EECS E6893 Big Data Analytics - Fall 2023

Homework Assignment 1: Clustering & Classification on Spark MLlib, and Hadoop System Monitoring

Due Friday, October 6th, 2023 by 5:00pm

TL;DR

1. Implement and run K-means clustering in Spark
2. Process data with Spark Dataframe and perform classification with Spark MLlib
3. Install and set up Hadoop, monitor Hadoop metrics
4. Submit your codes and report to Canvas

Abstract

The goals of this assignment are to (1) understand how to implement K-means clustering algorithm in Spark by utilizing *transformations and actions*, (2) understand the impact of using different distance measurements and initialization strategies in clustering, (3) learn how to use the built-in Spark MLlib library to conduct unsupervised and supervised learning, (4) have experience of monitoring Hadoop metrics.

In the first question, you will conduct **document clustering**. The dataset we'll be using is a set of vectorized text documents. In today's world, you can see applications of document clustering almost everywhere. For example, Flipboard uses LDA topic modelling, approximate nearest neighbor search, and clustering to realize their "similar stories / read more" recommendation feature. You can learn more by reading this blog post. To conduct document clustering, you will implement the classic iterative K-means clustering in Spark with different distance functions and initialization strategies.

In part 2, you will learn how to use the built-in Spark MLlib library to conduct supervised and unsupervised learning, then have experience of processing data with *ML Pipeline* and *Dataframe*.

You will load data into Spark Dataframe and perform **binary classification** with Spark MLlib. Use Logistic regression, Random Forest, Naive Bayes, Decision Tree, Gradient Boosting Trees, Multi-layer perceptron, Linear Support Vector Machine, One-vs-Rest to do classification to the same dataset, then evaluate each result and compare the performances of these methods.

In the last part, you will download and set up Hadoop, as well as install Hadoop in Pseudo Distributed Mode. You will learn to monitor metrics of Hadoop through HTTP API, and get familiar with the usages and importance of different metrics.

1. Iterative K-means clustering on Spark

The idea behind K-Means algorithm is straightforward: in each iteration, *k centroids* are initialized, each point in the space is assigned to the *nearest centroid,* and the centroids are re-computed based on the assignment of points to clusters. The pseudo code of iterative K-means clustering is as follows:

Algorithm 1 Iterative k-Means Algorithm

```
1: procedure ITERATIVE k-MEANS
2:     Select k points as initial centroids of the k clusters.
3:     for iterations := 1 to MAX_ITER do
4:         for each point p in the dataset do
5:             Assign point p to the cluster with the closest centroid
6:         end for
7:         for each cluster c do
8:             Recompute the centroid of c as the mean of all the data points assigned to c
9:         end for
10:    end for
11: end procedure
```

Note the word *nearest or closest*, we have to define a way to measure it. In class, Prof. Lin mentioned several distance functions. Here, we will use L2 (Euclidean) and L1 (Manhattan) distance as our similarity measurement.

Formally, say we have two points A and B in a d dimensional space, such that A = [a1,a2, ⋯, ad] and B = [b1,b2, ⋯, bd], the Euclidean distance (L2 distance) between A and B is defined as:

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

, and the Manhattan distance (L1 distance) between A and B is defined as:

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

A clustering algorithm aims to minimize *within-cluster cost function*, defined as:

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

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

for L2 and L1 distance.
Use the data provided to perform document clustering. A document is represented as a 58 dimensional vector of features. Each dimension (or feature) in the vector represents the importance of a word in the document. The idea is that documents with similar sets of word importance may be about the same topic.

The data contains 3 files: (in q1.zip on Canvas)

(1) *data.txt* contains the vectorized version of documents, which has 4601 rows and 58 columns.
(2) *c1.txt* contains k initial cluster centroids. These centroids were chosen by selecting k random points from the input data.
(3) *c2.txt* contains initial cluster centroids which are as far away as possible. You could do this by choosing first centroid c1 randomly, and then finding the point c2 that is farthest from c1, then selecting c3 which is farthest from c1 and c2, and so on.

For the homework questions, set the number of iterations to 20, and the number of clusters k to 10.

Homework submission for question one: (35%)
Implement iterative K-means in Spark. We've provided you with starter code *kmeans.py* on Canvas, which takes care of data loading. Complete the logic inside the *for loop*, and run the code with different initialization strategies and loss functions. Feel free to change the code if needed, or paste the code into a Jupyter notebook. Take a screenshot of your code and results in your report.

