ECS 271 Assignment 1

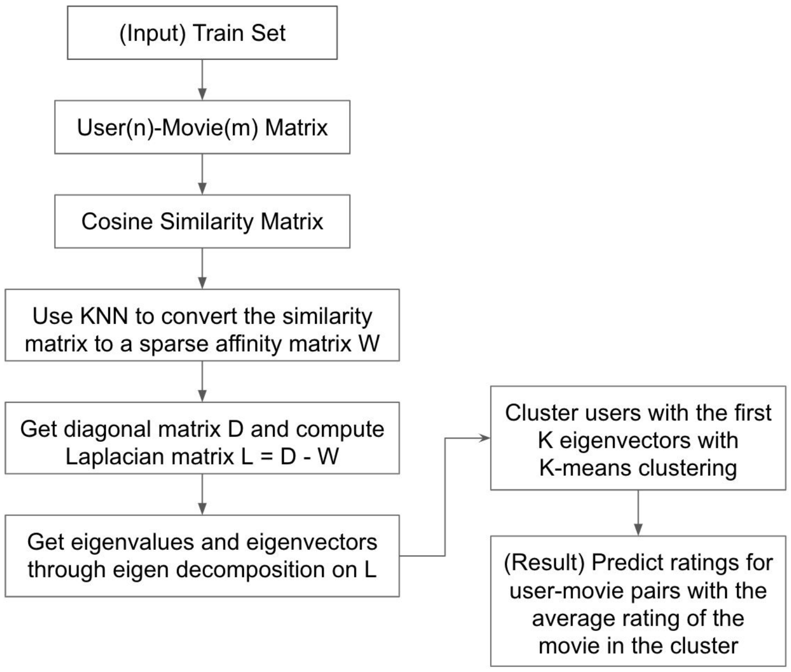
Unsupervised learning-Netflix

Haitong Li

1. Spectral Clustering:
   a. Reasoning:

      Spectral clustering is a clustering method based on graph theory. We will have a proper way to measure the distances between each node in the graph, assign weights to the edges and construct a Laplacian matrix and work with the eigenvalues and eigenvectors to assign labels to the nodes and form clusters. In the Netflix problem, we are going to use this method to identify clusters of users and then put the average rating from all the ratings to the movie from users in the same cluster as the predicted rating for a particular user-movie pair.

   b. Summary of approach:


(Input) Train Set
User(n)-Movie(m) Matrix
Cosine Similarity Matrix
Use KNN to convert the similarity matrix to a sparse affinity matrix W
Get diagonal matrix D and compute Laplacian matrix L = D - W
Get eigenvalues and eigenvectors through eigen decomposition on L
Cluster users with the first K eigenvectors with K-means clustering
(Result) Predict ratings for user-movie pairs with the average rating of the movie in the cluster

c. Method:

The first part of my approach is to define clusters of users with spectral clustering on the train dataset. After reading in the train set, I converted it to a user-movie matrix where the rows are the users, the columns are the movies, and the entries are the ratings for a specific user-movie pair. Since it was a sparse matrix, I filled the NaN cells with the mean rating for each user to produce better cluster assignments. I constructed the similarity matrix on user nodes with cosine_similarity() from sklearn.metrics.pairwise. Since it is a large dataset, the similarity measurement resulted in a relatively dense matrix which may interrupt the following Laplacian calculation. To minimize the noise, I used the KNN method, which only connects two nodes with an edge if they are within the K nearest neighbors of each other (K=25 in the implementation), to convert the cosine similarity matrix into a sparse affinity matrix W. The next step was to calculate the Laplacian matrix L = D (diagonal matrix of W) - W with numpy.linalg.eig() and get eigenvalues e and eigenvectors v. With the k (k=20 for the k-means clustering) eigenvectors selected, user clusters were identified through k-means clustering (sklearn.cluster.KMeans()).

