RECOMMENDER SYSTEMS

GET DATASETS FROM HERE: https://github.com/dipsankarb/recolab-data.git

1. In this problem the goal is to build a recommender systems using latent features. We are going to use Stochastic Gradient Descent to build the Latent Factor Recommendation Systems which can be in turn used to recommend movies to users.

   We are going to use the recommendation matrix R where each entry R_{iu} represents the rating given by user i to movie u. Size is m x n where m is the movies and n is the users. As already discussed, most of the elements in the matrix are unknown given that users sparsely rate the movies.

   We shall compute to find out two matrices P and Q such that R ≈ QP^T. Where the dimensions of Q is m x k and that of P are n x k. k is a parameter of the algorithm

   Error is given as:

$$E = \sum_{(i,u)\in ratings} (R_{iu} - q_i . p_u^T)^{\ 2} + \lambda \left[\sum_u \|p_u\|_2^2 + \sum_i \|q_i\|_2^2 \right]$$

   The $\sum_{(i,u)\in ratings}$ essentially means that we sum the pairs for which user has provided a rating that is the (i,u) entries are known. Qi denotes the Ith row of the matrix R and p_u the uth row for uth user. λ is the regularization parameter and || . || is the L_2 norm and $\|p_u\|_2^2$ is the square of the L_2 norm meaning the sum of the squares of the elements of p_u.

   Now, if ε_{iu} denotes the derivative of the error E with respect to R_{iu}, then ε_{iu} is given by:

$$\varepsilon_{iu} = 2 * (R_{iu} - q_i . p_u^T)$$

$$q_i := q_i + \eta * (\varepsilon_{iu} * p_u - 2 * \lambda * q_i)$$
$$p_u := p_u + \eta * (\varepsilon_{iu} * q_i - 2 * \lambda * p_u)$$

   Now, implement the algorithm and read every entry of R from disk, and update ε_{iu}, q_i, and p_u for every entry. Iterate until q_i and p_u stops changing. You can choose k=20, λ as 0.1 and the number of iterations at 40. Start with η=0.1 and try to find a good learning rate. The error E on the ratings dataset train.txt should be less than 65000 after 40 iterations. In this process you might encounter η to be too high causing the error function to converge at a high value or η to be too small causing the error function not having enough iterations to reduce to the desired value in 40 iterations. So choose η wisely.

Given:

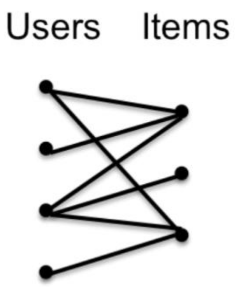
train.txt : This consists of the R matrix.
test.txt : This is the test data that you should use to test your solution. It essentially consists of the original matrix with some columns removed.

Submit:

Filled latent.py file with your code (need not be Spark implementation).
A plot (in a pdf) showing how E varies with the different iterations and shows its convergence.
State the value of η found.

2. Collaborative filtering:

Consider a user-item bipartite graph as shown below. An edge from user U to item I denotes that the user likes the item in I. We also represent ratings provided by the users using the matrix R where each row represents a user and each columns is an item. If user i likes item j then $R_{i,j}$=1 else 0. Assume we have m users and n items so R is m x n. Now, let us define a matrix P m xm which is a diagonal matrix and each diagonal item denotes the degree of every user which means the number of items user i likes. Similarly, matrix Q denotes a diagonal matrix for item where an entry denotes how many users liked item i.



$$R=\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P=\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Q=\begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

For the user-user case, we can define a recommendation matrix $\Gamma(i,j) = r_{i,j}$ where Γ can be defined as (please derive if you want to):

$$\Gamma = P^{-1/2}.R.R^T.P^{-1/2}.R$$

For the item-item case, it can be given as:

$$\Gamma = R.Q^{-1/2}.R^T.R.Q^{-1/2}$$

Now, from the given files we have:

ratings.txt: This is the ratings matrix R.

items.txt : This is the files containing titles of TV shows, in same order as of R given in the previous file.

You have to compute the user-user and item-item collaborative filtering recommendations for the 500th user of the system. Let's call him Mr. A. Now for doing that, we have deleted the first 100 rows of A's entries in R and replaced with 0s. The original viewings of Mr. A is given in another file orig.txt. We will now recommend 100 shows to A based on the other shows and see if it matches the ones that he actually watched.

Submit: collaborative.py with code for following:

1. Compute matrices P and Q.

2. Use user-user collaborative filtering (in a distinct function) to find first 100 set of shows that can be recommended to A. Display the top 5 with highest similarity (display the names from the files items.txt and don't show index of the show). Also show the scores. In case of a tie, choose the one with the lower score.

3. Repeat the above in another distinct function to do item-item collaborative filtering.