CSE891 Mini Project 1: Recommender System

This is the first of two mini projects in this class. Each mini project accounts for 10% fo your final grade. The project must be done individually, without collaboration with any other students. The project requires implementing and evaluating various methods for a joke recommender system. You must implement your methods in Python using the provided template program. You are prohibited from using any Python libraries for recommender systems (e.g., Python Surprise package) to do this project. You're expected to complete the project using numpy and the standard functions in Python. Please check with the instructor/TA first if you want to use other packages besides those provided by numpy. The project due date is Sunday, Oct 20, 2019 (before midnight).

1 Project Overview

Recommender system (RS) is an automated system designed to help provide meaningful suggestions to consumers by filtering out irrelevant information. RS is a key technology that drives the booming of e-commerce and online shopping over the past thirty years. RSs collect information about the preferences of consumers for a set of items (e.g., movies, songs, books, jokes) and recommend new items based on the previous choices of the consumers. In this project you will implement and evaluate three different methods for making item recommendations.

The dataset for this project is a subset of the Jester Online dataset for joke recommendation. The original dataset contains 4.1 Million continuous-valued ratings (between -10.00 to +10.00) of 100 jokes from 73,421 users. The data was collected between April 1999 - May 2003. You can access to the original dataset from following link: `http://goldberg.berkeley.edu/jester-data/`.

2 Tasks

The mini-project consists of 3 programming tasks to be completed:

1. Data loading and creation.
2. Recommendation.
3. Evaluation.

2.1 Task 1: Data Loading and Creation

The first task consists of the following steps:

1. Open and load the input data provided on the class webpage on D2L.
2. Create a ratings matrix R, where R_{ij} denotes the rating of user i for item j.
3. Create a 0/1 training mask matrix M, where M_{ij} is 1 if the user-item pair (i, j) belongs to the training set, and 0 otherwise.
4. Create a 0/1 test mask matrix T, where $T_{ij} = 1$ if the user-item pair (i, j) belongs to the test set, and 0 otherwise.

2.2 Task 2: Recommendation

The second task requires you to implement the following 3 approaches for recommender systems. Note that your computations should be performed on the training data only. The ratings given in the test data must not be used.

Approach 1 (Mean Imputation): Replace any missing ratings with the average ratings of the item in the training data. Specifically, the average rating for item j is given by

$$\overline{R}_j = \frac{\sum_u R_{uj} M_{uj}}{\sum_u M_{uj}}$$

Approach 2 (Matrix Completion): This method employs the singular value thresholding approach for imputing missing values in a matrix [1]. The approach assumes that the ratings matrix R is a product of two low rank matrices (latent factors), and thus, its missing entries can be inferred by learning the latent factors. The latent factors are obtained by solving the following optimization problem

$$\min ||\hat{R}||_* \quad \text{s.t } M_{ij} R_{ij} = M_{ij} R_{ij}$$

For implementation details and algorithm, refer to the pseudocode given in Section 5 of [1].

Approach 3 (Matrix Factorization): This method uses an alternating least-square (ALS) algorithm to decompose the ratings matrix R into a product of two matrices U and V. Specifically, we aim to minimize:

$$min_{U,V} ||(R - UV^\top) \odot M||_F^2 + \lambda_1 ||U||_F^2 + \lambda_2 ||V||_F^2$$

where $\odot$ denotes a Hadamard product operator and λ's are hyperparameters for sparsity regularization. To solve the above optimization problem you can do the following steps:

- Randomly intialize U and V
- Repeat until convergence:
  - Fix V and update U:
    - For $i = 1, \ldots, n$:

$$u_i^T = (\sum_{\{\forall j : M_{ij}=1\}} R_{ij} v_j^T)[\sum_{\{\forall j : M_{ij}=1\}} v_j v_j^T + \lambda_1 I]^{-1}$$

  - Fix U and update V:
    - For $j = 1, \ldots, m$

$$v_j^T = (\sum_{\{\forall i : M_{ij}=1\}} R_{ij} u_i^T)[\sum_{\{\forall i : M_{ij}=1\}} u_i u_i^T + \lambda_2 I]^{-1}$$

  - Compute $\hat{R} = (R \odot M) + (UV^\top \odot (1 - M))$

Where u_i^T be the i-th row of matrix U and v_j^T is the j-th row of matrix V(which means v_j^T is the j-th column of matrix V^T). For example if the dimension of R is 2000×100, U is 2000×10 and V is 100×10, then u_i^T is a row vector of length 10, v_j is a column vector of length 10 and v_j^T is a row vector of length 10. Furthermore both $\sum_{\{\forall j : M_{ij}=1\}} v_j v_j^T + \lambda_1 I$ and $\sum_{\{\forall i : M_{ij}=1\}} u_i u_i^T + \lambda_2 I$ are 10×10 matrix.

2.3 Task 3: Evaluation

The final task is to implement a Python function to compute the root mean square error (RMSE) of the predicted ratings $\hat{R}$ of the test data. Specifically, the formula for RMSE is given by:

$$RMSE = \sqrt{\frac{\sum_{ij} T_{ij}(R_{ij} - \hat{R}_{ij})^2}{\sum_{ij} T_{ij}}}$$

References

[1] J. Cai, E. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.