CSC4008 Assignment 4

Start: Nov 28 at 00:00

End: Dec 18 at 23:59

Correspondence TA/USTF: Yuyang Liang (yuyangliang@link.cuhk.edu.cn)

Important Notes

1. The assignment is an individual project, to be finished on one's own effort.
2. The work must be submitted before the deadline. Late submissions within 2 days will apply 20% penalty. Late submissions after 2 days will not be accepted.
3. Plagiarism is strictly forbidden, regardless of the role in the process. Notably, ten consecutive lines of identical codes are treated as plagiarism. Depending on the seriousness of the plagiarism, 30%-100% marks will be deducted.
4. Please read the document carefully. No argument will be accepted on issues that have been specified clearly in the documents.

1 Implementing k-means on Spark [50pts]

Note: This problem must be implemented in Spark. **Do not** use the Spark MLlib clustering library for this problem. You may store the centroids in memory if you choose to do so.

This problem will help you understand the details of implementing clustering algorithms on Spark. It will also help you understand the impact of using different distance metrics and initialization strategies in practice. Let $\mathcal{X}$ be a set of n data points in d-dimensional space $\mathbb{R}^d$. Given the number of clusters k and the set of k centroids $\mathcal{C}$, we define various distance metrics and their corresponding cost functions that they minimize.

Euclidean distance: Given two points A and B in d-dimensional space, where $A = [a_1, a_2 \cdots a_d]$ and $B = [b_1, b_2 \cdots b_d]$, the Euclidean distance between A and B is defined as:

$$||a - b|| = \sqrt{\sum_{i=1}^{d} ||a_i - b_i||^2}. \tag{1}$$

The corresponding cost function ϕ that is minimized when we assign points to clusters using the Euclidean distance metric is given by:

$$\phi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} ||x - c||^2. \tag{2}$$

Note that the distance value is squared in the cost function. This is intentional, as it is the squared Euclidean distance that the algorithm is guaranteed to minimize.

Manhattan distance: Given two points A and B in d-dimensional space, where $A = [a_1, a_2 \cdots a_d]$ and $B = [b_1, b_2 \cdots b_d]$, the Manhattan distance between A and B is defined as:

$$|a - b| = \sum_{i=1}^{d} |a_i - b_i|. \tag{3}$$

The corresponding cost function ψ that is minimized when we assign points to clusters using the Manhattan distance metric is given by:

$$\psi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} |x - c|. \tag{4}$$

Iterative k-means Algorithm: We learned the basic k-means algorithm in class which is as follows: k centroids are initialized, each point is assigned to the nearest centroid and the centroids are recomputed based on the assignments of points to clusters. In practice, the above steps are run for several iterations. We present the resulting iterative version of k-means in Algorithm 1.

Algorithm 1: Iterative k-means

Input : Dataset $\mathcal{X}$, number of clusters k, maximum number of iterations `MAX_ITER`
Output: Final clusters $\mathcal{C}$

```
Select k points as initial centroids of the k clusters;
for i ← 1 to MAX_ITER do
    for each point p ∈ X do
        Assign p to the cluster with the closest centroid;
    end
    Calculate the cost φ_i or ψ for this iteration;
    for each cluster c ∈ C do
        Recompute the centroid of c as the mean of all the data points assigned to c;
    end
end
```

Iterative k-means clustering on Spark: Implement iterative k-means using Spark. Please use the provided dataset for this problem, which contains three files:

1. `data.txt` contains the dataset which has 4601 rows and 58 columns. Each row is a document represented as a 58 dimensional vector of features. Each component in the vector represents the importance of a word in the document. The ID to download `data.txt` into a Colab is `1E-voIV2ctU4Brw022Na8RHVVRGOoNkO1`
2. `c1.txt` contains k initial cluster centroids. These centroids were chosen by selecting $k = 10$ random points from the input data. The ID to download `c1.txt` into a Colab is `1yXNlZWMqUcAwDScBrkFChOHJwR1FZXmI`

