3 Latent Features for Recommendations (35 points)

Note: Please use native Python (Spark not required) to solve this problem. It usually takes several minutes to run, however, time may differ depending on the system you use.

The goal of this problem is to implement the *Stochastic Gradient Descent* algorithm to build a Latent Factor Recommendation system. We can use it to recommend movies to users. We encourage you to read the slides of the lecture "Recommender Systems 2" again before attempting the problem.

Suppose we are given a matrix R of ratings. The element R_{iu} of this matrix corresponds to the rating given by user u to item i. The size of R is $m \times n$, where m is the number of movies, and n the number of users.

Most of the elements of the matrix are unknown because each user can only rate a few movies.

Our goal is to find two matrices P and Q, such that $R \simeq QP^T$. The dimensions of Q are $m \times k$, and the dimensions of P are $n \times k$. k is a parameter of the algorithm.

We define the error as

$$E = \left(\sum_{(i,u) \in \text{ratings}} (R_{iu} - q_i \cdot p_u^T)^2 \right) + \lambda \left[\sum_u \|p_u\|_2^2 + \sum_i \|q_i\|_2^2 \right]. \tag{5}$$

The $\sum_{(i,u)\in\text{ratings}}$ means that we sum only on the pairs (user, item) for which the user has rated the item, *i.e.* the (i, u) entry of the matrix R is known. q_i denotes the i^{th} row of the matrix Q (corresponding to an item), and p_u the u^{th} row of the matrix P (corresponding to a user u). q_i and p_u are both row vectors of size k. λ is the regularization parameter. $\|\cdot\|_2$ is the L_2 norm and $\|p_u\|_2^2$ is square of the L_2 norm, *i.e.,* it is the sum of squares of elements of p_u.

(a) [10 points]

Let ε_{iu} denote the derivative of the error E with respect to R_{iu}. What is the expression for ε_{iu}? What are the update equations for q_i and p_u in the Stochastic Gradient Descent algorithm? Please show your derivation and use ε_{iu} in your final expression of q_i and p_u.

(b) [25 points]

Implement the algorithm. Read each entry of the matrix R from disk and update ε_{iu}, q_i and p_u for each entry.

To emphasize, you are not allowed to store the matrix R in memory. You have to read each element R_{iu} one at a time from disk and apply your update equations (to each element) each iteration. Each iteration of the algorithm will read the whole file.

Choose $k = 20$, $\lambda = 0.1$ and number of iterations $= 40$. Find a good value for the learning rate η, starting with $\eta = 0.1$. (You may not modify k or λ) The error E on the training set ratings.train.txt discussed below should be less than 65000 after 40 iterations; you should observe both q_i and p_u stop changing.

Based on values of η, you may encounter the following cases:

- If η is too big, the error function can converge to a high value or may not monotonically decrease. It can even diverge and make the components of vectors p and q equal to ∞.
- If η is too small, the error function doesn't have time to significantly decrease and reach convergence. So, it can monotonically decrease but not converge *i.e.* it could have a high value after 40 iterations because it has not converged yet.

Use the dataset at `q3/data` within the bundle for this problem. It contains the following files:

- `ratings.train.txt`: This is the matrix R. Each entry is made of a user id, a movie id, and a rating.

Plot the value of the objective function E (defined in equation 5) on the training set as a function of the number of iterations. What value of η did you find?

You can use any programming language to implement this part, but Java, C/C++, and Python are recommended for speed. (In particular, Matlab can be rather slow reading from disk.) It should be possible to get a solution that takes on the order of minutes to run with these languages.

Hint: These hints will help you if you are not sure about how to proceed for certain steps of the algorithm, although you don't have to follow them if you have another method.

- *Initialization of P and Q: We would like q_i and p_u for all users u and items i such that $q_i \cdot p_u^T \in [0, 5]$. A good way to achieve that is to initialize all elements of P and Q to random values in $[0, \sqrt{5/k}]$.*
- *Update the equations: In each update, we update q_i using p_u and p_u using q_i. Compute the new values for q_i and p_u using the old values, and then update the vectors q_i and p_u.*
- *You should compute E at the end of a full iteration of training. Computing E in pieces during the iteration is incorrect since P and Q are still being updated.*

What to submit

(i) Equation for ε_{iu}. Update equations in the Stochastic Gradient Descent algorithm [3(a)]

(ii) Value of η. Plot of E vs. number of iterations. Make sure your graph has a y-axis so that we can read the value of E. Only one plot with your chosen η is required [3(b)]

(iii) Please upload all the code to Gradescope [3(b)]