CS6220 BDS 2021 Fall

Homework Assignment 1
(Programming Category)

Student Name:_____

Student Session: cs6220-A

You are given 3 types of programming problems. You only need to choose one of the 3 problems as your second homework. For the problem with multiple options, you are required to choose only one option. Feel free to choose any of your favorite programming languages, such as Java, C, Perl, Python, and so forth.

You are given three types of programming problems in the first homework assignment.
Problem 1 is designed to get hand-on experiences on learning Distributed/ Parallel Big Data Computing with Hadoop and/or Spark MapReduce. (page 2-3)

Problem 2 is designed to get Hand-on experience on solving a real world data analytic system problem using real world datasets, instead of benchmark datasets. (page 3-5)

Problem 3 is designed for those students who prefer programming option on learning algorithm (comparing learning efficiency with CPU v.s. CPU+CPU or comparing learning efficiency with deep learning v.s. shallow learning algorithms (page 5-11).

Each problem consists of two or more options (subproblems) and you only need to choose one option (subproblem) from one of the three Problems as your first homework. This provides sufficient diversity for students with different knowledge and educational backgrounds. Feel free to choose any of your favorite programming languages, such as Python, Java, C, Perl, and so forth.

Post Date: Monday of Week 2
Due Date: midnight on Friday of Week 3 (Sept 10)
(with no penalty extension to Saturday 9am, hard deadline).

Problem 1. Learning Distributed/Parallel Big Data Computing with Hadoop and/or Spark MapReduce

This problem has 3 options. You only need to choose one option. The last two

options are designed for those students who are familiar with Hadoop MapReduce programming or Spark and would like to try something more challenging.

Deliverable (for each of the three options):

(a) Source code and executable with readme. (provide URL if you use open source code packages)
(b) Screen Shots of your execution process/environments
(c) Input and output of your program
(d) Runtime measurements in excel plots or tabular format.
(e) Two scenarios that you choose to run the same MapRedue program to compare the performance or two programs solving the same problem with the same dataset.
(f) Provide your analytical report for your homework 2.

Hint:

a. If you use two or more datasets of different sizes, you report the scaling effect of your program.
b. If you use two different implementations of MapReduce programs running on the same dataset, you report the runtime performance optimization aspects of your two programs.

Option 1.
Suitable for students who are the beginner of Hadoop/Spark MapReduce

- Install HDFS and Hadoop MapReduce on your laptop.
- Run the word count map-reduce program, and report the runtime for two different sizes of datasets.
- Using MapReduce to solve another problem. You may choose one of the following or create a problem yourself.
  - Consider a dataset of 20 files, print the top 100 words occurring in the most files
  - Solving K-means clustering problem with a dataset given in Spark ML lib, R, Hadoop Mahout ML library, or Scikit-learn.

You may use excel file to generate your runtime statistics plot or organize the performance measurement data in a tabular format.

You are encouraged to learn by observing the runtime performance of Hadoop/ Spark MapReduce program through different ways of programming the same problem and show their impact on the runtime performance of the MapReduce job.

Option 2:
This option is designed for students who are familiar with Hadoop MapReduce or

Spark and are interested in hand-on comparison of them through example data problems and configuration tuning.

Comparing Hadoop MapReduce with Spark MapReduce

- Install HDFS, Hadoop MapReduce and Spark MapRedue on your laptop.
- Run the word count map-reduce program on both Hadoop and Spark, and report the runtime comparison.
- Using MapReduce to solve another problem of your choice. You may choose one of the following or create a problem yourself.
    - Consider a dataset of 20 files, print the top 100 words occurring in the most files
    - Solving K-means clustering problem with a dataset given in Spark ML lib, or R, or Hadoop Mahout ML library, or Scikit-learn.

Run your MapReduce program on both Hadoop and Spark to compare the performance results.

You may use excel file to generate your runtime statistics plot or organize the performance measurement data in a tabular format.

You are encouraged to learn by observing the runtime performance of Hadoop/ Spark MapReduce program through different ways of programming the same problem and show their impact on the runtime performance of the MapReduce job.