3. `c2.txt` contains initial cluster centroids which are as far apart as possible, using Euclidean distance as the distance metric. (You can do this by choosing 1st centroid c_1 randomly, and then finding the point c_2 that is farthest from c_1, then selecting c_3 which is farthest from c_1 and c_2, i.e., the minimum distance from c_3 to c_1 or c_2 is maximized, and so on). The ID to download `c2.txt` into a Colab is `1vfovle9DgaeK0LnbQTH0j7kRaJjsvLtb`

Tip: To download the datasets in Colab, you can follow the set-up instructions at the beginning of Tutorial 8[1] and replace the ids correspondingly. You can also upload the file from local.

Set number of iterations (`MAX_ITER`) to 20 and number of clusters k to 10 for all the experiments carried out in this question. Your Spark program should ensure that the correct amount of iterations are run.

(a) Exploring initialization strategies with Euclidean distance [25pts (5pts for code)]

1. Using the Euclidean distance (refer to Equation (1)) as the distance measure, compute the cost function $\phi(i)$ (refer to Equation (2)) for every iteration i. This means that, for your first iteration, you'll be computing the cost function using the initial centroids located in one of the two text files. Run the k-means on `data.txt` using `c1.txt` and `c2.txt`. Generate a graph where you plot the cost function $\phi(i)$ as a function of the number of iterations $i = 1..20$ for `c1.txt` and also for `c2.txt`. You may use a single plot or two different plots, whichever you think best answers the theoretical questions we're asking you about. [15pts]

   Hint: Note that you do not need to write a separate Spark job to compute $\phi(i)$. You should be able to calculate costs while partitioning points into clusters.

2. What is the percentage change in cost after 10 iterations of the k-means algorithm when the cluster centroids are initialized using `c1.txt` vs. `c2.txt` and the distance metric being used is Euclidean distance? Is random initialization of k-means using `c1.txt` better than initialization using `c2.txt` in terms of cost $\phi(i)$? Explain your reasoning. [5pts]

   Hint: To be clear, the percentage refers to $(\phi(1) - \phi(11))/\phi(1)$.

(b) Exploring initialization strategies with Manhattan distance [25pts (5pts for code)]

1. Using the Manhattan distance metric (refer to Equation (3)) as the distance measure, compute the cost function $\psi(i)$ (refer to Equation (4)) for every iteration i. This means that, for your first iteration, you'll be computing the cost function using the initial centroids located in one of the two text files. Run the k-means on `data.txt` using `c1.txt` and `c2.txt`. Generate a graph where you plot the cost function $\psi(i)$ as a function of the number of iterations $i = 1..20$ for `c1.txt` and also for `c2.txt`. You may use a single plot or two different plots, whichever you think best answers the theoretical questions we're asking you about. [15pts]

[1] `https://colab.research.google.com/drive/1LBTmcyI9l-aL1zm32PDzs69QmtTeDv1j?usp=sharing`

Hint: This problem can be solved in a similar manner to that of part (a). Also note that It's possible that for Manhattan distance, the cost do not always decrease. k-means only ensures monotonic decrease of cost for squared Euclidean distance. Look up k-medians to learn more.

2. What is the percentage change in cost after 10 iterations of the k-Means algorithm when the cluster centroids are initialized using `c1.txt` vs. `c2.txt` and the distance metric being used is Manhattan distance? Is random initialization of k-means using `c1.txt` better than initialization using `c2.txt` in terms of cost $\psi(i)$? Explain your reasoning. [5pts]

   Hint: You can plot the two curves in the same plot to compare the starting points of two curves to analyze which one is better. `c2.txt` *is generated based on Euclidean distance.*

2 Recommender Systems [50pts]

Note: Please use native Python (Spark not required) to solve this problem. If you run into memory error when doing large matrix operations, please make sure you are using 64-bit Python instead of 32-bit (which has a 4GB memory limit).

