is the "**pseudoinverse**"



dense but small

CUR: A = C U R

Huge but sparse Big but sparse

The next recommender system method being used is CUR decomposition. The CUR decomposition is a matrix factorization method that approximates a given matrix using a subset of its columns and rows. The C, U, and R matrices obtained by the decomposition can be used to make recommendations based on a user's past behavior. In our program, the CUR decomposition is performed on a sparse matrix, and the resulting matrices are used to generate recommendations. The program also uses SVD (Singular Value Decomposition) to obtain a low-rank approximation of the intersection matrix, which is formed by taking the intersection of the selected rows and columns. The resulting low-rank approximation is used to compute the U matrix, which is then used in the final CUR matrix computation.

We first select the rows and columns with highest probabilistic weight from the given sparse matrix. We calculate the probabilistic weight of each column/row by summing the squares of its entries and dividing by the sum of squares of all the entries in the matrix (Frobenius norm) . We discard columns/rows with zero Frobenius norm. Next, we create an intersection matrix W by taking the intersection of the rows and columns selected. We then apply SVD (singular value decomposition) on the intersection matrix W to get its left singular vectors, singular values, and right singular vectors. Using the obtained matrices, we compute the U matrix by performing a matrix multiplication of the transposed right singular vectors, the reciprocal of the squared singular values, and the transposed left singular vectors. Finally, we compute the CUR approximation of the original sparse matrix by performing a matrix multiplication of the C, U, and R matrices.

Pros:

1. Easy interpretation: Since the basis vectors are actual columns and rows.
2. Sparse basis: Since the basis vectors are actual columns and rows.

Cons:

1. Duplicate columns and rows: Columns of large norms will be sampled many times.

Error Calculation

RMSE:

- **Root-mean-square error** (RMSE)
  - $\sqrt{\sum_{xi}(r_{xi} - r^*_{xi})^2}$ where r_{xi} is predicted, r^*_{xi} is the true rating of x on i

Spearman Correlation:

- Spearman's *correlation* between system's and user's complete rankings

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

where d_i = absolute difference in user and predicted ratings
n = number of ratings

Precision on top-k:

k=100

relevance=3

Value for each user = (number of predicted ratings over relevance)/(number of ratings actually greater than relevance)

Precision on top k=Mean of all values of relevance

Libraries/Packages used

1. Random: This package gives us access to the random function, which returns a completely random value. We used it to split the data into the training and testing sets.
2. Numpy: NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices. We used it for a lot of calculations involving matrices and arrays.

3. Scipy: SciPy is a collection of mathematical algorithms and convenience functions built on the NumPy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. We used it to manipulate our matrices for calculations and things like that.
4. Os: The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc. We used it for fetching data from the files we had.
5. Math: math is a built-in module in the Python 3 standard library that provides standard mathematical constants and functions. It allowed us to use a lot of mathematical functions.
6. Time: The Python time module provides many ways of representing time in code, such as objects, numbers, and strings. We used it to calculate the time taken by our programs to execute.
7. Operator: The Python operator module is one of the inbuilt modules in Python, and it provides us with a lot of functions such as add(x, y), floordiv(x, y) etc., which we can use to perform various mathematical, relational, logical and bitwise operations on two input numbers. This library, again, helped us perform mathematical operations.

Table

Method	**RMS**	**Spearman correlation(%)**	**Precision over top 100(%)**	**Time taken(secs)**
Collaborative Filtering without baseline	0.0066736165 9075542	99.999999866 41724%	99.700733477 09781%	63.10
Collaborative Filtering with baseline	0.0083324433 5863803	99.999999791 75594%	99.682272879 71672%	106.31
SVD with 100% retain energy	0.0021089225 502688236	99.999999986 6602%	99.519673001 35884%	16.03
SVD with 90% retain energy	0.0021089225 50268823	99.999999986 6762%	99.519673101 35884%	15.50

CUR with 100% retain energy	0.002261645334683975	99.99999998465819%	99.25426280761134%	8.89
CUR with 90% retain energy	0.002261645334683975	99.99999998465819%	99.25426280761134%	8.02