Problem 2: Hand-on experience on solving a real world data analytic system problem.

This problem is designed for those students who are interested in using Hadoop MapReduce to solve problems using a real world dataset. You are recommended to use software packages from Spark ML library, Hadoop Mahout directly or use Hadoop MapReduce or Spark as the underlying platform and running some of your favorite ML library on top, such as Scikit-learn (http://scikit-learn.org/), Weka (http://www.cs.waikato.ac.nz/ml/weka/) or R https://bigdata-madesimple.com/10-r-packages-machine-learning/.

We problem two real datasets as option 1 and option 2. You may choose a real world dataset of your favorite ones which are accessible in public domain and address the problem on mining your dataset in option 3. The deliverable will be the same as the option 1 and option 2.

Problem 2.1 Mining the Kaggle million songs dataset

The million songs dataset is a big data mining challenge. One of the earlier Kaggle challenges. It contains (1) the full listening history for 1M users, (2) half of the listening history for 110K users (10K validation set, 100K test set). You are asked to predict the missing half.

http://www.kaggle.com/c/msdchallenge
Statistics of the dataset:

- Large Data Set
  - 1,019,318 users
  - 384,546 MSD songs
  - 48,373,586 (user, song, count)
- Kaggle Competition: offline evaluation
  - Predict songs a user will listen
  - Training: 1M user listening history
  - Validation: 110K users

This is a well-known big data challenge and there are software postings in the Public domain. Feel free to use any of them you found helpful.

Deliverable:

(1) URL of the dataset
(2) The subset of the data you used in your program or model training and testing.
(3) The software packages and tools or library you use in your programs
(4) The open source code you leveraged in developing your MillionSongMiner.
(5) The experiments you conducted to report your mining accuracy, ideally over three different datasets extracted from the million-songs dataset in three different sizes (e.g., 2x, 5x, 10x of your initial smaller dataset, say 1000, or 10,000 songs.)
(6) Analysis of your hand-on experiences, with three lessons learned or three observations you wish to make.

Problem 2.2 Mining the White House visitor Log dataset

The White House Visitor Log dataset available in the directory of programming assignment in CSV. It was an older version in 2009 from http://www.whitehouse.gov/files/disclosures/visitors/WhiteHouse-WAVES-Released-0827.csv which was replaced by recent datasets. Feel free to use the recent datasets whitehouse.gov.

You are required to write an efficient program (such as using MapReduce) to find the following information:

(i) The 10 most frequent visitors (NAMELAST, NAMEFIRST, NAMEMID) to the White House.

(ii) The 10 most frequently visited people (visitee_namelast, visitee_namefirst) in

the White House.

(iii) The 10 most frequent visitor-visitee combinations.

(iv) Some other interesting statistics that you can think of.

Throughout this programming assignment, do not limit your programs to run with a single design choice (such as one reduce task only).

Deliverable:

(1) URL of the dataset
(2) The software packages and tools or library you use in your programs
(3) The open source code you leveraged in developing your visitor log miner
(4) The experiments you conducted to report your mining accuracy and time, ideally over three different datasets in three different sizes (e.g., 2x, 5x, 10x of your initial dataset.)
(5) Analysis of your hand-on experiences, with three observations you wish to make.

Option 2.3: You may choose a real world dataset of your favorite ones which are accessible in public domain and address the problem on mining your dataset.

Problem 3. Hand on comparative experience on Improving the performance of supervised learning.

This assignment helps you to gain some hand-on experience with two options: (3.1) CPU v.s. CPU+GPU for supervised learning. (3.2) shallow v.s. deep algorithms for supervised learning.

(1)Software download. You may choose traditional machine learning algorithm such as SVM, Decision Tree (Random Forest), or neural network model, such as DenseNet, ResNet, LeNet, AlexNet, etc. from a popular machine learning framework of your choice, such Scikit-Learn, TensorFlow (google), Caffe, Torch/ PyTorch, OpenCV (Intel), CNTK (Microsoft).
https://scikit-learn.org/
https://www.tensorflow.org.
https://caffe.berkeleyvision.org
http://torch.ch or https://pytorch.org
https://docs.opencv.org/master/index.html
https://www.pyimagesearch.com/2017/08/21/deep-learning-with-opencv/
https://docs.microsoft.com/en-us/cognitive-toolkit/setup-cntk-on-your-machine
https://keras.io

You may also use Keras, a high-level neural networks API, written in **Python**, for this homework as it is capable of running on top of TensorFlow, CNTK, Theano, etc..