Consider a user-item bipartite graph where each edge in the graph between user U to item I, indicates that user U likes item I. We also represent the ratings matrix for this set of users and items as R, where each row in R corresponds to a user and each column corresponds to an item. If user i likes item j, then $R_{i,j} = 1$, otherwise $R_{i,j} = 0$. Also assume we have m users and n items, so matrix R is $m \times n$.

Let's define a matrix P, $m \times m$, as a diagonal matrix whose i-th diagonal element is the degree of user node i, i.e. the number of items that user i likes. Similarly, a matrix Q, $n \times n$, is a diagonal matrix whose i-th diagonal element is the degree of item node i or the number of users that liked item i. See Figure 1 for an example.

(a) [5pts]

Define the non-normalized user similarity matrix $T = R * R^T$ (multiplication of R and transposed R). Explain the meaning of T_{ii} and $T_{ij} (i \neq j)$, in terms of bipartite graph structures (See Figure 1) (e.g. node degrees, path between nodes, etc.).

(b) [10pts]

Let's define the *item similarity matrix*, S_I, $n \times n$, such that the element in row i and column j is the cosine similarity of item i and item j which correspond to column i and column j of the matrix R. Show that $S = Q^{-1/2} R^T R Q^{-1/2}$, where $Q^{-1/2}$ is defined by $Q_{rc}^{-1/2} = 1/\sqrt{Q_{rc}}$ for all nonzero entries of the matrix, and 0 at all other positions.

Repeat the same question for *user similarity matrix*, S_U where the element in row i and column j is the cosine similarity of user i and user j which correspond to row i and row j of the matrix R. That is, your expression for S_U should also be in terms of some combination of R, P, and Q. Your answer should be an operation on the matrices, in particular you should not define each coefficient of S_U individually.

Your answer should show how you derived the expressions.

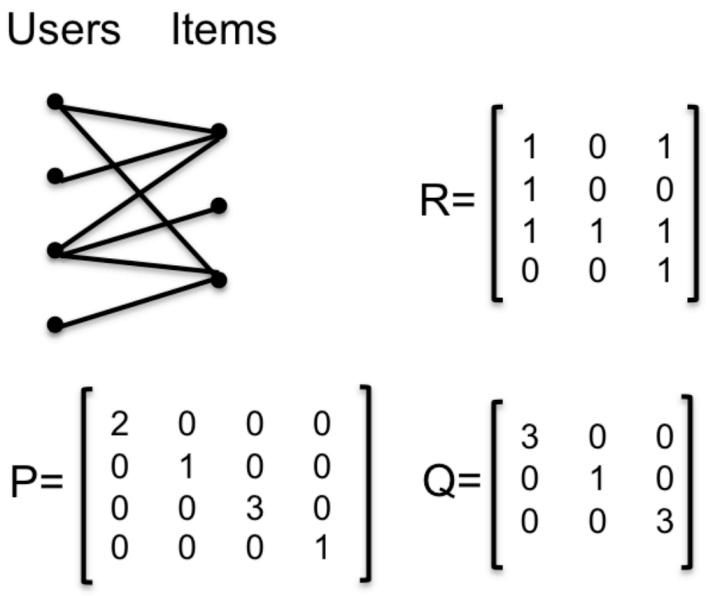

Figure 1: User-item bipartite graph.

Hint: To make the element-wise square root of a matrix, you may write it as matrix to the power of $\frac{1}{2}$.

(c) [10pts]

The recommendation method using user-user collaborative filtering for user u, can be described as follows: for all items s, compute $r_{u,s} = \sum_{x \in U} \text{cos-sim}(x, u) \times R_{xs}$ and recommend the k items for which $r_{u,s}$ is the largest.

