Big Data Classification: Hadoop and Machine Learning Study

Student: Junjun Ruan, CSC693-Big Data Analytics and Machine Learning[1]

Instructor: Shan Suthaharan, Department of Computer Science, UNC-Greensboro

Abstract

Big Data classification is used to predict the class of the new comings, and it is a big challenge for humans. As the data sets become very huge, they can't be analyzed efficiently in tradition applications; what's more, the data's high dimension, imbalance, inaccuracy and incompletion make the problem more difficult to be solved. Thus some new approaches need to be came up with to process the large data sets. Hadoop's MapReduce can be used to improve the performance of data processing. Many machine learning (ML) algorithms are proposed to deal with the classification problem. Thus machine learning using MapReduce is an efficient and promising way to solve the problem. This paper discusses the understanding of Hadoop and MapReduce. In particular this paper discusses different ML algorithms, applies SVM on two data sets: Iris Plant (IRIS) and Teaching Assistant Evaluation (TAE) data sets, and compares the performance on Hadoop environment with non-Hadoop environment.

Keywords

Big Data, Hadoop, MapReduce, machine learning, Support Vector Machine

1 Introduction

The term Big Data comes when the large collection of data can't be stored and processed by traditional applications. Since the data keeps growing everyday, many Big Data problems come out and new technologies are needed.

The first problem is related to the definition and characteristics of big data. It is more reasonable to determine the characteristics of big data in C^3 than in V^3 space because of cardinality, complexity and continuity[1].

The next challenge is how to manage such large amount of data. Firstly, more efficient network topology is required for Big Data analysis, although currently there is an integration of modern technology which consists of four units: user Interaction and learning system(UILS), Network Traffic Recording System(NTRS), HDFS and Cloud Computing Storage System(CCSS)[1]. Secondly, it is necessary to minimize the communication time between client and cloud server. Finally, since data is stored in public cloud, it is easy to be accessed by attackers thus causes the security problem.

Another challenge is concerned with the classification and learning of Big Data since sometimes the data is very complex. The traditional way to deal with it is machine learning. However, it has several problems since data is growing timely, but ML usually does a single learning task and fits on only one particular dataset. Among all the technologies in ML, the support Vector Machine technology performs better in accuracy. However it should be improved in the future. Representation Learning can help it in computation efficiency. On the other hand, machine lifelong learning(ML3) technology is needed considering with the long-term learning of big

[1]This course is developed and taught by Dr.Shan Suthaharan at the Department of Computer Science, University of North Carolina at Greensboro (UNCG) in Fall 2014. The development and delivery of the course is funded by the Center for the Science of Information, Purdue, University through a sub-award approved by the National Science Foundation, and partially funded by UNCG

data.

Recently Leon Bottou [2]and John Langford[3] came up with some technology to scaling up machine learning. In order to get the minimum of loss function, Bottou make a bold attempt, that is, using an inferior algorithm. It is surprising that estimation and approximation error were reduced, although optimization error was increased. In such way, Bottou improved predictive performance. Nowadays, as the amount of data from industries became extraordinary large, John came up a new method called feature hashingBrian, which is used to reduce feature dimensionality and then the approximation error.

The final problem is about user interaction with big data, as it will help machine lifelong learning technology in classification. There are two aspects of user interaction: data visualization and data uncertainty. They are both needed to be solved in the future.

This paper concentrates on solving Big Data classification problem for IRIS and TAE data sets. Section 3 shows the observation information of the two data sets; section 4 illustrates the configuration of Hadoop and the simple examples of MapReduce; section 5 discusses many kinds of machine learning algorithms such as Support Vector Machine (SVM), decision tree, random forest, deep learning and so on, implements SVM in R language and evaluates its performance on both Hadoop and non-Hadoop platform.

2 Background

Big Data classification problem has long been studied, it is used to identify which class the new comings belong to according to the learning of training data. One famous example is to determine a given email belong to “spam“ or “non-spam“ classes. Before dealing with the data, there are several things need to be known.

Firstly, it is important to clear the difference between classification and clustering. If the data sets have defined class labels, classification method can be used to separate data into classes. However, if the data sets do not have fixed labels, clustering should be used to group those have a close relationship. In the context of machine learning, classification is related to supervised machine learning, while clustering is related to unsupervised machine learning. In order to learn deeply in classification, this paper chooses two labelled datasets to analyze.

Secondly, many machine learning algorithm are proposed by researches, such as Support Vector Machine (SVM), decision tree, random forest, deep learning and so on. The original SVM algorithm was invented by Vapnik [12], then Mangasarian and Musicant [12] pointed out the Lagrangian Support Vector Machines (LSVM) based on Vapnik’s study, which is more optimal. The main idea of SVM and LSVM is to construct the hyperplane in a high space to get the larger margin to separate any class with lower error. Next is decision tree, it is a tree-like graph, choose one feature as a root, each branch stands for the outcome of the attribute testing, each internal node represents a "test" on an attribute and each leaf stands for a class label [14]. Based on decision tree, random forest was first proposed in the paper by Leo Breiman and Adele Cutler [15], which is a multitude of decision tree. Recently, a more complex but efficient algorithm was developed, which is called deep learning. It originated in the research of artificial neural network, which tries to imitate the way humans thinking. This paper discusses all these algorithms and implements LSVM to solve the classification problem.

3 Datasets

To be able to learn deeply in the classification, two data sets are chosen: IRIS [5] and TAE [7] data sets.

Before processing these data, it is important to check whether they are imbalanced, inaccurate or incomplete. If so, they should be to balanced, corrected and completed immediately.

After that, several useful statistical information of the datasets need to be extracted to help understand. Usually the statistical information includes number of observations, dimension of data, mean of each feature. Also some plots should be generated to have a good visualization.

According to the above two rules, the detail procedure of dealing with the two data sets are shown in the following two subsections.

3.1 Iris Plant Data

In IRIS data file, it has 150 instances and four columns represent four features, they are sepal length in cm, sepal width in cm, petal length in cm and petal width in cm. The last column is about class labels. All the instances are separated into three classes: Iris Setosa, Iris Versicolour and Iris Virginica.

Firstly, from Figure 1, it is easy to see the pie chat has 33.3% for each of 3 classes, so the data are already balanced.