In the rest of the problem description, I will use TensorFlow (TF) to describe the homework but you can replace TF with one of your favorite ML/DL frameworks.

(2)Dataset Preparation
You are asked to use an image dataset, which has images about at least two types of objects, such as Cat v.s. Dog, to train an image classification model to perform a binary classification task, with high accuracy. The training algorithm can be either a neural network algorithm or a traditional ML algorithm, such as SVM.

You may find such image datasets in the public domain such as Kaggle, ImageNet, or creating your own.

Some example datasets are

(a) Color rich single object images like color photos of cats and dogs, or color photos of people. Such as Kaggle (https://www.kaggle.com/c/dogs-vs-cats), or CIFAR10, (https://www.cs.toronto.edu/~kriz/cifar.html), or ImageNet (https://github.com/tensorflow/models/tree/master/tutorials/image/imagenet). You may also choose simple grey-scale images. One example is MNIST. (http://yann.lecun.com/exdb/mnist/)
(b) MNIST dataset. http://yann.lecun.com/exdb/mnist/.
(c) USPS dataset. https://www.kaggle.com/bistaumanga/usps-dataset
(d) Traffic Sign Recognition. https://www.kaggle.com/c/traffic-sign-recognition LISA dataset: http://cvrr.ucsd.edu/LISA/lisa-traffic-sign-dataset.html
(e) p2-trafficsigns. https://github.com/ vxy10/p2-TrafficSigns
(f) The face database. https://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html

You can also find most of them at https://git-disl.github.io/GTDLBench/datasets/

(3)Training Algorithm Choices: If you want to use a traditional ML algorithm, such as Logic Regression, SVM, Decision Tree (Random Forest), you will need to find the image dataset with CSV format instead of raw pixel format. For example, you can find MNIST CSV or AT&T Face CSV from https://github.com/git-disl/GTDLBench under datasets.

(4)Training v.s. Test data partition: You can use 80:20, namely using 80% of the images as the training dataset (say 40 pictures of cat and 40 pictures of dog images in a dataset of 100) for training and 20% (10 pictures of cat and 10 pictures of dog images) as the test dataset. Once you learned to use TensorFlow CNN algorithm or SVM to train a binary classifier model over 100 images, then

training another binary classifier on 1000 images or 5000 images will be straightforward.

This programming homework provides two options. You are required to choose one option.

If you have no hand-on experience with any deep learning software framework, then you are recommended to choose the second option of this problem.

Problem 3.1 Hand-on Experience on supervised learning by using/ and comparing two different big data computing platforms: (i) CPU and (ii) CPU + GPU.

Requirement. You are asked to perform the supervised learning on varying input data complexity and varying model complexity (changing K classes, say K=2, 10, 100). You are asked to use an input dataset that is N-dimensional vector representation, such as an image dataset.

(1) You can choose a deep Neural Network algorithm of your favorite, such as CNN, ResNet, LeNet, MobiNet, etc., you will need to choose your favorite deep learning framework (library), say TensorFlow. After download TensorFlow at https://www.tensorflow.org. Following the instruction to install it on your laptop or desktop computer: https://www.tensorflow.org/install/install_linux#ValidateYourInstallation. There are several options to install TensorFlow on machines without GPU card(s). For example, installing Tensorflow on virtualenv will isolate your Tensorflow's python environment.

(2) You start with a dataset and a learning task. If the learning task is to train a **binary** classifier (K=2), say learn to identify dog and cat. Then you first decide the size and resolution of your dataset for the input. Each decision on size/resolution gives you a different version of your dataset. By using each version of the dataset, you can train a different binary classifier. By obtaining multiple classifiers trained by the same CNN algorithm on different versions of the same real-world dataset, you may be able to compare and analyze the testing accuracy from different classifiers, such as reasons why the accuracy is good or not so good.

