Birla Institute of Technology and Science-Pilani,
Hyderabad Campus
First Semester 2020-21

Information Retrieval (CS F469)

Design Document – Recommender Systems

by

Rojan Sudev	2019H1030008H
Faishal Hussain Siddiqui	2019H1030012H
Arpit Roy	2019H1030118H
Suraj Abhiman Shinde	2019H1030507H

Under the guidance of

Dr. Aruna Malapati

Recommender Systems

Overview

The immense measure of information accessible on the Internet has prompted the improvement of recommendation systems. This project and report tend to the constraints of current calculations used to implement recommendation systems, assessment of experimental results, and conclusion. This report provides a detailed summary of the project

Goals

1. Collaborative Filtering(user-user,item-item)
2. Collaborative Filtering with a baseline approach(user-user,item-item)
3. Singular Value Decomposition
4. Singular Value Decomposition with 90% retained Energy
5. CUR Decomposition with 100%
6. CUR Decomposition with 90%

Collaborative Filtering

Assumptions

- We can predict the rating of item i by user x through a set of other users whose ratings are “similar” to x’s ratings or by a set of other items whose ratings are “similar” to i’s ratings.

Formulation

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

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

r_{xj} => rating of item i by user x

Pros and Cons

Pros

- No feature selection is needed. Recommendations are based on user-user or item-item similarity.
- Takes care of strict and lenient raters.

Cons

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

Collaborative Filtering with baseline

Assumptions

- We can predict the rating of item i by user x through a set of other users whose ratings are “similar” to x’s ratings or by a set of other items whose ratings are “similar” to i’s ratings.
- We also need to consider the deviation of that item i and user x from the over-all mean of the corpus.

Formulation

$$r_{xi} = b_{xi} + \sum_{j \varepsilon N(i;x)} S_{ij} . (r_{xj} - b_{xi}) \div \sum_{j \varepsilon N(i;x)} S_{ij}$$

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

b_{xi} =>baseline estimate for r_{xi}

μ => overall mean movie rating

b_x => rating deviation of user x

b_i => rating deviation of movie i

Pros and Cons

Pros

- No feature selection is needed. Recommendations are based on user-user or item-item similarity.
- Takes care of strict and lenient raters.
- Demographics to deal with a new user problem
- For any negative or zero predictions we are replacing it with its baseline estimator.

Cons

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

Singular Value Decomposition

Assumptions

- Matrix Decomposition is perfect i.e. there is no error due to floating-point computation.
- Retaining 90% of energy allows us to retain enough information to approximately reconstruct the original matrix.

Formulation

$$A_{[m\times n]} = U_{[m\times r]}\Sigma_{[r\times r]}(V_{[n\times r]})^T$$

A : Input data matrix (m x n matrix)
U : Left singular matrix (m x r matrix)
Σ : Singular values (r x r diagonal matrix (r: rank of matrix))
V : Right singular vectors (n x r matrix)

$$A \simeq U\Sigma V^T = \Sigma_i \sigma_i u_i \circ v_i^T$$

σ_i => scalar
u_i => vector
v_i => vector

Rows of V^T are eigenvectors of $D^T.D$ = basis function
Σ is diagonal, $\delta_{ii} = \sqrt{\lambda_i}$ (i^{th} eigenvalue)
Rows of U are coefficients for basis function in V
(here we assumed m>n, and rank(D) = n)

Pros and Cons

Pros

- Discover hidden correlations topics: Correlation in two dimensions in 2 matrices gives us less information
- Remove redundant and noisy features: Features that are highly correlated to some other feature or feature that are giving noisy data can be removed by reducing the dimensions of the sigma matrix.
- Interpretation and visualization: The new space between user and movie can give much more insight than the previous space.
- Optical low-rank approximation in terms of Frobenius norm.
- Easier Storage and processing of the data.

Cons

- Interpretability Problem: A singular vector specifies a linear combination of all input columns or rows.
- Lack of Sparsity: Singular Matrices are dense.

CUR Decomposition

Assumptions

- We can approximately reconstruct the original matrix multiplying selected columns matrix and selected rows matrix and Pseudo inverse of SVD of the intersection of the former two matrices.
- We need to eliminate the last 1% of the eigenvalues to avoid predicted values beyond our range.

Formulation

$$||A - CUR||_F \leq ||A - A_k||_F + \varepsilon||A||_F$$

Where A is Original Matrix CUR is Matrix obtained by CUR multiplication A_k is a matrix obtained by retaining k dimensions

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

Here the selected block represent the Frobenius norm of the entire matrix

$$C_d(:,i) = A(:,j) / \sqrt{cP(j)}$$

$$W = XZY^T$$

$$U = W^+ = YZ^+X^T$$

Pros and Cons

Pros

- Easy Interpretation: Since the basis vectors are actual columns and rows.
- Sparse Basis: Since the basis vectors are actual columns and rows.

Cons

- 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 (user-user)	2.108244269195342	0.5827909270216962	0.9999997333183634	654.0901770591736
Collaborative (item-item)	2.0142462779806065	0.5988947062245492	0.9999997565687134	311.151978969574
Collaborative along with Baseline approach *(user-user)	0.9900917714069557	0.632154384077461	0.9999999411830964	314.19730019569397
Collaborative along with Baseline	0.9231188427785346	0.6234511343804537	0.9999999488710957	285.0583381652832

approach *(item-item)				
SVD	1.03358745 89243765	0.5505881378 396289	0.9999999998 398202	431.06211519 241333
SVD with 90% retained energy **	1.03358745 89243765	0.5505881378 396289	0.9999999998 398202	212.14293599 128723
CUR	1.03200531 65490173	0.7161668875 635079	0.9999999998 403102	40.230790853 500366
CUR with 90% retained energy **	1.03307413 33730147	0.7051220455 047493	0.9999999998 399793	31.915229082 107544

Packages/Library Used

Every module is coded in python. Following packages or library are used

1. Numpy
2. Scipy CSR Sparse Matrix
3. Sklearn Metrics
4. Math
5. Panda