A5

Submit Assignment

Due May 24 by 11:59pm **Points** 28 **Submitting** a file upload **File Types** zip

Note: This problem requires substantial computing time. Don't start it at the last minute.

This problem will help you understand the details of implementing clustering algorithm (k-means) on Spark. In addition, this problem will also help you understand the impact of using various distance metrics and initialization strategies in practice.

Let us say we have a set X of n data points in the d-dimensional space R^d. Given the number of clusters k and the set of k centroids C, we now proceed to define various distance metrics and the corresponding cost functions that they minimize.

Euclidean distance

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

$$||A - B|| = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \quad (1)$$

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

$$\phi = \sum_{x \in X} \min_{c \in C} ||x - c||^2 \quad (2)$$

Manhattan distance

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

$$|A - B| = \sum_{i=1}^{d} |a_i - b_i| \quad (3)$$

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

$$\psi = \sum_{x \in X} \min_{c \in C} |x - c| \quad (4)$$

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 Algorithm

1: Select k points as initial centroids of the k clusters.

2: for iterations := 1 to MAX_ITER do

    3: for each point p in the dataset do

        4: Assign point p to the cluster with the closest centroid

    5: end for

    6: for each cluster c do

        7: Recompute the centroid of c as the mean of all the data points assigned to c

    8: end for

9: end for

Iterative k-means clustering on Spark: Implement iterative k-means using Spark. Please use the dataset in **kmeans.zip**.

The zip has 3 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.

2. c1.txt contains k initial cluster centroids. These centroids were chosen by selecting k = 10 random points from the input data.

3. c2.txt contains initial cluster centroids which are as far apart as possible (i.e., k-means++).

Set number of iterations (MAX_ITER) to 25 and number of clusters k to 10 for all the experiments carried out in this question. Your program should ensure that the correct amount of iterations are run.

(a) **Exploring initialization strategies with Euclidean distance [14 pts]**

1. [10 pts] Using the Euclidean distance (refer to Equation 1) as the distance measure, compute the cost function φ(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 φ(i) as a function of the number of iterations i=1..25 for c1.txt and also for c2.txt. (Hint: Note that you do not need to write a separate Spark job to compute φ(i). You should be able to calculate costs while partitioning points into clusters.)

2. [4 pts] Looking at the cost after 10 iterations when 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 φ(i)? Explain your reasoning.

(b) **Exploring initialization strategies with Manhattan distance [14 pts]**

1. [10 pts] Using the Manhattan distance metric (refer to Equation 3) as the distance measure, compute the cost function ψ(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 ψ(i) as a function of the number of iterations i=1..25 for c1.txt and also for c2.txt. (Hint: This problem can be solved in a similar manner to that of part (a))

2. [4 pts] Looking at the cost after 10 iterations when 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 ψ(i)? Explain your reasoning.

You can finish this assignment using either the Jupyter notebook or the cluster version.

What to submit:

A zip file containing

(i) Your source code for (a) and (b) (notebook or .py)

(ii) A plot of cost vs. iteration for two initialization strategies for (a) (notebook or image)

(iii) A plot of cost vs. iteration for two initialization strategies for (b) (notebook or image)

(iv) Your answers for (a)2 and (b)2 (notebook or pdf)

NOTE: please make sure (ii) and (iii) are easy to follow.