(1) Run clustering on *data.txt* with *c1.txt* and *c2.txt* as initial centroids and use L1 distance as similarity measurement. Compute and plot the *within-cluster cost* for each iteration. You'll need to submit two graphs here. (10%)
(2) Run clustering on *data.txt* with *c1.txt* and *c2.txt* as initial centroids and use L2 distance as similarity measurement. Compute and plot the *within-cluster cost* for each iteration. You'll need to submit two graphs here. (10%)
(3) t-SNE is a dimensionality reduction method particularly suitable for visualization of high-dimensional data. Visualize your clustering assignment result of (2) by

reducing the dimension to a 2D space. You'll need to submit two graphs here. (5%)

(4) For L2 and L1 distance, are random initialization of K-means using *c1.txt* better than initialization using *c2.txt* in terms of cost? Explain your reasoning. (5%)

(5) What is the time complexity of the iterative K-means? (5%)

2. Binary Classification on Spark

In this question, we are going to use the Adult dataset, which is publicly available at the UCI Machine Learning Repository. You can download the *adult.data* from this link, and add .csv extension to the download file.

This data derives from census data and consists of information about 48k individuals and their annual income. We are interested in using the information to predict if an individual earns less than 50K or larger than 50k per year. It is natural to formulate the problem as binary classification.

The model we use for binary classification is *Logistic Regression*, *Random Forest*, *Naive Bayes*, *Decision Tree*, *Gradient Boosting Trees*, *Multi-layer perceptron*, *Linear Support Vector Machine*, *One-vs-Rest*, totally 9 models. Take Logistic Regression as an example, it takes one more step than linear regression: apply the sigmoid function to compress the output values to 0~1 and compare with the threshold to see which cluster it should belong to. Linear regression outputs continuous values while logistic regression outputs probabilities in the range 0 to 1 which can then be mapped to two or more discrete classes. You could learn more details like how Spark performs optimization here.

Homework submission for question two: (35%)

1. Data loading: (7.5%)
   Read the csv file into a Dataframe. You could set *"inferschema"* to true and rename the columns with the following information: "age", "workclass", "fnlwgt", "education", "education_num", "marital_status", "occupation", "relationship", "race", "sex", "capital_gain", "capital_loss", "hours_per_week", "native_country", "income". You could learn more about Dataframe from this doc.
2. Data preprocessing: (7.5%)
   Convert the categorical variables into numeric variables with ML Pipelines and Feature Transformers. You will probably need *OneHotEncoderEstimator,*

StringIndexer, and VectorAssembler. Split your data into the training and test sets, respectively, with a ratio of 70% and 30% and set the seed to 100.

3. Modelling: (10%)
   Train logistic regression, Random Forest, Naive Bayes, Decision Tree, Gradient Boosting Trees, Multi-layer perceptron, Linear Support Vector Machine, One-vs-Rest from Spark MLlib with the training set.
4. Evaluation: (10%)
   Apply all the models on the test set, analyze and compare their accuracies to rank these methods.

3. Monitoring Hadoop metrics

Hadoop is a framework that allows for reliable, scalable, distributed computing. In this question, we are going to install and set up Hadoop. Hadoop is made up of four core modules that are supported by a large ecosystem of supporting technologies and products. The modules include

- Hadoop Common: The common utilities that support the other Hadoop modules.
- Hadoop Distributed File System (HDFS™): A distributed file system that provides high-throughput access to application data.
- Hadoop YARN: A framework for job scheduling and cluster resource management.
- Hadoop MapReduce: A YARN-based system for parallel processing of large data sets.

Through the following exercises, we are going to get familiar with these modules and learn the practical skills to monitor the Hadoop metrics.

Homework submission for question three: (30%)

Read through this guide as well as the tutorial to finish the following questions.

(1) Download and setup Hadoop: (7.5%)
Follow the download instructions in the corresponding tutorial we provide and install Hadoop in Pseudo Distributed Mode. Provide screenshots of each step in the “Verify Hadoop installation” section of the tutorial.

(2) HDFS metrics monitoring: (7.5%)
- Monitor HDFS metrics through HTTP API. Provide a screenshot in your report. (3.75%)
- Select five metrics that you think are most important. Explain their usages and why they are important. (3.75%)

(3) MapReduce counters monitoring: (7.5%)
- Collect MapReduce counters related information through the web UI. Provide a screenshot. (3.75%)
- Describe how to monitor the execution of MapReduce tasks through related metrics. (3.75%)

(4) YARN metrics monitoring: (7.5%)
- Monitor YARN metrics through HTTP API. Provide a screenshot in your report. (3.75%)
- Select five metrics that you think are most important from the returned json. Explain their usages and why they are important. (3.75%)