Design Document

CS F469 IR ASSIGNMENT - 2

# Recommender Systems

# - Collaborative Filtering (with and without baseline)

# - SVD (100% and 90% energy)

# - CUR (100% and 90% energy)

…………………………………………………………………………………………………………………………………………………………….

2016A7PS0006H R MONITH SOURYA

2016A7PS0033H SANJAY DEVPRASAD

2016A7PS0049H NIKHIL SREEKUMAR NAIR

2016A7PS0086H N VISHNU TEJA

# Dataset

We used the movielens 20m (1000 users) and 150k ratings. Eigenvalues = 996 (Reference for calculation of errors and time taken to predict)

# Assumptions

* **Neighbours considered in collaborative filtering:**

After running empirical tests with multiple k values, all being multiples of 15, it was observed that 60 yielded the most optimal solution. Thus k = 60 was chosen to be the final nearest neighbours valuation.

* **Rows and columns in CUR:**

With reference to class slides, rank of user rating matrix is calculated after which, 4 times the value is taken to select the rows and columns.

# Packages used

* **numpy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
* **scipy.sparse:** a better alternative to employing to sparse matrices than what numpy has to offer.
  + **csc\_matrix, csr\_matrix:** sparse representation of matrices can be done in either of two ways – by leading column indices (csc\_matrix) or leading row indices (csr\_matrix)
  + **linalg:** helps employ mathematical and matrix operations that correspond to eigen value or simply any general matrix application scenario.
* **pandas, csv:** data analysis and manipulation
* **sklearn.metrics:** imported mean square error for root mean square error calculations wrt collaborative, SVD, and CUR recommended models.
* **math:** for calculating basic mathematical expressions. In our models, we extensively used the math module to enable smoother square root functionality, especially in the case of the calculation of RMSE.

# Formulas

* **Splitting into train and test:**

The original dataset is intended to be split into a *train* dataset and a *test* set. 10% non-zero values are chosen at random in each row, and turned to zero. Due to the nature of our dataset, every occurrence of zero can be safely treated as an empty entry due to the lowest rating by any user being 0.5.

The models are run through this *train* dataset and appropriate predictions are made. These are then compared with those in *test* (which comprises only those values that were omitted from *train*).

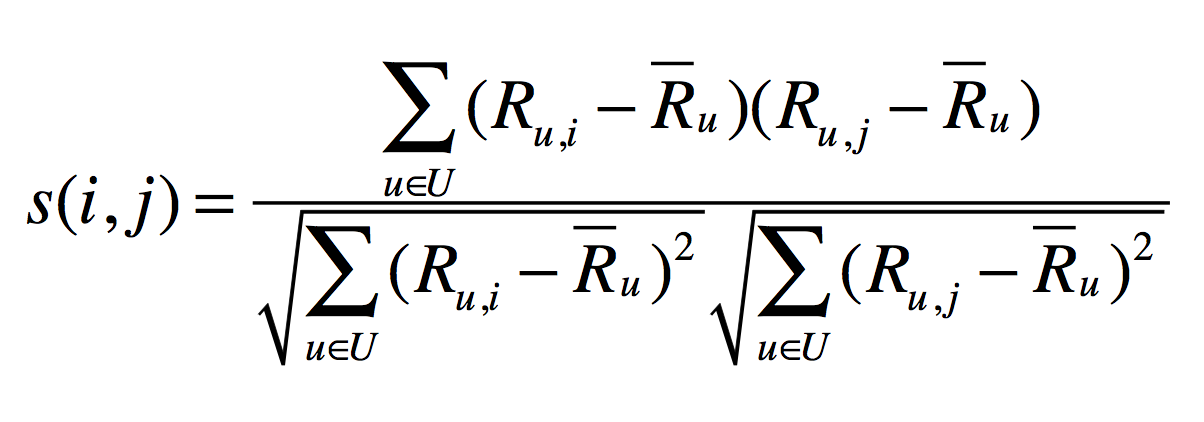
Appropriate error calculations follow once the model predicts using the *train* set.

* **Collaborative:**

After the train/test split, the user vectors are mean adjusted to account for generous and strict (at rating) users.

Similarity between users and items are calculated by employing cosine similarity.

In the case of baseline, we first define similarity between items (s(i,j) between items i and j) using Pearson coefficient.



Then, we select the k nearest neighbours N(i; x) after running empirical tests on multiple possible k values and finalising on an optimal one.

Rating is estimated as the weighted average:

where

= baseline estimate for

μ = overall mean movie rating

= rating deviation of user x = (avg. rating of user x) – μ

= rating deviation of movie i = (avg. rating of movie i) – μ

The weighted average of the ratings is taken on the basis of similarity scores, after taking into consideration the nearest neighbours, and that yields the respective prediction for that case.

* **Singular Value Decomposition (SVD) and CUR:**

After the train/test split, the user vectors are mean adjusted to account for generous and strict (at rating) users. This is achieved by calculating the user bias, which is simply the row average over all non-zero elements in every user vector.

After obtaining matrices U, V, and S, query matrices are multiplied to SVT. A two-dimensional matrix of all such user values ideally yields the entire prediction set.

For 90% energy, the sum of squares of the matrix S are used to compute the extent of dimensionality reduction.

CUR follows slightly similar steps. In the case of CUR, the C and R matrices are drawn up using random distribution of the columns and rows of the user ratings matrix. Then matrix W is found, which is the intersection of C and R. Matrix U is the Moore-Penrose pseudoinverse of matrix W. Matrices C, U, and R are analogous to U, S, and V in SVD.

* **Error calculations:**

**RMSE** – Standard error calculation between Predicted values and Actual Values.

**Precision on top k –** default k value is taken as 25 and actual rating above 3.5 is taken as relevant item to the user.

**Spearman rank correlation –** The rank of the predictions are calculated for every user and subtracted with actual rank of the predicted movie from test dataset and squared. This value is then substituted in the formula to get the correlation. It is always less than 1.

**Time –** It was calculated after loading matrix into RAM till output is printed.

# Observations, inferences and other findings: (150,000 movie ratings)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Recommender System Technique** | **RMSE** | **Precision on top K(25)** | **Spearman Rank Correlation** | **Time taken for prediction** |
| Collaborative | 2.77 | 0.92 | 0.992 | 1.28/ per value (347 sec) |
| Collaborative along with Baseline | 1.23 | 0.93 | 0.993 | 2.69/ per value (200 sec) |
| SVD | 0.96 | 0.96 | 0.993 | 81s |
| SVD with 90% retained energy | 1.04 | 0.92 | 0.992 | 53s |
| CUR | 0.992 | 0.94 | 0.99 | 42s |
| CUR with 90% retained energy | 0.9939 | 0.96 | 0.991 | 33s |