EE379K
Spring 2016
Dr. Sriram Vishwanath

Lab 4 – Random Forests Using MLLib

Assigned: 4/5/16 **Due: 4/14/16 by 11:59pm**

This lab will get you familiarized with Spark and MLLib, which is the machine learning library for Spark.

Setup

Go into your Git directory and do a "git pull" to get the new materials for this assignment. You should see a "lab4" directory if the pull was successful. If you do not see it, contact me.

This lab will require you to use Spark. To install it, go to the following link (in your VM):

http://spark.apache.org/downloads.html

Choose the 1.6.1 release (the default), and choose the "Pre-built for Hadoop 2.6 and later" package type. Then download the tgz file. Once you have it downloaded, extract the contents of the tgz file with the command:

tar –xvzf spark-1.6.1-bin-hadoop2.6.tgz

I recommend moving the resulting folder to your home directory.

PySpark

We will be using PySpark (which is a Python wrapper around the Spark core library). There are two ways to run Spark programs: either through the Spark interactive shell, or standalone. I suggest running in standalone mode for this assignment (though you may find interactive mode handy for experimentation). To run in interactive mode, open a terminal inside the Spark folder and execute the following command:

./bin/pyspark

This will open a regular Python interpreter imbedded with a Spark context (which is called **sc**).

To run a regular python file using Spark, use the following command:

./bin/spark-submit --master local[*num_threads*] my_program.py

The num_threads argument controls how many logical threads Spark will use for parallel computation.

For more information on using Python and Spark, look here:
https://spark.apache.org/docs/latest/programming-guide.html

Protip: If you want to reduce the amount of bloat in your output, put this line in your code after initializing your SparkContext:

sc.setLogLevel("ERROR")

MLLib

MLLib is a machine learning library for Spark (basically the scikit-learn for parallel machine learning algorithms), and you will be using it to train decision trees and random forests in this lab. MLLib comes packaged with Spark and examples for it can be found in the **examples/src/main/python/mllib/** directory. I would highly suggest you look at these examples when doing this lab.

Classification on the ACRENE Dataset

In the lab4 folder, there is a folder called "dataset" that contains the ARCENE dataset, which was in the NIPS 2003 machine learning competition. This is a labeled dataset of mass-spectrometric data, and all of the features are **continuous**. There are five files in the "dataset" folder:

- **dataset.pdf** – this file will tell you how to parse the dataset.
- **arcene_train.data** – this is the training dataset.
- **arcene_train.labels** – these are the labels the training dataset.
- **arcene_test.data** – this is the testing dataset.
- **arcene_test.labels** – these are the labels the testing dataset.

Your job: The task of this dataset is to distinguish cancerous vs regular patterns from mass-spectrometric data. You will train decision tree and random forest classifiers on the training dataset to minimize error on the testing dataset.

Your first step will be to clean the dataset. That is, you will need to read in the data and parse it. This dataset has no missing attributes, so you will not need to deal with that.

Part 1: Decision Tree

Now, you split your training dataset into a train and validate set. **Note that the test set is NOT the validation set!** The validation set is simply a subset of the training data. The test set is only for your final testing, when you are confident that your model is correct.

You may use any method to split your training data into a train and validation set. I would suggest using the randomSplit method. Look at the Python classification sample here to find out how: http://spark.apache.org/docs/latest/mllib-decision-tree.html

You may use other libraries to clean your data, but your model training must be done using Spark and MLLib!

Now, train a decision tree on the post-split training dataset.

In your report, answer the following questions:

- Fix the maxBins to 32 and leave the other parameters as default. What is the validation set accuracy using the Gini impurity and the accuracy using the Entropy impurity?
- Fix the impurity measure and vary the number of bins. Plot the validation set accuracy for atleast 10 different maxBin numbers.
- Fix the maxBins and the impurity measure. Vary the maxDepth from 1 to 5 and plot the accuracy.

Now, keep tweaking the parameters until your validation accuracy is as high as you think it will get. As in the previous lab, keep in mind that you can also remove features from your dataset. In this case removing features will be tricky since the features are not labeled, but you can try anyway. Then, with your final parameters, train your decision tree on the **entire train set** and run it on the test set.

In your report, answer the following questions:

- What were the final parameters you used for your model?
- With these parameters, what was the accuracy on the validation set?
- With the same parameters, what was the accuracy on the test set?

Part 2: Random Forests

Now, try using a random forest classifier to fit your training data. The documentation is here: http://spark.apache.org/docs/latest/mllib-ensembles.html#random-forests

In your report, answer the following questions:

- Plot the validation set accuracy as the number of trees changes. Keep increasing the number of trees until either the accuracy stops increasing or the training time goes over 20 minutes. Let **T** be the most number of trees you choose.
- Plot the training time of **T** trees as a function of the number of threads. Train with 1 upto **N** cores (where **N** is the number of threads your CPU is capable of). Remember that to train with **n** threads you use the following switch: --master local[**n**]

As in part 1, tweak parameters until your validation set accuracy is as high as you think it will get. Then, run your classifier on the test set.

In your report, answer the following questions:

- What were the final parameters you used for your model?
- With these parameters, what was the accuracy on the validation set?
- With the same parameters, what was the accuracy on the test set?