THE UNIVERSITY OF SYDNEY

School of Information Technologies
Dr. Ying Zhou

COMP5349: Cloud Computing **Sem. 1/2018**

Assignment 2: Spark Machine Learning Application

Group Work: 20% **10.05.2018**

1 Introduction

This assignment tests your ability to analyze and tune the performance of Spark application. The assignment contains three stages. All stages use the MNIST data set of handwritten digits(`http://yann.lecun.com/exdb/mnist/`). The original data set contains four files to store training image, training label, test image and test label. The files have been converted to csv format and are stored in School of IT Hadoop cluster under `/share/MNIST` (`http://soit-hdp-pro-1.ucc.usyd.edu.au:50070/explorer.html/share/MNIST`)

2 Stage One: KNN Classifier

In this stage, you are asked to develop a KNN classifier to classify 10,000 test images. The KNN classifier works by finding K nearest neighbor of an input object from the training set and using the neighbors' labels to determine the input object's label. In the classic setting, the object is given a label most common among its neighbors.

Each MNIST image is a 28 x 28 grayscale image, using the raw pixel as feature would have a dimension of 748. You are required to run PCA to reduce the dimensions before running the KNN classifier.

There are two hyperparameters: the reduced dimension d and the k nearest neighbours. Both should be configurable.

You should implement exact KNN, not an approximation. **Your code should utilize various Spark features and design patterns** to optimize the performance in terms of execution time.

Running the classifier on MNIST test data should produce the predicted label of each test image as well as the following summary statistics for each label: precision, recall and f1-score.

3 Stage Two: Performance Analysis

In this stage, you are asked to run your KNN classifier with different combinations of hyperparameters and execution properties. For reduced dimension d, you should choose

two values between 50-100; For the nearest neighbour number k, you should choose two values between 5 and 10. We are only interested in the execution property that controls the maximum parallelism, namely num-executors and executor-cores or total-executor-cores. You should choose three levels that suits your execution environment. If you do performance testing on the cluster, the choice could be quite open; If you do performance testing on your own machine, you are restricted by the number of cores your CPU have. In total, there will be 12 combinations of all different values.

For each combination, you should record important summary execution statistics such as execution time and total I/O cost. You may include other interesting metrics. You should also produce a few diagrams to plot individual statistics under various settings and explain your observation in the report.

4 Stage Three: Spark Classifier Exploration

MNIST is a widely used data set for classification problem. Spark machine learning library contains a number of classifiers that can be applied on MNIST data set. In this stage, you are asked to run two such algorithms on MNIST data set and compare their performance in terms of accuracy and execution statistics. You need to further explore various parameter settings of each classifier.

Once you have formed group, your tutor will assign the pair of classifiers your group will work on in this stage and some further instructions with respect to those particular classifiers.

You should use the classifiers implemented in Spark MLib library for this stage.

5 Group Collaboration

This is a group assignment. Each group can have up to three members. You are asked to use git to collaborate among members, you can use the university's hosting service, GitHub or BitBucket. Once you have created the repository on a hosting platform, you should give your tutor access to your repository.

A peer evaluation form will be distributed in week 10 for you to report any issue within group before week 11. Group members are expected to make fair contribution to the project. If members of your group do not contribute sufficiently you should alert your tutor as soon as possible. The tutor has the discretion to scale the group's mark for each member based on their contribution to the project.

6 Deliverable

There are two deliverables: **source code** and **project report** (around 10 pages). Both are due on **Wednesday 30th of May 23:59 (Week 12)**. Please submit the source code and the

pdf version of your report as a zip file in Canvas. Please do not include any thing other than the source code and the report in your zip file.

You need to **demo** your stage one and stage three implementation in week 12 during tutorial time. Please also submit a hard copy of your report together with signed group assignment cover sheet during the demo. Please note that **ALL** members of a group must attend the demo and explain your contribution. Group members who do not attend the demo will not receive any mark, unless he(she) has been granted special permission for not attending.