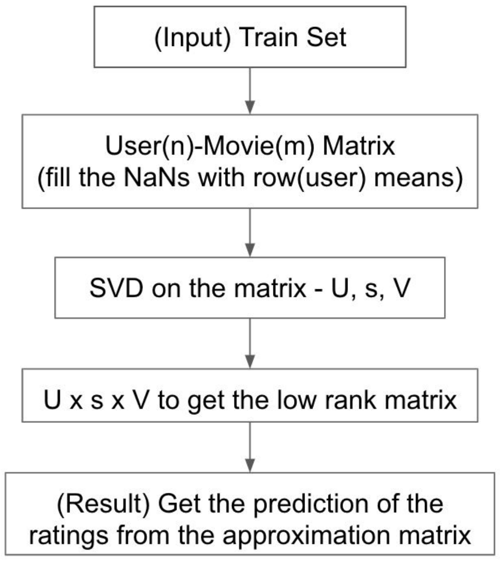
The second part was to make rating predictions for the test set. I read in the test set file and for each user-movie pair, assigned the average rating to the movie from all the users in the same cluster as the current user as the predicted rating. The results were saved as an output file.

For tuning the hyper-parameters, I used cross validation which separated the train set into 1-n:n datasets (n being the ratio of data held for validation, n=0.1 and 0.2 in implementation), in which the 1-n sized set was the train set and the n sized set was the test set. Then the predictions were made in the same process as above, and the mean square errors (between the actual rating value for the test set and the predicted value) and correct rates were computed for k (for k-means) = 1-30, and k=20 was chosen as a relatively stable parameter for k-means. The same validation went through for the K for KNN graph and K=25 was chosen.

2. Matrix Completion:
    a. Reasoning:

       The matrix completion method factors the input matrix into two or more matrices representing the latent features of different types of items in the input. We will choose k latent features of each kind of entity, and the multiplication of the chosen parts of the matrices will result in a lower rank approximation of the original matrix. In the Netflix problem, we are trying to investigate the latent features of the users and the movies, which interact and determine how a user will rate a movie. Basically, we are going to fill in the blanks in the original matrix with the approximation made by multiplying the latent features matrices.

    b. Summary of approach:


(Input) Train Set
User(n)-Movie(m) Matrix
(fill the NaNs with row(user) means)
SVD on the matrix - U, s, V
U x s x V to get the low rank matrix
(Result) Get the prediction of the ratings from the approximation matrix

c. Method:

First, I converted the train set to a user-movie matrix. In order to facilitate the singular value decomposition, I filled in the NaN values with the row (user) means. Then I used scipy.sparse.linalg.svds() to do the SVD with k=20 on the matrix and got the user feature matrix U, s, and movie feature matrix V. Next, I multiplied U, s, and V to get the low rank approximation matrix. To get the prediction results, for each user-movie pair in the test set, I queried on the low rank matrix and assigned the corresponding value as the predicted rating.

To tune the hyper-parameter (k), I did cross validation similar to that for the spectral clustering method and calculated the mean squared errors for different k settings (tested 5-40 and 50, 60, 70), and found k=20 to be the most stable and best performing parameter for the SVD.

3. Difference between the results from the two methods:

As I have observed, since the results from spectral clustering are highly dependent on the input data and the cluster relationship, the predicted ratings for the same cluster will be all the same, and if the cluster distribution is skewed (too many people fall into the same group), the predicted value will be very close to the average of all ratings, which may not be very meaningful. Another point is that the predictions made with spectral clustering do not take the different characteristics of the users into account but only consider how similar they are, and it may be the reason why this method cannot fit into large scaled data and is not very accurate.

In my opinion, the matrix completion method is more reasonable for this particular problem. Since we are predicting the ratings for a defined set of users and movies, it is natural to analyze the characteristics of the users and different types of movies and then to make predictions accordingly. This method also gives better performance, as it can be observed in the validation that the mean squared error is overall lower than the results from spectral clustering.