For example, if you use TensorFlow ResNet as your training algorithm, then you can prepare four versions of the input datasets: 28x28-CatDog-100, 64x64-CatDog-100, 28X28-CatDog-1000, 64x64-CatDog-1000. Report the trained model size in MB, the test accuracy, the training time and test time.

Alternatively, if you choose to use SVM or Random Forest as your algorithm to train your cat-dog binary classifier, then you can also show the test accuracy, the training time, the test time and the model size too.

Hint: This option allows you to choose any algorithm (shallow or deep) to train four versions of the k-class classifier, each version representing two different data input complexity (e.g., two different input image resolutions, 32x32, 128x128) and two different learning task complexity (e.g., k=2, 10). If you control your input data size to be not too large, your training with CPU will be able to complete within your manageable time.

Goal of Learning: Report your observations and experiences on how different input data complexity (varying resolution, varying size) and learning task complexity (varying K, varying resolution) may impact on the performance of your model training time and accuracy when using CPU and when using CPU+GPU.

(3) **Outlier Test Scenario using only CPU+GPU as your computing platform**. Once you complete the first four steps, you are asked to a new class called "unknown" with equal amount of training images that contain neither cat nor dog, such as flower, human, airplane … and use these out of distribution examples to perform two tasks:

a. as test dataset. Report the test accuracy and test time for your pre-trained k-class classifier.
b. To train a new classifer with K+1 classes, , with K classes as regular k-class classification and 1 additional class as "unknown" class. Perform the same tests above (using IID test dataset and out of distribution test dataset). Report your training measurement in terms of training time, accuracy, test time and accuracy and the trained model size. Then extend K=2+1 to larger value, such as k=11 (10+1), repeat the above and compare your experience with k=3.

Hint. The easiest is to start with 3-class classifier such as cat and dog and unknown. For any out of distribution images, such as flowers, people, cellphones, airplane, etc., they will be trained to be classified as unknow class.

(4)Deliverable:

(i) provide URL of your open source code package and dataset download.
(ii) Screen shots of your execution process/environments
(iii) **Input Analysis**: *Use a table to report your training configuration parameters*:
   a. the input dataset (total # images, for each image, the resolution, storage size in KB or MB per image, the total storage size of each version of your datasets in MB or GB).
   b. choose and show 5 sample images per class for all K classes in each of the four datasets.

c. the neural network (NN) model you choose to be the training algorithm, such as LeNet, ResNet, DenseNet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for the 4 configurations you use to train your 4 CNN classifiers for performing the same classification task.

   If you are using traditional ML algorithm such as SVM or Decision Tree/ Random Forest, then you are asked to show the kernel you use for training, such as which linear or non-linear kernel is used.

d. You are asked to report the default training convergence condition used in your training algorithm, usually it is the termination condition in terms of the total #iterations for training to complete.

(iv) **Output Analysis:** *Report the performance comparison and analysis of your four K-class CNN classifiers and for each of the four classifiers:*

a. You are asked to provide a table (ideally in an excel file, but not required) in terms of training time, testing time and testing accuracy.
b. You are asked to record the trained model size in MB.
c. You are asked to add the test accuracy and time for your outliner dataset in requirement.
d. Report your comparison with CPU only and CPU+GPU platform for this option.
e. You are asked to make at least three observations from your experimental comparison of the four classifiers.

Problem 3.2 Hand-on Experience on supervised learning using/comparing shallow algorithm v.s. deep algorithm.

Goal of Learning: Report your observations and experiences on how different input data complexity (varying resolution, varying size) and learning task complexity (varying K, varying resolution) may impact on the performance of your model training when using shallow algorithms for training compared to using deep neural network (the NN layer should be more than 3).