7 Report Structure

The report should contain three sections corresponding to the three stages: Design, Performance Analysis, and Spark Classifier Exploration. You may include an introduction section at the beginning and an appendix at the end.

The design section should describe the architecture of your implementation. The description should be similar to those in assignment one report. You may include short code snippet in this section to help explain certain point. You should highlight performance optimization features you use in the implementation. This section should also include a sample result of running your classifier.

The Performance Analysis section should give an overview of the execution environment as well as the hyperparameters values you have chosen. It should include a few diagrams highlighting performance variations under various settings. You may include screenshots of history server pages. You should also include enough explanation of the performance statistics and differences observed.

The Spark Classifier Exploration section should briefly describe the two algorithms you investigate. You may also describe how you prepare the data for the classifier and how you collect the output. The section should also include a couple of diagrams/tables to compare execution statistics and accuracy of the two algorithms. You should describe interesting findings you discover about the algorithms.

8 Stage Three Further Details

Below are the list of five classifiers, some with a few variations, you may explore in stage three. We focus on exploring execution statistics as well as prediction accuracy. You only need to investigate two out of the five classifiers. For each classifier, you are asked to explore one parameter that might affect the execution statistics as well as the prediction accuracy. You also need to compare the execution statistics and prediction accuracy of the two classifiers.

- **Decision Tree**: in general, a Decision Tree's time complexity depends on the number of training samples and the dimension of the feature vector. You are asked to investigate the impact of feature vector's dimensionality on the execution statistics and

prediction accuracy of Spark's Decision Tree implementation. The dimension variations can be created with a dimensionality reduction algorithm like PCA. You should study at least three different dimension values, including one representing the raw pixel value feature vector (unreduced feature vector). You can pick the other two dimension values.

- **Random Forest**: the time complexity of Random Forest algorithm depends on all those in a Decision Tree (see above) as well as the number of trees. You are asked to investigate the impact of tree number on execution statistics and prediction accuracy of Spark's Random Forest implementation. You should study at least three different tree numbers. You can pick the actual values.

- **Logistic Regression**: the time complexity of Logistic Regression algorithm depends on the number of training samples, the dimension of the feature vector, and the number of iterations. You are asked to explore the impact of feature vector's dimension on execution statistics and prediction accuracy of Spark's Logistic Regression implementation. The dimension variations can be created using a dimensionality reduction algorithm like PCA. You should study at least three different dimension values, including one representing raw pixel value feature vector (unreduced feature vector). You can pick the other two dimension values.

- **Naive Bayes**: the time complexity of Naive Bayes algorithm depends on the number of training samples as well as the dimension of the feature vector. You are asked to explore the impact of the dimension of the feature vector on the execution statistics and prediction accuracy of Spark's Naive Bayes implementation. The dimension variations can be created using a dimensionality reduction algorithm like PCA. You should study at least three different dimension values, including one representing raw pixel value feature vector (unreduced feature vector). You can pick the other two dimension values.

- **Multilayer Perceptron Classifier**: A Multilayer Perceptron Classifier has many hyperparameters that could affect the execution statistics as well as the prediction accuracy. Groups assigned to explore this classifier need to investigate only one of the following:

  - Effect of hidden layer size: Build a Multilayer Perceptron classifier with one hidden layer. Choose three values between 50 and 100 as the hidden layer size. Investigate the impact of hidden layer size on execution statistics and prediction accuracy.
  - Effect of iteration number: Build a Multilayer Perceptron classifier with one hidden layer and a fixed layer size. Choose three *maxIter* values with the smallest being 100. Train the classifier with those values to investigate the effect of iteration number on prediction accuracy as well as execution statistics.

– effect of block size: Build a Multilayer Perceptron classifier with one hidden layer and a fixed layer size. Choose three *blockSize* values, with the maximum being 30(MB). Train the classifier with those values to investigate the effect of block size on prediction accuracy as well as execution statistics.