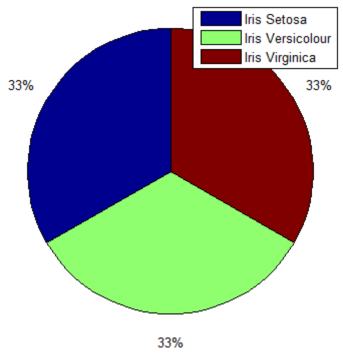


Figure 1: *Ratio of each classes*

Furthermore, the data is correct and complete, so it needn't to be processed any more. Then in Figure 2, instance 20 is picked up and all of its feature is to be analyzed.

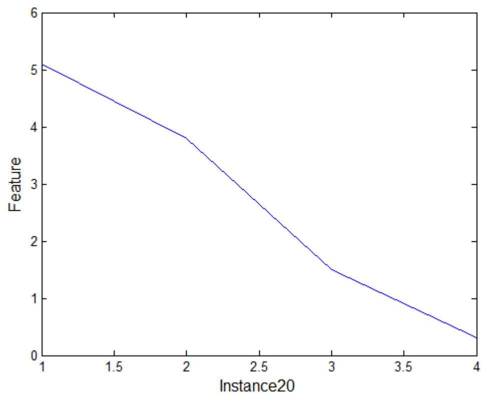


Figure 2: *Observe Features for Instance 20*

Finally, the mean and standard deviation values of each feature are calculated separately in Figure 3 and Figure 4.

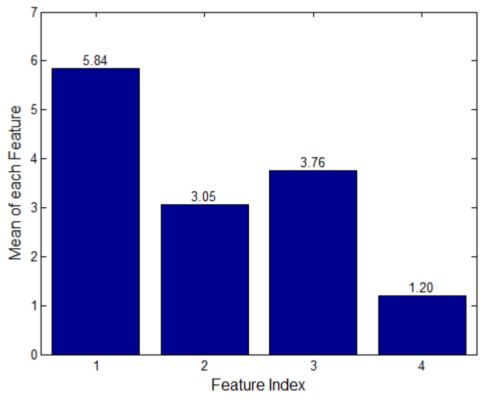


Figure 3: *Mean of Each Feature*

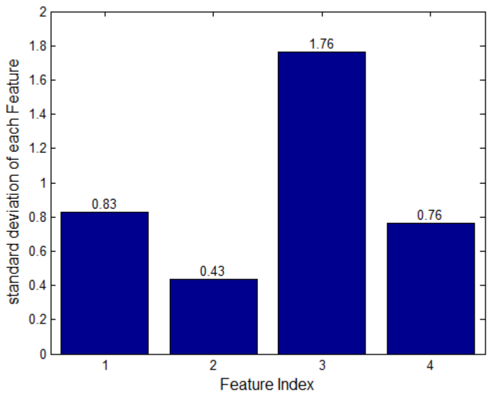


Figure 4: *Standard Deviation of Each Feature*

3.2 Teaching Assistant Evaluation Data

The data consists of 151 teaching assistant assignments at the statistics Department of the University of Wisconsin-Madison[6], each instance has five features (Whether of not the TA is a native English speaker, course instructor categories, course categories, summer or regular semester, class size) and one column of class labels(low, medium, high). Some useful statistical information from dataset is got.

In the first, the classes of the data are divided into three equal-sized categories; consequently, it needn't to be processed.

Moreover, instance 12 is chosen and all features of it is shown in Figure 5.

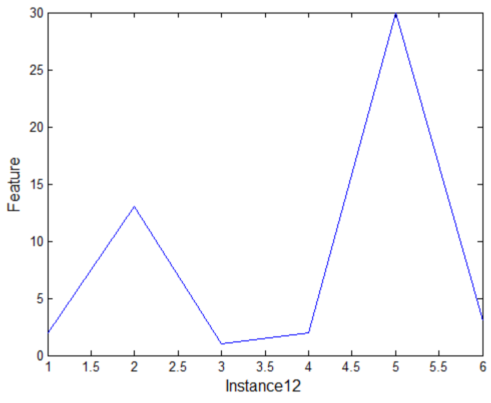


Figure 5: *Observe Features for Instance 12*

In the end, Figure 6 and Figure 7 show the mean and standard deviation of each feature.

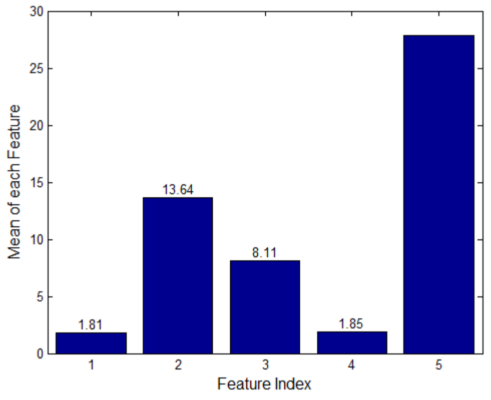


Figure 6: *Mean of Each Feature*

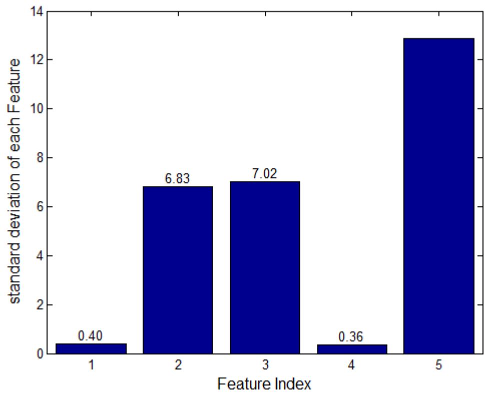


Figure 7: *Standard Deviation of Each Feature*

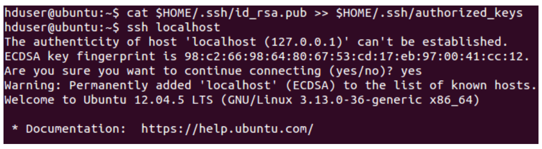
4 Computing Environment

Hadoop is a useful tool for storing and processing large amount of datasets. In this section, we will have a deep understanding of hadoop, including the procedure of installation hadoop, some MapReduce examples.

4.1 Hadoop Configuration

I had Ubuntu 12.04 LTS 64 bits on my computer. The following steps is mostly according to Piyush's personal website [8] and the tutorial on Michael G. Noll's website [9].

