Information Retrieval (CS F469)
Assignment 2

Design Document- Recommender Systems

By:
Ayush Kumar Tiwary- 2016A2PS0567H
Mukesh Basira- 2016A7PS0116H
Srijan Soni- 2016A4PS0328H
Vivek Pratap Deo- 2016A7PS0056H

Overview

As the World Wide Web continues to grow at an exponential rate, the size and complexity of many websites grow along with it. For the users of these web sites 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.

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

Collaborative filtering gathers information by using the recommendations of other people. It is based on the idea that people who agreed in their evaluation of certain items in the past are likely to agree again in the future. A person who wants to see a movie for example, might ask for recommendations from friends. The recommendations of some friends who have similar interests are trusted more than recommendations from others. This information is used in the decision on which movie to see.

Assumptions:

The users who have similar preferences in the past are likely to have similar preferences in the future. It is this assumption which allows us to take a user's history and extrapolate into their future and predict items which they might enjoy.

Formulation:

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

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

Pros and Cons:

Pros:

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

Cons:

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.

Collaborative Filtering with Baseline

Assumptions:

The users who have similar preferences in the past are likely to have similar preferences in the future.

Formulation:

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

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) – μ
b_i = Rating deviation of movie i.
r_{xi} = rating of item by user i.

20 Closest neighbors were considered while calculating rating.

Pros and Cons:

Pros:

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

Cons:

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.

Singular Value Decomposition

Assumptions:

1) Data lies on or near a low d-dimensional space.
2) Retaining 90% of energy allows us to retain enough information to approximately reconstruct the original matrix.

Formulation:

$$\mathbf{A}_{[m\times n]} = \mathbf{U}_{[m\times r]}\Sigma_{[r\times r]}\left(\mathbf{V}_{[n\times r]}\right)^{\top}$$

A: Input data matrix (m x n matrix=> m users, n movies)
r : rank of the matrix A
U: Left singular vectors (m x r matrix=> m users, r concepts)
Σ: Singular values (r x r diagonal matrix=> strength of each 'concept')
V: Right singular vectors (n x r matrix=> n movies, r concepts)

$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i$$

Matrix A is the sum of different matrices which is represented as outer product of different vectors.

σ_i = scalar
u_i = vector
v_i = vector

Pros and Cons:

Pros:

1) Discover hidden correlations/topics. For example, words that occur commonly together.
2) Remove redundant and noisy features. Features that are highly correlated to some other feature can be removed by reducing the dimensions of 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 hard to interpret.

CUR Decomposition

CUR matrix decomposition is a low-rank matrix decomposition algorithm that is explicitly expressed in a small number of actual columns and/or actual rows of data matrix. A CUR matrix approximation of a matrix A is three matrices C, U, and R such that C is made from columns of A, R is made from rows of A, and that the product CUR closely approximates A.

Assumptions:

We can approximately reconstruct the original matrix multiplying selected columns matrix and selected rows matrix and Pseudo inverse of SVD of intersection of the former two matrices.

Formulation:

$$\|\mathbf{A} - \mathbf{CUR}\|_\mathbf{F} \leq \|\mathbf{A} - \mathbf{A_k}\|_\mathbf{F} + \varepsilon\|\mathbf{A}\|_\mathbf{F}$$

A is Original Matrix.
A_k is the “best” rank k approximation to A (that is, A_k is SVD of A)

Where

$$\|\mathbf{A}\|_F \equiv \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2}$$

$$P(x) = \sum_i \mathbf{A}(i,x)^2 / \sum_{i,j} \mathbf{A}(i,j)^2$$

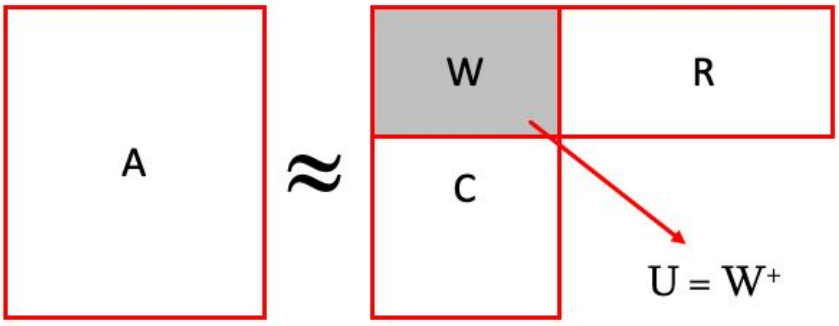
$$\mathbf{C}_d(:,i) = \mathbf{A}(:,j)/\sqrt{cP(j)}$$

Let W be the “intersection” of sampled columns C and rows R.

$$\mathbf{W} = \mathbf{XZY}^\top$$

Z^+ is reciprocals of non-zero singular values. W^+ is the pseudoinverse.

$$U = W^+ = YZ^+X^\top \qquad Z^+_{ii} = 1/Z_{ii}$$



Pros and Cons:

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.

Results

We evaluated our code based on 3 factors:-

1) Root Mean Square Error
2) Spearman Correlation Coefficient.
3) Top k Precision.

Recommender System Technique	**Root Mean Square Error (RMSE)**	**Precision on top K**	**Spearman Rank Correlation**	**Time taken for prediction**
Collaborative Filtering	0.785856390	0.89364303	0.9999956030	172.02 sec
Collaborative Filtering with Baseline	0.754405648	0.960880195	0.999995947	186.42 sec
SVD	$7.35624*10^{-13}$	0.8838630806	0.9999999999	76.57 sec
SVD with 90% retained energy	0.751826998	0.759168704	0.9999999999	68.91 sec
CUR	3.248058695	0.552755905	0.999999999	42.66 sec
CUR with 90% retained energy	3.419289905	0.2629921259	0.999999999	37.44 sec

Packages/Libraries Used:

The whole project is done with Python 3. The following packages are used in the project:

1) Numpy
2) Pandas
3) Math
4) Collections