EECS E6893 Big Data Analytics - Fall 2019

Homework Assignment 1: Clustering, Classification, and Spark MLlib

Due Friday, October 4th, 2019 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. 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 supervised and unsupervised learning, (4) have experience of processing data with *ML Pipeline* and *Dataframe*.

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 compare with the one implemented in Spark MLlib.

In the second question, you will load data into Spark Dataframe and perform **binary classification** with Spark MLlib. We will use logistic regression model as our classifier, which is one of the foundational and widely used tools for making classifications. For example, Facebook uses logistic regression as one of the components in its online advertising system. You can read more in a publication here.

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:

(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 number of iterations to 20, and the number of clusters k to 10.

Homework submission for question one: (60%)

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 *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. (20%)
(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. (20%)
(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. (10%)
(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 with Spark MLlib

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 into binary classification.

Homework submission for question two: (40%)

First upload the csv file to GCP cloud storage, and finish the following steps. Provide screenshots to your code / notebook and results.

(1) Data loading: (10%)
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: (10%)
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 training set and test set with ratio of 70% and 30% and set the seed to 100.

(3) Modelling: (10%)
Train a logistic regression model with train set. Learn more about models provide in Spark MLlib here. After training, plot ROC curve and Precision-Recall curve of your training process.

(4) Evaluation: (10%)
Apply your trained model on the test set. Provide the value of area under ROC, accuracy, and confusion matrix. You should expect the accuracy to be around 85%.

Feel free to do EDA, feature engineering, or try other more complicated models.