Recommender Systems

Recommender systems are widely used in many industrial companies, especially location-based service such as Yelp [1]. Yelp provides a very big dataset for research purpose (See the Kaggle page https://www.kaggle.com/yelp-dataset/yelp-dataset). In this task, we will explore this dataset using the learned recommendation algorithms. To make the task feasible on most of the laptops and PCs, we have extracted a managable subset of the datasets, which contains the reviews on businesses in Las Vegas. The data format for users and businesses are exactly the same as in the description on the Kaggle page. The review file only preserves the review id, business id, user id, stars (ratings) and date to keep the dataset small.

Preprocessing: Split the review data into training and test datasets. Use the 20% **latest reviews** (with largest "date" attribute) for testing and the remaining 80% for training. The collaborative filtering algorithm does not need to train, but the similarity computation can only conduct using ratings in the training set.

The tasks are specified as follow:

1. **Item-based collaborative filtering**: In this task, we will build an item-based CF algorithm on Yelp.

   (a) Our first step is to pre-compute the similarity between each pair of businesses (items) on the training data, such that we don't have to compute the similarities on-the-fly. Here, we use cosine similarity for simplicity. **Report (1) the time for computing all pairwise similarities, and (2) report the similarities between the following pairs of businesses.**

Pairs	Business 1	Business 2
1	rjZ0L-P1SRLJfiK5epnjYg	cL2mykvopw-kodSyc-z5jA
2	6H8xfhoZ2IGa3eNiY5FqLA	XZbuPXdyA0ZtTu3AzqtQhg
3	rfwJFFzW6xW2qYfJh14OTA	G58YATMKnn-M-RUDWg3lxw
4	0QSnurP5Ibor2zepJmEIlw	6-lmL3sC-axuh8y1SPSiqg

   (b) Try to compute the above cosine similarity using matrix multiplication. **Hints:** Let $\mathbf{R}$ be the $m \times n$ rating matrix, where m and n are the number of users and number of businesses, respectively. $\mathbf{R}^T\,\mathbf{R}$ gives the pairwise dot product of business vectors. Also notice that the diagonal of $\mathbf{R}^T\,\mathbf{R}$ gives the self-dot product of each business, which is the square of the L2-norm of each business vector used in the denominator of the cosine similarity! **Report the time for computing the whole pairwise similarity matrix.**

(c) Implement the item-based collaborative filtering algorithm based on the above similarity matrix. Use top-20 similar businesses rated by the user to predict the ratings. **Report the RMSE.**

(d) Extend the above CF method by incorporating the bias from all the transactions (b_g), bias from each user u (b_u = [avg rating of user u] - b_g), and the bias from each business i (b_i = [avg rating of user i] - b_g), the recommendation score of user u to business i is r **Report the RMSE.**

2. **Latent factor model**: Hold-out 1/8 of the latest reviews in the training data as validation set to tune the number of latent factors. This results in a data split of 70% training, 10% validation, and 20% testing.

(a) Implement a latent factor model with number of latent factors k. Use stochastic gradient descent with fixed learning rate $\eta = 0.01$ and regularization hyper-parameter $\lambda_1 = \lambda_2 = 0.3$. Run SGD for 20 epochs and **report RMSE for** $k = \{8, 16, 32, 64\}$ **on both training (i.e., the minimized loss) and validation sets.**

(b) Extend the latent factor model with bias. Use stochastic gradient descent with fixed learning rate $\eta = 0.01$ and regularization hyperparameter $\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = 0.3$. Run SGD for 20 epochs and **report RMSE for** $k = \{8, 16, 32, 64\}$ **on both training and validation sets.**

(c) Select the best k for each of the above two versions of latent factor models, and apply on the test dataset. **Report RMSE of the two models on test dataset.**

Note: You could use any python package to **validate** your implementation, to compute the pairwise cosine similarity, or to do the stochastic gradient descent. However, directly calling any package of CF and latent factor models will get zero mark