Similarly, the recommendation method using item-item collaborative filtering for user u can be described as follows: for all items s, compute $r_{u,s} = \sum_{x \in I} R_{ux} \times \text{cos-sim}(x, s)$ and recommend the k items for which $r_{u,s}$ is the largest.

Let's define the recommendation matrix, Γ, $m \times n$, such that $\Gamma(i, j) = r_{i,j}$. Find Γ for both item-item and user-user collaborative filtering approaches, in terms of R, P and Q. Your final answer should describe operations on matrix level, not specific terms of matrices.

Hint: For the item-item case, $\Gamma = RQ^{-1/2}R^T RQ^{-1/2}$.

Your answer should show how you derived the expressions (even for the item-item case, where we give you the final expression).

(d) [25pts (5pts for code)]

In this question you will apply these methods to a real dataset. The data contains information about TV shows. More precisely, for 9985 users and 563 popular TV shows, we know if a given user watched a given show over a 3 month period.

Use the provided dataset for this problem, which contains two files:

1. `user-shows.txt` This is the ratings matrix R, where each row corresponds to a user

and each column corresponds to a TV show. $R_{ij} = 1$ if user i watched the show j over a period of three months. The columns are separated by a space.

2. `shows.txt` This is a file containing the titles of the TV shows, in the same order as the columns of R.

We will compare the user-user and item-item collaborative filtering recommendations for the 500th user of the dataset. Let's call him Alex. (i.e. with Python's 0-based indexing, Alex=users[499].)

In order to do so, we have erased the first 100 entries of Alex's row in the matrix, and replaced them by 0s. This means that we don't know which of the first 100 shows Alex has watched. Based on Alex's behaviour on the other shows, we will give Alex recommendations on the first 100 shows.

1. Compute the matrices P and Q.
2. Using the formulas found in part (c), compute Γ for the user-user collaborative filtering. Let S denote the set of the first 100 shows (the first 100 columns of the matrix). From all the TV shows in S, which are the five that have the highest similarity scores for Alex? In case of ties of similarity scores between two shows, choose the one with smaller index. Do not write the index of the TV shows, write their names using the file `shows.txt`. [10pts = 5 $\times$ 2pts for each]
3. Compute the matrix Γ for the movie-movie collaborative filtering. From all the TV shows in S, which are the five that have the highest similarity scores for Alex? In case of ties between two shows, choose the one with smaller index. Again, hand in the names of the shows. [10pts = 5 $\times$ 2pts for each]

For sanity check, your highest similarity score for user-user collaborative filtering should be above 900, and your highest similarity score for movie-movie filtering should be above 31.

What to submit

You need to submit the following two files to BlackBoard. Please format your files as "student_id.pdf" and "student_id.py" (or "student_id.ipynb"). For example, if your student id is 123456789, then you should submit "123456789.pdf" and "123456789.py" (or "123456789.ipynb"). You can have several code files for different problems. You can submit several files in one submission. Don't submit them in different submission.

1. A writeup contains
   - A plot of cost vs. iteration for two initialization strategies for 1(a). [15pts]
   - Percentage improvement values and your explanation for 1(a). [5pts]
   - A plot of cost vs. iteration for two initialization strategies for 1(b). [15pts]
   - Percentage improvement values and your explanation for 1(b). [5pts]
   - Interpretation of T_{ii} and T_{ij} for 2(a). [5pts]

- Expression of S_I and S_U in terms of R, P and Q and accompanying explanation for 2(b). [10pts]
- Expression of Γ, for both item-item and user-user cases, in terms of R, P and Q and accompanying explanation for 2(c). [10pts]
- The names of five TV shows that have the highest similarity scores for Alex for the user-user collaborative filtering (no need to report the similarity scores) for 2(d). [10pts]
- The names of five TV shows that have the highest similarity scores for Alex for the item-item collaborative filtering (no need to report the similarity scores) for 2(d). [10pts]

2. Your code for 1. [10pts]
3. Your code for 2(d). [5pts]