CSCE 4013-006: Assignment 3

Due 11:59pm Friday, November 2, 2018

1 k-means on Spark

This problem requires you to implement the k-means algorithm on Spark. The input is a set $\mathcal{X}$ of n data points in the d-dimensional space $\mathbb{R}^d$, the given number of clusters k, and the set of k initial centroids $\mathcal{C}$. The distance between any two points is computed using the Euclidean distance.

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:

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

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

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

The pseudo code of k-means algorithm is shown in Algorithm 1.

Algorithm 1: k-means Algorithm

```
Select k points as initial centroids of the k clusters;
repeat
    foreach point p in the dataset do
        Assign point p to the cluster with the closest centroid;
    foreach cluster c do
        Update the centroid of c as the mean of all the data points assigned to it;
until convergence;
```

Task Implement the k-means algorithm using Spark. In this task, we use $k = 10$. Please use `data.txt` as the data points and `centroid.txt` as the initial centroids.

1. `data.txt` contains the dataset with 1000 rows (i.e., $n = 1000$) and 20 columns (i.e., $d = 20$).
2. `centroid.txt` contains 10 initial cluster centroids.

For the convergence condition, you can either set the number of iterations to 20 or use a threshold 0.01.

Run the k-means on `data.txt` and `centroid.txt`. Generate a graph where you plot the cost function $\phi(i)$ as a function of the number of iterations.

Hint `template_kmeans.java` is a template of the java code for solving this problem. You can either directly use it in your project, or use it as a reference and write your code on your own.

What to submit:

1. The source code (.java or .py files).
2. The plot of cost vs. iteration.

2 PageRank on Spark

This problem requires you to implement the PageRank algorithm on Spark. The input file `graph.txt` is a randomly generated graph (connected and no dead-end). It has $n = 1000$ nodes (numbered 1,2,...,1000), and $m = 8192$ edges. There may be multiple directed edges between a pair of nodes, and your solution should treat them as the same edge. The first column refers to the source node, and the second column refers to the destination node.

Let $G = (V, E)$ be the directed graph and $M = [M_{ji}]_{n\times n}$ be an $n \times n$ matrix as defined in the class such that for any $i, j \in [1, n]$:

$$M_{ji} = \begin{cases} \frac{1}{deg(i)} & \text{if } (i \to j) \in E \\ 0 & \text{otherwise} \end{cases}$$

where $deg(i)$ is the number of outgoing edges of node i in G. If there are multiple edges in the same direction between two nodes, treat them as a single edge. By the definition of PageRank, assuming $1-\beta$ to be the teleport probability, and denoting the PageRank vector by the column vector r, we have the following equation:

$$r = \beta M r + \frac{1-\beta}{n}[1]_{n\times 1},$$

where $[1]_{n\times 1}$ is the $n \times 1$ vector with all entries equal to 1.

Based on above equation, the iterative procedure to compute PageRank works as follows:

1. Initialize: $r^{(0)} = \frac{1}{n}[1]_{n\times 1}$
2. For i from 1 to k, iterate: $r^{(i)} = \beta M r^{(i-1)} + \frac{1-\beta}{n}[1]_{n\times 1}$

Run the aforementioned iterative process on Spark for 40 iterations (assuming $\beta = 0.8$) and obtain the PageRank vector r. The matrix M can be large and should be processed as an RDD in your solution. Compute the following:

- List the top 5 node ids with the highest PageRank scores.
- List the bottom 5 node ids with the lowest PageRank scores.

For a sanity check, we have provided a smaller dataset `graph_small.txt`. In that dataset, the top node has id 53 with value 0.036.

Hint `template_pagerank.java` is a template of the java code for solving this problem. You can either directly use it in your project, or use it as a reference and write your code on your own.

What to submit:

1. The source code (.java or .py files).
2. List 5 node ids with the highest and least PageRank scores.