Running Spark KNN on AWS Documentation

Input: Unclassified data points D text file, classified data points C text file, EC2 master cluster URL U, Number of nearest neighbors K, HDFS output URL H

Step 1: Initialize spark context across U cluster's nodes

Step 2: Convert lines of D into D.rdd and lines of C into C.rdd

Step 3: Calculate the K nearest neighbors to each point in D.rdd from C.rdd and remember the classes from C.rdd for each point in D.rdd

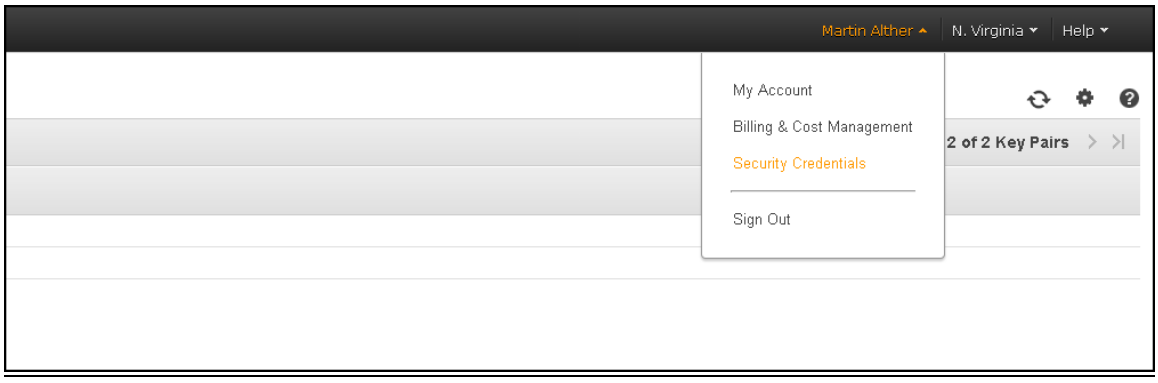
Step 4: For each point in D.rdd, find the most frequent class from its K Nearest Neighbors and assign that point in D.rdd that class to create newly classified E.rdd

Step 5: Convert E.rdd to hadoop partition and save it at the HDFS path H

Output: HDFS partitions of newly classified data

Preparing to use Amazon AWS

Step 1: Login to the AWS console. Under the account name at the top right, select security credentials.



Step 2: Click on the Access Keys tab and get or create the AWS access key id and AWS secret access key, then save them someplace for later.

Step 3: Under Services select ‘Ec2’. Click on Launch Instance button.

Step 4: Under the Quick Start column on the left,click on Community AMIs

Step 5: Search for “ubuntu 14.0”, select the first AMI “ubuntu-trusty-14.04-amd64-server-20140607.1 (ami-864d84ee)” and click next

Step 6: Select the “m3.medium” instance type under the general purpose family and click next.

Step 7: Next is 'Configure Instance' step, we do not want to make any changes to that so, click on “Next: Add Storage” button on the bottom right.

Step 8: You can change the Size of the instance based on your need, but we really don’t need to so keep the defaults. Click on “Next: Tag instance”; on bottom right.

Step 9: This is just to give a name to this instance, for now, give any name like 'spark_knn'. Click “Next: Configure Security Group”; on bottom right.

Step 10: Next comes the 'Configure Security Group', keep the default settings and click “Next: Review and Launch”.

Step 11: Just check that what we configured is showing up. Click on “launch”.

Step 12: In the 'Select existing Key pair or create a new key pair' dialog box, select create new key pair and name it. We gave the name 'dmkd_spark'. This key pair file will be used to login to the instance as well as the cluster we will set up. Acknowledge and click on Launch instance.

Step 13: If everything went well Launch Status page will show up with “Your instance is now launching” statement. Click on “View Instance” button on bottom right. You could see you instance in the running instances by the name 'spark_knn’.

Step 14: When you select the instance, Instance description shows up at the bottom. Copy the 'Public DNS' value, it will be something like 'ec2-.....-amazonaws.com'

Windows Login

Step 1: Open ‘Puttygen’ and in the dialog box click on ‘Load’ and select the keypair file which you downloaded in Step 12. In the search dialog box, select ‘All files’ at the bottom right which will show the ‘.pem’ file. Click on open and click ‘OK’ for successful import notice. Click on ‘Save private key’ at the bottom right and click ‘yes’ to the warning. Save the file with the same name and without the ‘.pem’ extension, Putty will automatically add the .ppk extension to the newly created file.

Step 2: Open Putty and in the ‘Host name’ type ‘ubuntu@<DNS value which you copied in Step 14>. Like ‘ubuntu@ec2-.....-amazonaws.com’.

Step 3: Under the connection category in the left panel, select the ‘+’ near ‘SSH’ and click on ‘Auth’. Browse for the ‘.ppk’ whichwe created in Step 12 and click open. If everything goes well, it will connect to the amazon instance.

Step 4: Now, open WinSCP and connect to the instance by passing ubuntu as the username and the DNS value from step 14 as the hostname. Click advanced, and on the left under SSH click on the tab called authentication. Select the .ppk you made as the private key file.

Step 5: After all that is set, you can now attempt to login. If everything goes well you will see the directories of the EC2 instance. Simply open the folder that contains your .ppk and .pem keypair files and drop them into the server under the ubuntu folder.