- Step1: Login to Ubuntu system, and add a new account name hduser for hadoop
- Step2: Install openJDK
  $ sudo apt-get install openjdk-7-jdk
- Step3: Verify JAVA version, the result is java version "1.7.0_25"
  $ java -version
- Step4: Create a symlink from 'java-7-openjdk-amd64' to 'jdk'
  $ cd /usr/lib/jvm
  $ sudo ln -s java-7-openjdk-amd64 jdk
- Step5: Install SSH
  $ sudo apt-get install openssh-client
  $ sudo apt-get install openssh-server
- Step6: Add hadoop group and user
  $ sudo addgroup hadoop
  $ usermod -a -G hadoop hduser
  To verify that hduser has been added to the group hadoop:
  $ groups hduser
- Step7: Configure SSH, and we don't need to enter password when use SSH in the future
  $ ssh-keygen -t rsa -P "" $ cat ~/.ssh/id_rsa.pub » ~/.ssh/authorized_keys
  $ ssh localhost



```
hduser@ubuntu:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
hduser@ubuntu:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is 98:c2:66:98:64:80:67:53:cd:17:eb:97:00:41:cc:12.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 12.04.5 LTS (GNU/Linux 3.13.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
```

Figure 8: *ssh locahost*

- Step8: Disable IPv6, since it will cause problems in Hadoop
  $ sudo gedit /etc/sysctl.conf

Add the following line to the end of the file:

```
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Check whether IPv6 is enabled, "1" means disabled

```
$ cat/proc/sys/net/ipv6/conf/all/disable_ipv6
```

- Step9: Download Hadoop from the Apache Download Mirrors website, the version I choose is hadoop 2.2.0, and save it in Downloads directory.

- Step10: Extract Hadoop and move it to /usr/local and Make sure to change the owner of all the files to the hduser user and hadoop group:

```
$ cd Downloads
$ sudo tar vxzf hadoop-2.2.0.tar.gz -C /usr/local
$ cd /usr/local
$ sudo mv hadoop-2.2.0 hadoop
$ sudo chown -R hduser:hadoop hadoop
```

- Step11: Open the .bashrc file to add several path, and then save and close the file:

```
$ cd
$ sudo gedit .bashrc
```

Add the following lines to the end of the file:

```
# Hadoop variables
export JAVA_HOME=/usr/lib/jvm/jdk/
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
# end of paste
```

- Step12: Open hadoop-env.sh to update the path of JAVA_HOME, and then save and close file:

```
$ sudo gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/jdk/
```

Restart the system and re-login

- Step13: Verify the Hadoop Version

```
$ hadoop version
```

The next few steps are to configure hadoop.

- Step14: Open core-site.xml file, and add the following lines between the <configuration> and </configuration> tags

```
$sudo gedit /usr/local/hadoop/etc/hadoop/core-site.xml
<property>
<name>hadoop.tmp.dir</name>
<value>/app/hadoop/tmp</value>
<description>A base for other temporary directories.</description>
</property>
<property>
<name>fs.default.name</name>
<value>hdfs://localhost:9000</value>
</property>
```

- Step15: Open yarn-site.xml file, and add the following lines between the <configuration> and </configuration> tags

```
$sudogedit/usr/local/hadoop/etc/hadoop/yarn-site.xml
```

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

- Step16: Open mapred-site.xml.template file to add the following lines between the <configuration> and </configuration> tags, and then save it as mapred-site.xml

$ sudo gedit /usr/local/hadoop/etc/hadoop/mapred-site.xml.template

```
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

- Step17: Make folders for namenode and datanode
  $ sudo mkdir -p $HADOOP_HOME/yarn_data/hdfs/namenode
  $ sudo mkdir -p $HADOOP_HOME/yarn_data/hdfs/datanode

- Step18: Open hdfs-site.xml.template file to add the following lines between the <configuration> and </configuration>

  $sudo gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml

```
<configuration>
<property>
<name>dfs.replication</name>
<value>1</value>
</property>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop/yarn_data/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop/yarn_data/hdfs/datanode</value>
</property>
</configuration>
```

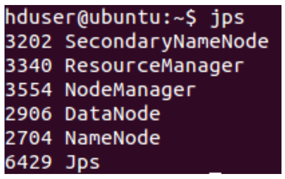
- Step19: Format the namenode with HDFS
  $ hdfs namenode -format

- Step20: Start Hadoop Services
  $ start-all.sh

  Here, I come cross an Error, ssh: connect to host localhost port 22: Connection refused. And it only occur in Ubuntu 64 bits system.

  The solution is to add the following two lines to the end of .bashrc file:
  exportHADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
  exportHADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"

- Step21: To check if hadoop is running
  $ jps
  Here is the result:



```
hduser@ubuntu:~$ jps
3202 SecondaryNameNode
3340 ResourceManager
3554 NodeManager
2906 DataNode
2704 NameNode
6429 Jps
```

Figure 9: *Hadoop process*

Next, we need to verify if HDFS can work.

- Step22: Create someFile.txt in home directory, paste any text you want in to the file and save it.

- Step23: Create Home Directory In HDFS
  $ hadoop fs -mkdir -p /user/hduser

- Step24: Copy file someFile.txt from local disk to the user's directory in HDFS
  $ hadoop fs -copyFromLocal someFile.txt someFile.txt

- Step25: Get a directory listing of the user's home directory in HDFS, here we can see someFile.txt
  $ hadoop fs –ls

- Step26: Display the contents of the HDFS file /user/hduser/someFile.txt
  $ hadoop fs –cat /user/hduser/someFile.txt

- Step27: Get a directory listing of the HDFS root directory
  $ hadoop fs –ls /

- Step28: Copy that file to the local disk, named as someFile2.txt
  $ hadoop fs –copyToLocal /user/hduser/someFile.txt someFile2.txt

- Step29: Delete the file from hadoop hdfs
  $ hadoop fs –rm someFile.txt

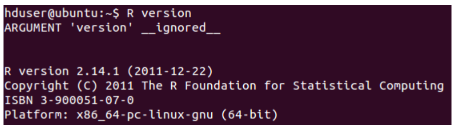
  Finally, we need to install R following the instruction of this website http://bighadoop.wordpress.com/2013/02/25/r-and-hadoop-data-analysis-rhadoop/

- Step30: Install R base package.
  $ sudo apt-get install r-base

  When checking R version, it shows 2.15.1, and it will cause error further in installing rJava package. In order to be compatible with its packages at follows, R version should be above 3.0. For Ubuntu 12.04 LTS, it can not be updated directly through this command$ apt-get update. As a matter of fact, we need to follow the instruction on this website `http://cran.r-project.org/bin/linux/ubuntu/`. But there comes a GPG error in the middle of the update process, then just go to the website of BIOINFORMATICS AT VIIKKI CAMPUS to fix it.