Requirement. For traditional ML algorithms, you may choose from e.g., Scikit-Learn, such as Random Forest, Support Vector Machine SVM to train a shallow model based K-class classifier. Then train deep neural network models on the same dataset(s) to produce deep K-class classifier. You can use any of your favorite deep Neural Network algorithm, such as CNN, ResNet, LeNet, etc., from any deep learning framework, say TensorFlow. After download TensorFlow at https://www.tensorflow.org. Following the instruction to install it on your laptop or

desktop computer: https://www.tensorflow.org/install/install_linux#ValidateYourInstallation. There are several options to install TensorFlow on machines without GPU card(s). For example, installing Tensorflow on virtualenv will isolate your Tensorflow's python environment.

You start with a dataset and a learning task. If the learning task is to train a (K=2) **binary** classifier for dog/cat. Then you first decide the size and resolution of your dataset for the input. Each decision on size/resolution gives you a different version of your dataset. By using each version of the dataset, you can train a different binary classifier. By obtaining multiple classifiers trained by the same CNN algorithm on different versions of the same real-world dataset, you may be able to compare and analyze the testing accuracy from different classifiers, such as reasons why the accuracy is good or not so good.

For example, if you use TensorFlow Resnet as your training algorithm, then you can prepare four versions of the input datasets: 28x28-CatDog-100, 64x64-CatDog-100, 28X28-CatDog-1000, 64x64-CatDog-1000. Report the trained model size in MB, the test accuracy, the training time and test time.

Next, you use Logic Regression or SVM or Random Forest as your algorithm to train your K-class classifier, record the test accuracy, the training time, the test time and the model size. Make the comparison with deep neural network trained K-class classifier(s).

(4) **Outlier Test Scenario using only CPU+GPU as your computing platform**. Once you complete the first four steps, you are asked to a new class called "unknown" with equal amount of training images that contain neither cat nor dog, such as flower, human, airplane … and use these out of distribution examples to perform two tasks:

a. as test dataset. Report the test accuracy and test time for your pre-trained k-class classifier.
b. To train a new classifer with K+1 classes, , with K classes as regular k-class classification and 1 additional class as "unknown" class. Perform the same tests above (using IID test dataset and out of distribution test dataset). Report your training measurement in terms of training time, accuracy, test time and accuracy and the trained model size. Then extend K=2+1 to larger value, such as k=11 (10+1), repeat the above and compare your experience with k=3.

Hint. The easiest is to start with 3-class classifier such as cat and dog and unknown. For any out of distribution images, such as flowers, people, cellphones, airplane, etc., they will be trained to be classified as unknow class.

(4)Deliverable:

(v) provide URL of your open source code package and dataset download.

(vi) Screen shots of your execution process/environments

(vii) **Input Analysis**: *Use a table to report your training configuration parameters*:

e. the input dataset (total # images, for each image, the resolution, storage size in KB or MB per image, the total storage size of each version of your datasets in MB or GB).

f. choose and show 5 sample images per class for all K classes in each of the four datasets.

g. the training v.s. testing data split ratio and size used in your classifier training. (If you use 1000 images with 8:2 training v.s. testing ratio, then 800 for training and 200 for testing.)

h. the neural network (NN) model you choose to be the training algorithm, such as LeNet, ResNet, DenseNet, and the default NN structures (e.g., CNN with at least 2~5 convolutional layers), and the default hyper-parameters, such as neuron size, the number of weight filters and size, the min-batch size, #epochs/#iterations (convergence), in a table for the 4 configurations you use to train your 4 CNN classifiers for performing the same classification task.

   If you are using traditional ML algorithm such as SVM or Decision Tree/ Random Forest, then you are asked to show the kernel you use for training, such as which linear or non-linear kernel is used.

i. You are asked to report the default training convergence condition used in your training algorithm, usually it is the termination condition in terms of the total #iterations for training to complete.

(viii) **Output Analysis:** *Report the performance comparison and analysis of your four K-class CNN classifiers and for each of the four classifiers:*

f. You are asked to provide a table (ideally in an excel file, but not required) in terms of training time, testing time and testing accuracy.

g. You are asked to record the trained model size in MB.

h. You are asked to add the test accuracy and time for your outliner dataset in requirement

i. Report your comparison between model trained using deep v.s. shallow learning algorithms.

j. You are asked to make at least three observations from your experimental comparison of the four classifiers.