Installing Spark

Step 1: Once logged in on the Putty terminal, you will need to download and unpack the latest version of Spark from spark.apache.org/downloads.html using a command such as:

wget http://d3kbcqa49mib13.cloudfront.net/spark-1.0.2.tgz

tar –xvzf spark-1.0.2.tgz

Step 2: You will also need to have Java 7 installed by entering the comands:

sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update

sudo apt-get install oracle-jdk7-installer

And make sure Java 7 is installed with *java -version*

Step 3: Find the version of Scala that your version of Spark uses from either the README.md file in the Spark folder or from http://spark.apache.org/docs/latest

Step 4: Find the version of Scala that Spark is dependent on from http://www.scala-lang.org/download/all.html and then find the debian package for it, and then use a command like this to unpackage and install it:

wget http://www.scala-lang.org/files/archive/scala-2.10.4.deb

sudo dpkg –i scala-2.10.4.deb

Step 5: Download and install the Simple Build Tool(sbt) debian package from http://www.scala-sbt.org/0.13/tutorial/Installing-sbt-on-Linux.html using commands like:

wget http://dl.bintray.com/sbt/debian/sbt-0.13.5.deb

sudo dpkg –i sbt-0.13.5.deb

Step 6: Navigate into the Spark folder and execute the command "sbt/sbt assembly" to install Spark on the machine.

Launching the EC2 cluster

Step 1: Set environment variables for the AWS access key and secret access key that we saved in **Preparing to use AWS Step 2** with the commands:

export AWS_ACCESS_KEY_ID=<Access Key Here>
export AWS_SECRET_ACCESS_KEY=<Secret Access Key Here>

If they aren't set immediately later on, try *source .bashrc, source .bash_profile,* or *source .profile*

Step 2: In the Spark folder you had downloaded, navigate to the directory named "spark-1.0.0/ec2".

Step 3: Run the "spark-ec2" file with these arguments:

./spark-ec2 -k <keypair> -i <key-file> -s <num-slaves> --instance-type=<INSTANCE_TYPE> launch <cluster-name>

Where <keypair> is the name of the key pair we saved in **Windows login Step 5**,

<key-file> is the .pem file associated with that generated key pair

<num-slaves> is the number of slave instances to launch with the master instance

<INSTANCE_TYPE> is the type of instance to be launched

and <cluster-name> is the name of the cluster we give it and will work with from now on in the EC2 scripts. **We recommend using 3 as the number of slaves and r3.large as the instance type.**

```
Qwuke@VistaHome /cygdrive/c/Users/Qwuke/Documents/spark-1.0.0/spark-1.0.0/ec2
$ spark-ec2 -k MYKEYPAIR -i MYKEYPAIR.pem -s 2 --instance-type=r3.xlarge launch MYSPARKCLUSTER --resume
```

If everything works correctly, you will see the master and slave nodes launching and installing Spark automatically, and at the end it will print the URL of the master node. You may get an error for not setting your .pem private key file permissions to 600, or the launch script may time out after waiting for an instance to initiate. However, you can fix the permissions and run the same launch command from the spark-ec2 file and add the argument '--resume', which will rerun the launch command from where it left off.

Step 4: After the cluster has finished initializing, you can ssh into its root directory with:

./spark-ec2 -k <keypair> -i <key-file> login <cluster-name>

Step 5: To end the EC2 Spark cluster, ssh out of the cluster and use:

./spark-ec2 destroy <cluster-name>

Run this command before deleting anything else in the EC2 dashboard on AWS.

Running code or KNN on the cluster

Step 1: Use WinSCP with the user "root", the URL of the master node that was printed at the end of the EC2 launch script, and the .pem file for authentication to transfer your zipped spark code and data files.

Step 2: Now from the Putty terminal of the micro-instance, login to the cluster using **Step 4 of Launching the EC2 Cluster.** Once inside you will need to unzip the KNN code folder.

Step 3: The master node will now contain the directory with the code and data, so you will need to run a command in the terminal such as: *~/spark-ec2/copy-dir /KNNDir* to copy or Rsync the folder to all the slave nodes. If you are missing any of these files or things such as hadoop, you may have to logout and destroy the cluster and then restart it again.

Step 4: In order to pass a data file into a spark executable when it is being run on the master AND the slave nodes, you will have to load the data file into HDFS with the command *ephemeral-hdfs/bin/hadoop fs -put path/datafile.txt*. This will make the data accessible to the entire cluster. For the KNN program, you will need to do this for both the untrainedexamples.txt and trainedexamples.txt.

Step 5: Now that everything is ready, pass the spark executable to the spark-submit file in the Spark folder bin. You will also need to pass arguments for its main class and the URL of the master node to spark-submit. Example code: *\bin\spark-submit --class SparkKNN --master spark://ec2-55-5-55-5-5-55-5.compute-1.amazonaws.com:7077 KNNDir\knnspark.jar <K-number-of-nearest-neighbors> <trained datafile-name/hdfs location> <untrained datafile-name/hdfs location> <hdfs IP path/directory where the newly classified data will be stored>*. For the last argument, use something like *hdfs://master-ec2-node-url:9000/directorynameInHdfs* to save the outputs as hadoop partitions.

Step 6: This will run the Spark executable on all of the nodes. You can now extract the partitions from HDFS and examine the results by combining them using a command like: *hadoop fs -getmerge /directorynameInHdfs / /desired/local/output/file.txt*

Stopping the cluster

Step 1: Goto the Ec2 directory on your local machine from where you launched the cluster, in the terminal.

Step 2: Type the following command in the terminal:

$ ./spark-ec2 destroy <your cluster-name>

Cleanup (Important)

Note: If you use the destroy command you may not have to terminate your cluster's instances

Step 1: Logon to Amazon AWS and under Services select 'Ec2'.

Step 2: Under the 'Instances' tab in the left column; click on 'Instances'.

Step 3: Locate all your Hadoop instances and select them. On the top locate 'Actions' drop down button and click 'Stop' to stop the instances. You can start it and connect to the same settings whenever you want. If you terminate it, you have to create a new instance all together.