- Step31: Install RHadoop packages with their dependencies. rmr requires RCpp, RJSONIO, digest, functional, stringr and plyr, while rhdfs requires rJava. Download these package from Cran and Github website and then start to install. These packages are verified compatible very well by me.
  $ sudo R CMD INSTALL Rcpp_0.11.2.tar.gz
  $ sudo R CMD INSTALL codetools_0.2-9.tar.gz
  $ sudo R CMD INSTALL RJSONIO_1.3-0.tar.gz
  $ sudo R CMD INSTALL digest_0.6.4.tar.gz
  $ sudo R CMD INSTALL functional_0.6 .tar.gz
  $ sudo R CMD INSTALL stringr_0.6.2.tar.gz
  $ sudo R CMD INSTALL plyr_1.8.1.tar.gz
  Before install rmr, we need to install bitops, reshapre2 and caTools first.
  $ sudo R CMD INSTALL bitops_1.0-6.tar.gz
  $ sudo R CMD INSTALL reshape2_1.4.tar.gz
  $ sudo R CMD INSTALL caTools_1.17.1.tar.gz
  $ sudo R CMD INSTALL rmr2_2.3.0.tar.gz
  $ sudo JAVA_HOME=/usr/lib/jvm/jdk/jre R CMD javareconf
  $ sudo R CMD INSTALL rJava_0.9-6.tar.gz
  $ sudo HADOOP_CMD=/usr/local/hadoop/bin/hadoop R CMD INSTALL rhdfs_1.0.8.tar.gz

  In this way, R is installed and should be compatible with HADOOP. Check its version by the following way:
  $ R version Here is the result:



```
hduser@ubuntu:~$ R version
ARGUMENT 'version' __ignored__


R version 2.14.1 (2011-12-22)
Copyright (C) 2011 The R Foundation for Statistical Computing
ISBN 3-900051-07-0
Platform: x86_64-pc-linux-gnu (64-bit)
```

Figure 10: *Check R version*

It is well known that Hadoop Distributed File System (HDFS) and the MapReduce framework can work compatibly. HDFS is used for restoring files. The advantage is that it has high speed performance. For MapReduce, the main concept is (k,value) pattern. It will be explained in next section. The following diagram shows how they work.

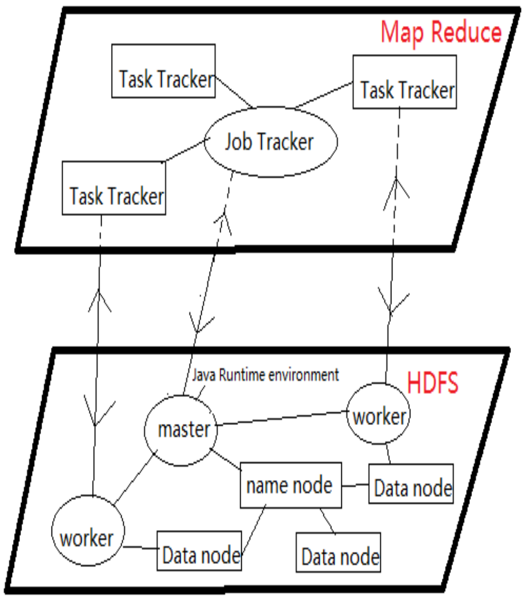


Figure 11: *MapReduce Framework*

Since it is important for us to know some directory, so I draw the directory structure diagram. The highlighted red directory are very important.

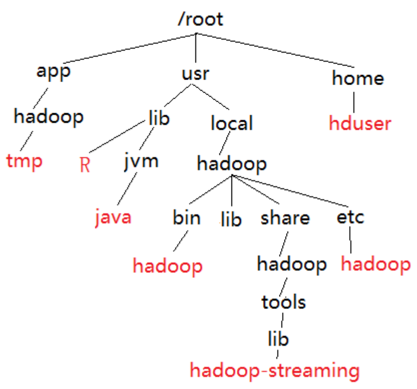


Figure 12: *Hadoop Directory Structure*

In this tree, from left to right, HDFS file storage directory are the child of tmp directory. R language directory is the child of lib directory. Java_Home environment variable is under jvm directory. Hadoop executable file is in the hadoop directory under bin directory. The configuration files of Hadoop like core-site.xml and yarn-site.xml are all in the hadoop directory under etc directory.

4.2 Programming Example

I choose R language as my programming language, because R is designed for statistical analysis, and it also has outstanding graph function. Here is an example of R programming, read data from IRIS data set, and then split it into separate files according to class labels. As it mentioned in section 3, this data set has three labels. So finally we can get three individual files named by labels, they are Rsplit-Iris-setosa.txt,Rsplit-versicolor.txt,Rsplit-virginica.txt. The codes file is Rsplit.r. It runs without Hadoop environment.

4.3 MapReduce Examples

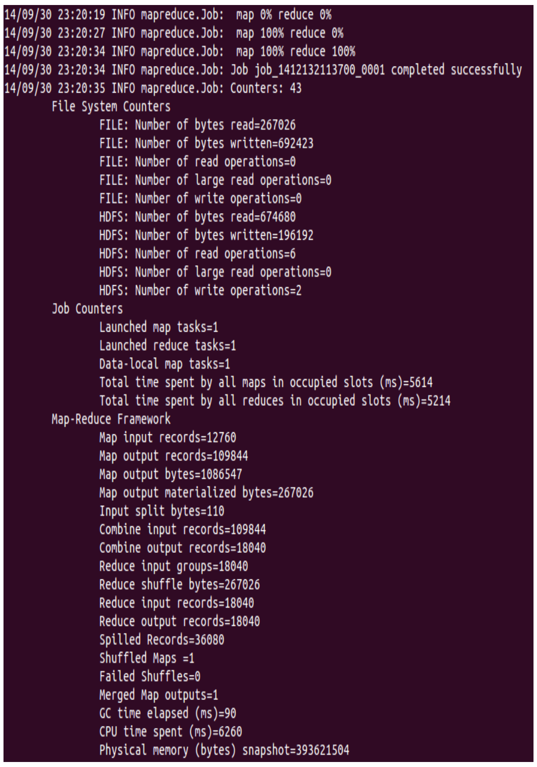
In order to understand the concept of MapReduce deeply, I chose two examples in the following subsections. One is a famous example called wordcount, the other one is about splitting file, which is working on my selected data sets above.

4.3.1 Standard Example

Wordcount is one of the basic examples in MapReduce Framework, and it is in Hadoop installation package. The main idea of wordcount is to count the frequency of words. Here is the steps about how to run this example.

