CS F469 INFORMATION RETRIEVAL

Design Document

Assignment 2: Recommender System

Team Members:

1. Aditya Desai: 2015A3PS0211H
2. Snigdha Grandhi: 2015A8PS0495H
3. Aayush Barathwal: 2015A7PS0136H
4. Meghna Vasudeva: 2015B3A70664H

- **Aim**:
  To implement and compare various techniques of construction of Recommender Systems. The techniques implemented are:
  -Collaborative filtering
  -Collaborative filtering along with baseline
  -SVD
  -SVD with 90% retained energy
  -CUR
  -CUR with 90% retained energy

- **Design Overview**:
  - Problem Description:
    To apply several techniques of recommender system in a user-movie rating matrix to further predict the ratings and then make a comparison.
  - Technologies used:
    - Programming Language: Python 3.5.2
    - Data is contained is '.csv' files
  - Dataset:
    MovieLens 100K dataset is used which consists of 100,000 ratings from 1000 users on 1700 movies. The format of the data is <userID, movieID, rating(on the scale of 1 to 5), timestamp>. We then pre-processed the data in MS Excel and got rid of the timestamp and sorted the data is ascending order of <userID,movieID> tuple.
  - Application Operation:
    - **Collaborative Filtering**:
      Collaborative filtering, also referred to as social filtering, filters 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:
      - Assumes that similar users will like similar items. Does not take into account the diverse variety a person might have.
    - Pros:
      - Can give reasonably accurate predictions.
    - Cons:
      - Does not take into account the fact that some users are "hard " raters and some are "soft" raters.

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

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

- **Collaborative along with baseline approach:**
  This technique is same as collaborative filtering except that the ratings are modified according to a baseline approach. The ratings are normalized by adjusting with global average of all ratings and then adjusting with the local average rating of the user. Further, the same steps of collaborative filtering are carried out.
- Assumptions:
  - Assumes that similar users will like similar items. Does not take into account the diverse variety a person might have.

- Pros :
  - Takes into account the fact that some users are “hard ” raters and some are “soft” raters.
- Cons :

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

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

- **SVD**:
  SVD will decompose the initial user-movie matrix into 3 matrices for dimensionality reduction. The 3 matrices are calculated on the fundamentals of eigenvalues and eigenvectors of the original matrix.
  X denotes the utility matrix, and U is a left singular matrix, representing the relationship between users and latent factors**.** S is a diagonal matrix describing the strength of each latent factor, while V transpose is a right

singular matrix, indicating the similarity between items and latent factors.

- Assumptions:
  - Assumes that there is a low rank approximation possible for transforming the utility space into concept space.
- Pros :
  - Converts the points into concepts which may represent the data more accurately due to reduction in dimensionality
  - Optimal low-rank approximation in terms of Frobenius norm.
- Cons :
  - Is computationally expensive to calculate the svd of large and sparse matrices.
  - Takes O(m*n^3)
  - Sparse matrices waste a lot of memory.
  - Interpretability problem; A singular vector specifies a linear combination of all input columns and rows.

Goal: Minimize the sum of reconstruction errors:

$$\sum_{i=1}^{N}\sum_{j=1}^{D}\|x_{ij} - z_{ij}\|^2$$

- where x_{ij} are the "old" and z_{ij} are the "new" coordinates

- **SVD with 90% retained energy**:

  This technique follows the same steps as SVD but takes only some eigenvalues into account in the sigma matrix. The smallest of these are removed and respective changes are made into U and V so that the dot product doesn't face problems. The removal is on the basis that 90% of energy should be retained, ie sum of the squares of the remaining eigenvalues should be 90% of original sum of squares of every eigenvalues. Further, same steps of SVD are followed.
- Assumptions:
  - Assumes that there is a low rank approximation possible for transforming the utility space into concept space.

- Pros :
    - Converts the points into concepts which may represent the data more accurately due to reduction in dimensionality

- Cons :
    - Is computationally expensive to calculate the svd of large and sparse matrices.
    - Takes O(m*n^3)
    - Sparse matrices waste a lot of memory

- **CUR:**
  - **Let:**

    $\mathbf{A_k}$ be the "best" rank ***k*** approximation to **A** (that is, $\mathbf{A_k}$ is SVD of A)

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

    Theorem [Drineas et al.]

    CUR in O(**m·n**) time achieves

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

    with probability at least **1-δ**, by picking

    - $O(k \log(1/\delta)/\varepsilon^2)$ columns, and
    - $O(k^2 \log^3(1/\delta)/\varepsilon^6)$ rows

    In practice: Pick 4*k* cols/rows

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

- Assumptions:
  - Assumes that there is a low rank approximation possible for transforming the utility space into concept space.

- Pros:
  - Easy Interpretation, Since the basis vectors are actual columns and rows.
  - Computationally much more efficient than vanilla SVD
  - Takes O(mn)

        - Sparse Basis: Since the basis vectors are actual columns and rows
    - Cons:
        - May not be very accurate due to random error introduced due to sampling of rows and columns.
        - Duplicate rows and columns; Columns of large norms will be sampled many times.

    - **CUR with 90% retained energy:**
      It is similar to CUR. We have to take 90% energy of the SVD decomposition of W in the process of getting the CUR matrix, where W is the intersection of the rows and columns selected.
      <u>Assumptions</u>:
        - Assumes that there is a low rank approximation possible for transforming the utility space into concept space.

    - <u>Pros:</u>
        - Easy Interpretation, Since the basis vectors are actual columns and rows.
        - Computationally much more efficient than vanilla SVD
        - Takes O(mn)
        - Sparse Basis: Since the basis vectors are actual columns and rows
    - <u>Cons</u>:
        - May not be very accurate due to random error introduced due to sampling of rows and columns.
        - Duplicate rows and columns; Columns of large norms will be sampled many times.

- **<u>Comparison Table:</u>**

Recommender System Technique	Root Mean Square Error	Precision on top K	Spearman Rank Correlation	Time taken for prediction
Collaborative	0.9323	0.225	0.999	7.918e-05
Collaborative along with baseline	0.914	0.232	0.999	8.807e-05
SVD	0.921	0.120	0.999	0.00013
SVD with 90% retained energy	0.915	0.137	0.999	0.000136
CUR	0.833	0.14	0.999	8.209e-05
CUR with 90% retained energy	0.831	0.154	0.99982	8.116e-05

- **Handling generous raters and strict raters:**
  Some raters are generous, like they may give a rating of 3 even to a bad film. Whereas, some raters might be strict, like they may give only 3 to a good film. To tackle this problem, we subtracted the average rating of a user needs to be subtracted from the real rating to make the rating ground even. We did this in the beginning itself after sorting to reduce the bias in raters and an unbiased rating matrix is produced for predicting other ratings.

- **Packages used**:
  - Numpy
    - NumPy is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
  - Pandas
    - Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools.
  - Math

        - It provides access to the mathematical functions defined by the C standard.

- Time
    - This module provides various time-related functions.

- Norm
    - Matrix or vector norm.

- Dot
    - Dot product of two arrays.