Firstly, download an ebook “The Outline of Science“ from website, and store it in tmp directory.

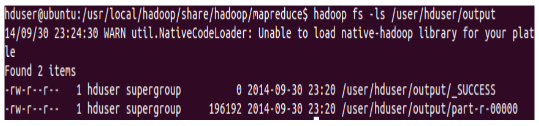
Secondly, use following commands to copy local example data to HDFS and execute the examples.
$ hadoop fs -mkdir -p /user/hduser
$ hadoop fs -copyFromLocal /tmp /user/hduser
$ hadoop fs -ls /user/hduser
$ hadoop jar hadoop*examples*.jar wordcount /user/hduser /user/hduser/output



```
14/09/30 23:20:19 INFO mapreduce.Job:  map 0% reduce 0%
14/09/30 23:20:27 INFO mapreduce.Job:  map 100% reduce 0%
14/09/30 23:20:34 INFO mapreduce.Job:  map 100% reduce 100%
14/09/30 23:20:34 INFO mapreduce.Job: Job job_1412132113700_0001 completed successfully
14/09/30 23:20:35 INFO mapreduce.Job: Counters: 43
        File System Counters
                FILE: Number of bytes read=267026
                FILE: Number of bytes written=692423
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=674680
                HDFS: Number of bytes written=196192
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=5614
                Total time spent by all reduces in occupied slots (ms)=5214
        Map-Reduce Framework
                Map input records=12760
                Map output records=109844
                Map output bytes=1086547
                Map output materialized bytes=267026
                Input split bytes=110
                Combine input records=109844
                Combine output records=18040
                Reduce input groups=18040
                Reduce shuffle bytes=267026
                Reduce input records=18040
                Reduce output records=18040
                Spilled Records=36080
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=90
                CPU time spent (ms)=6260
                Physical memory (bytes) snapshot=393621504
```

Figure 13: *Wordcount Example*

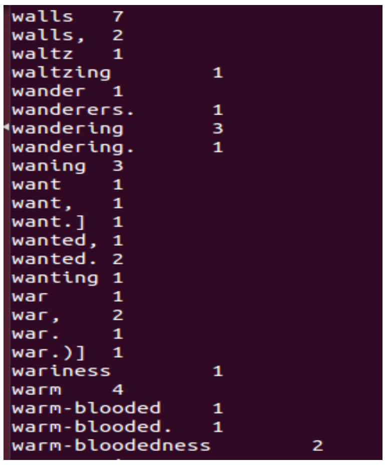
$ hadoop fs -ls /user/hduser/output



```
hduser@ubuntu:/usr/local/hadoop/share/hadoop/mapreduce$ hadoop fs -ls /user/hduser/output
14/09/30 23:24:30 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your plat
le
Found 2 items
-rw-r--r--   1 hduser supergroup          0 2014-09-30 23:20 /user/hduser/output/_SUCCESS
-rw-r--r--   1 hduser supergroup     196192 2014-09-30 23:20 /user/hduser/output/part-r-00000
```

Figure 14: *Output File Name*

$ hadoop fs -cat /user/hduser/output



```
walls   7
walls,  2
waltz   1
waltzing        1
wander  1
wanderers.      1
wandering       3
wandering.      1
waning  3
want    1
want,   1
want.]  1
wanted, 1
wanted. 2
wanting 1
war     1
war,    2
war.    1
war.)]  1
wariness        1
warm    4
warm-blooded    1
warm-blooded.   1
warm-bloodedness        2
```

Figure 15: *Output File Content*

In this example, the first phase is mapper. All the content is read line by line. Each mapper takes a line and break it into words. So for each line, it forms (key,val) pattern, each word is key and value is '1'. For example, there are one line in the file is 'to waste its venom on creatures it does not want.]', convert it to (key,val) pair will be as follows:
<"to",1>
<"waste",1>
<"its",1>
<"venom",1>
<"on",1>
<"creatures",1>
<"it",1>
<"does",1>
<"not",1>
<"want.]",1>
After mapper procedure is done, there comes the reduce phase. All the keys will be grouped together, and then sum up the value of the same key. Thus got figure 15.

4.3.2 My Example

For file-splitting problem, we begin to work on Hadoop environment, that is, using rmr2 and rhdfs library. In IRIS and TAE data sets, they have label in the fifth and sixth column separately. Thus it is easy to split the data into individual files according to labels. Finally, we got Iris-setosa.txt,versicolor.txt,virginica.txt for IRIS data sets, and 1.txt, 2.txt,3.txt for TAE data sets. The code files name are iris.r and tae.r.

It should have good performance when working on Hadoop environment with big data sets. However, the data sets here are very small, we can not see big difference about it in section 4.1 (without Hadoop environment).

5 Machine Learning

During data processing, data sets are usually split into two parts, one is called training data, and the other is testing data. Apply learning methods to training data so that functions can be got to test the testing data. There should be some restriction on the function so that it can work well on both training data and testing data with minimum error; otherwise, it may work well on training data but not well on testing data. The procedure of learning methods from data is the so called Machine Learning(ML). There is growing interest in developing different kinds of ML algorithms, such as Support Vector Machine(SVM), decision tree, random forest, deep learning and so on.

5.1 Types of Machine Learning

Support vector machine is used for classification and regression. For a given training data set as input, in order to separate them into two classes, SVM recognize the training pattern, develop a module to predict the belongings of the new coming ones. The training pattern $w = \sum_i v_i x_i$ is called support vectors[10], where the name Support Vector Machine comes from. SVM is of good usability, since it is based on statistical learning theory, on the other hand, it have several classes that is strong enough to handle special cases in practice, such as neural nets, radial basis function(RBF) nets, and polynomial classifiers [10] and so on. Based on SVM, Lagrangian proposed a more optimal algorithm called

Lagrangian Support Vector Machines(LSVM) [12], which will be applied to our data sets later.

SVM can be applied to both linear classification(see Figure 16) [13] and non-linear classification(see Figure 17)[13]. For linear classification, the original optimal hyperplane algorithm can be applied to get the maximum margin; For non-linear classification, the input can be regarded as relative linear when mapping into high-dimension space with kernel trick computing doc product. For example, if two classes are hard to be classified in 2D space as shown in Figure 17 [13], but when they are plotted in 3D space (in Figure 18) [13], they are easy to be discriminate in visualization. The key point is to find the kernel methods for transmitting. In this simple example, the transformation is $[x, y] = [x^2, xy, y^2]$. Also, by the choice of kernel functions, different machine learning can be got. Actually, kernels can be used widely in other algorithms related to doc product, one of them is principal component analysis(PCA)[10], which will be demonstrated in the next subsection.

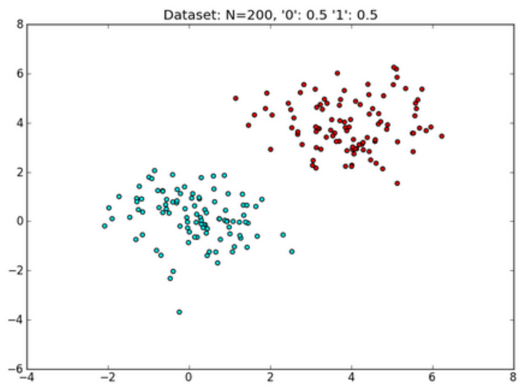


Figure 16: *Linear Classification*

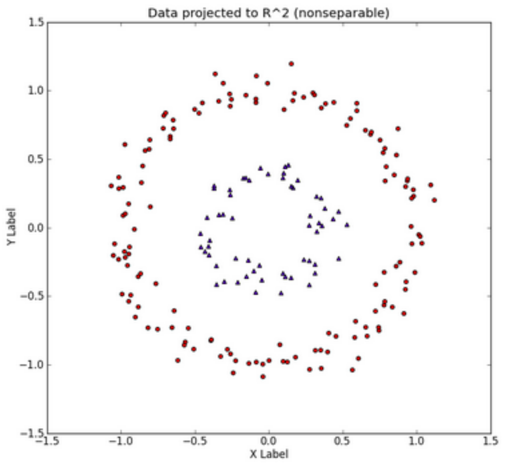


Figure 17: *Non-linear Classification in 2D graph*

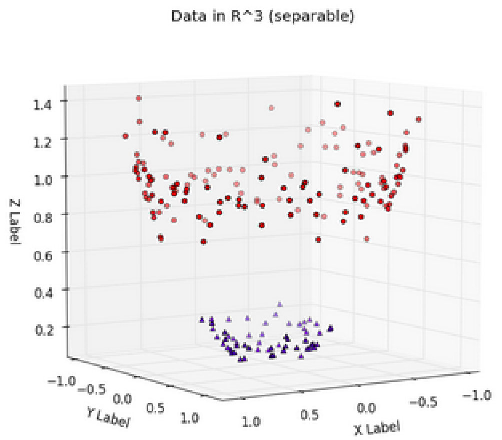
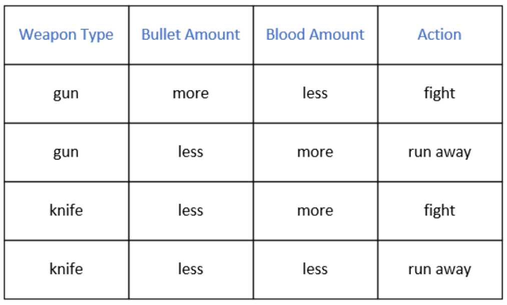


Figure 18: *Non-linear Classification in 3D graph*

SVM can be applied to different cases. One application is for testing text categorization, which was proposed by Susan Dumais[11]. In the past, most text categorization is done by humans; what's more, an advanced way is to construct rules by humans. Both of them are time consuming and not automatic. Now with inductive-learning techniques [10], test categorization can be done automatically and efficiently. In this method, feature selection is very a very important aspect, Dumais use the mutual information between each feature as input to the SVM [10] so that can reduce feature space. Another famous application is face-detection, which is done by Edgar OSuna[10]. The face detection problem is to determine if any human face is existed in the given digital image. SVM plays an important role in classification with a kernel function to reduce training error. Furthermore, the SVM approach can be widely applied to detect other objects in digital images.

The other ML method decision tree, it is used for computing if the strategy reach the goal and calculating conditional probabilities, which is measured by math terminologies, such as Entropy and Gini Index. Entropy is used for measuring the radio of information amount containing; Gini Index is a measurement of error, the big value indicates less error. A simple example related to CS Game is shown below about how to build a decision tree and how to compute entropy. In Figure19, according to the formula: $Entropy = -\sum p_i log p_i, E = -\sum_{i=1}^{2} p_i log_2 p_i = -\frac{1}{2}log\frac{1}{2} - \frac{1}{2}log\frac{1}{2} = 1$. This situation is of maximum uncertainty and it is most difficult to predict the outcome of next coming one. Next, according to Gini Index formula : $GI = 1 - \sum p_i^2, GI = 1 - ((\frac{1}{2})^2 + (\frac{1}{2})^2) = \frac{1}{2}, GI(weapontype = gun) = 1 - ((\frac{1}{2})^2 + (\frac{1}{2})^2) = \frac{1}{2}, GI(weapontype = knife) = 1 - ((\frac{1}{2})^2 + (\frac{1}{2})^2) = \frac{1}{2}, thusGI(weapontype) = \frac{1}{2} \times \frac{1}{2} + \frac{1}{2} \times \frac{1}{2} = \frac{1}{2}, weapontype(Informationgain) = \frac{1}{2} - \frac{1}{2} = 0$. In this way, we can easily get $GI(bulletamount) = \frac{1}{3}, bulletamount(Informationgain) = \frac{1}{2} - \frac{1}{3} = \frac{1}{6}; GI(bloodamount) = \frac{1}{2}, bloodamount(Informationgain) = 0$. From the above computation, we can get less error with bullet amount, so it should be the root in decision tree, as in Figure 20. What's more, it is easy to see if using bullet amount, in the first level of tree, with condition of more bullet amount, we can get class label fight. So in decision tree, the choosing of root can be a big issue, with the right root, the tree can be of less level.



Weapon Type	Bullet Amount	Blood Amount	Action
gun	more	less	fight
gun	less	more	run away
knife	less	more	fight
knife	less	less	run away

Figure 19: *CS Game Table*

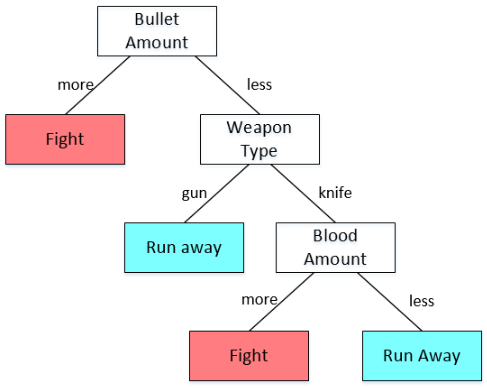


Figure 20: *Decision Tree*

Another ML approach is random forest; it operates by constructing of multiple decision tree, it is developed by Leo Breiman [15]. The main idea behinds random forest is first sampling training set with replacement, then transferring to trees.

The relatively new ML approach is deep learning and the main idea of it is to use many layers of feature extraction and transformation, and finally get the module for classification. The key point is to find the weight for transformation. The three approaches of deep learning are: No-Drop, DropOut and DropConnect [16], they are shown in Figure 21 22 23, which are from this website[17]. In DropOut, subset of activations were randomly selected to set as zero in each layer; while in DropConnect, subset of weights were randomly selected to set as zero [17]. Both of them reduced overfitting and increased performance.

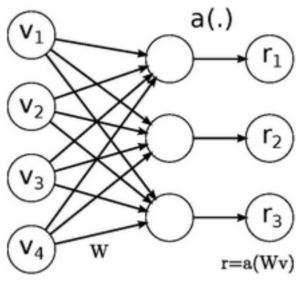


Figure 21: *No-Drop*

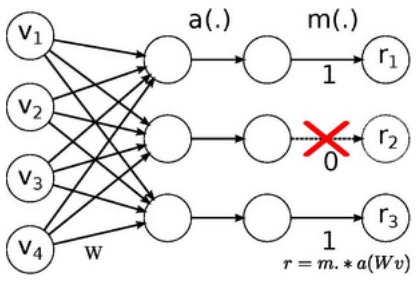


Figure 22: *DropOut*

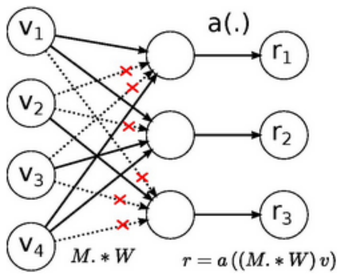


Figure 23: *DropConnect*

5.2 SVM On Real Data Sets

This paper uses SVM to process the two data sets. Firstly, in order to analyze IRIS data set, the following steps of tasks should be done.

- Step 1: Read the IRIS data set file, split it according to class labels and store them separately with their corresponding class labels. It is done in the fourth section. The three files are: Iris-virginica.txt, Iris-setosa.txt and Iris-versicolor.txt.

- Step 2: Increase the size of each data set(each class file) by adding 5000 of new observations randomly. In a laptop, Hadoop cannot handle real big data; if increase too much data, when running hadoop, it will make the laptop out of memory.

- Step 3: Select any two class files randomly and merge them. Compare the performance on hadoop with non-Hadoop. My programming randomly selected two class files: Iris-virginica.txt and Iris-setosa.txt. On hadoop environment, it takes 46.24 seconds; while on non-hadoop environment, it takes 0.13 seconds.

- Step 4: Split the merged data set in Step 3 into multiple files according to the features and store them separately with each feature as file name. Since the IRIS data set has four features, we get f1.txt, f2.txt, f3.txt and f4.txt.

- Step 5: Produce two-dimensional and three-dimensional plots using several combinations of two features and three features respectively. Once again, do this on both Hadoop and non-Hadoop environments. We choose feature 1 and 2 to do the two-dimensional plot, feature 1,2 and 3 to do the three-dimensional plot. The plots are shown in Figure 24 and 25 separately. Obviously, in 2-D plot, it is not easy to classify the class Iris-virginica and Iris-setosa; while in 3-D plot, it is easy to discriminate them. The separate running time on hadoop and non-hadoop is 1.46 seconds and 56.55 seconds.

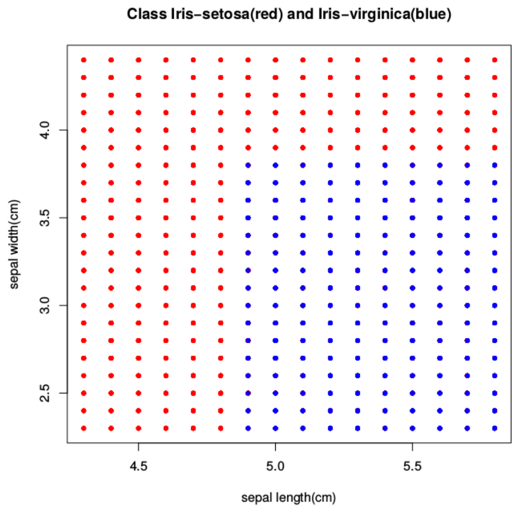


Figure 24: *Feature 1 and 2 in 2-D plot*

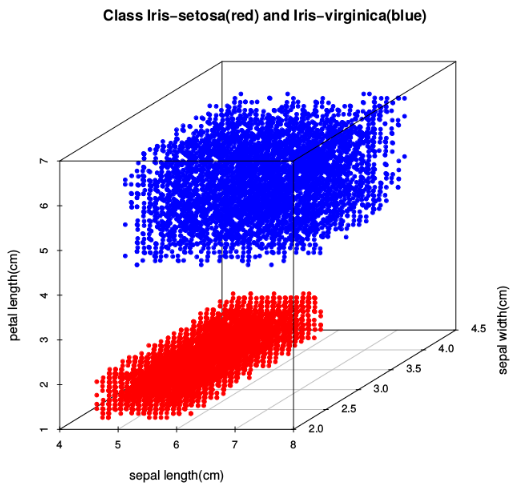


Figure 25: *Feature 1,2 and 3 in 3-D plot*

- Step 6: Select a few combinations of features based on visual interpretations, and classify them using the SVM implementations, and compare the performance on Hadoop and non-Hadoop environment. We chose the same classes as Step 3, the procedure of our programming of SVM is:

  (a) Read the file containing feature 1,2 and corresponding label, extract features and labels, then store them into two separate matrices. Set one class label as 1, and the other one as -1.

  (b) For the feature matrix, get its dimension: m is the number of rows, n is the number of columns, then normalize the two features.

  (c) Next, append $m \times n$zeros matrix to the right side of the normalized matrix, so combined matrix's dimension is $m \times (2n)$. Then append append a $m \times (2n)$ identity matrix right below the combined matrix, now the dimension of the combined matrix is $(m + 2n) \times (2n)$, and then replace the left most corner $n \times n$ small matrix with the same size of negative identity matrix, we call this matrix B_1.

  (d) On the other hand, we generate a $(2n) \times (2n)$ zero matrix, and replace the left corner $n \times n$ small matrix with the same size of identity matrix $\times \lambda, \lambda = 0.95$ is chosen, after applying sqrt() function to it, we get matrix B_2. Next, multiply B_1 and B_2, get matrix A.

  (e) Next, for the label matrix, add 2n rows of ones to it, so that its dimension can become $(m + 2n) \times 1$, we call it P_e matrix.

  (f) Generate a $(m + 2n) \times (m + 2n)$ dimension zeros matrix D with its diagonal value equals to the corresponding value in P_e matrix.

  (g) In P_e matrix, if the values are the label in original class, set them to 1, otherwise, set them to 0. Here, the last 2n columns values should be changed to 0. The new matrix is called matrix e.

  (h) Next, append -e matrix to the right side of matrix A, get the new matrix C.

  (i) Multiply matrix C and D, we get H.

  (j) Finally, based on Lagrangian's paper's code 2 [12], with input matrix A,D, nu = 1,itmax = 1000, and tol = 0.0001, output w and γ. $w1$ is the sum of the first and third column in w, $w1$ is the sum of the second and fourth column in w. In our example, $w1 = 0.97, w2 = -0.48, \gamma = -0.09$. With the three values, it is easy to draw a line, which consists of several points, to separate two classes.

  The result of applying LSVM to our data set is shown in Figure 26, allowing few errors. The separate running time on Hadoop and non-Hadoop is 10.33 seconds and 50.62 seconds. In all above steps related to comparison of Hadoop and non-Hadoop environment, it does not work efficiently with Hadoop because the data is not big enough to make a distinguish.

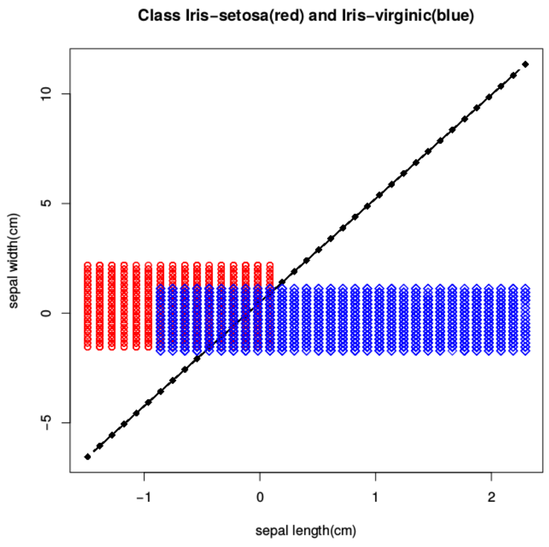


Figure 26: *Classification in SVM*

Then, we need to apply this SVM function to our testing data to see if it works well or not. The testing data set has 500 instances for each feature, and the testing result, which is in Figure 27, shows that the SVM function performances similar as in the training set.

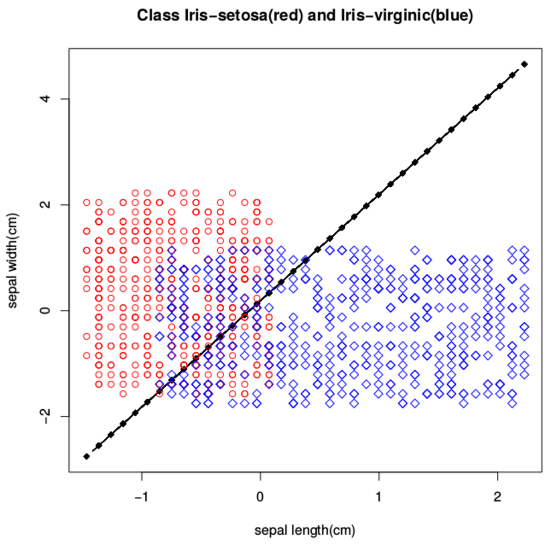


Figure 27: *Testing SVM function on the testing data set*

On the other hand, for TAE data set, do the above six steps. We chose feature 2(course instructor categories) and 3(course categories), class 2 (medium) and 3(high). By comparing the efficiency of Hadoop with that of non-Hadoop environment during step 3, 5 and 6, non-Hadoop always performances better. In step 5, the 2-D and 3-D plots are shown in Figure 28 and 29 respectively. In this data set, the class 2 and 3 are hard to be separated even in 3-D plot.

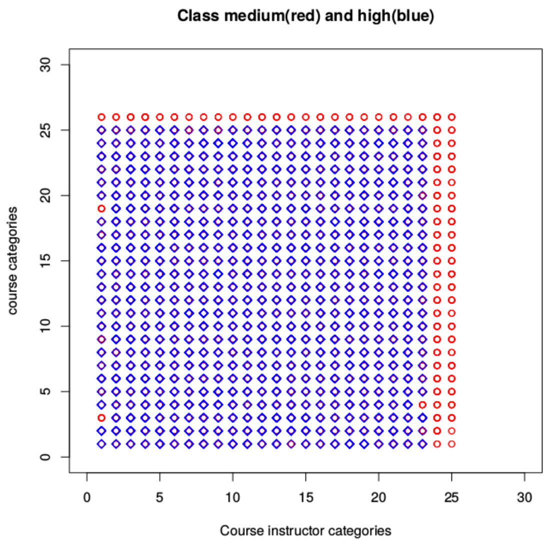


Figure 28: *Feature 2 and 3 in 2-D plot*

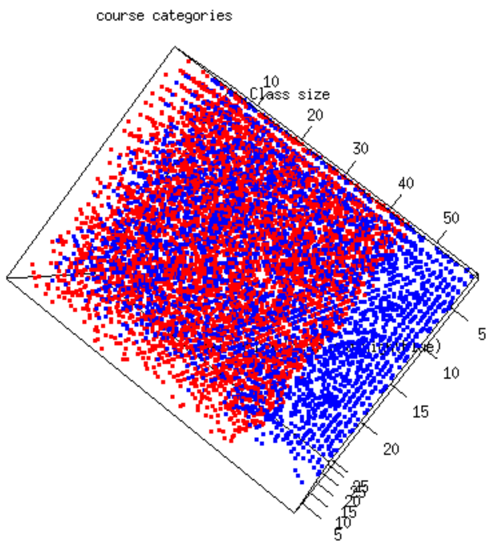


Figure 29: *Feature 2,3 and 5 in 3-D plot*

After applying LSVM, we get the result shown in Figure 30. However, LSVM doesn't work well, since points of the two classes intersect in a large common areas. We may need to try the other ML methods in the future. Since LSVM function does not performance well in training data set, it is meaningless to be tested on testing data.

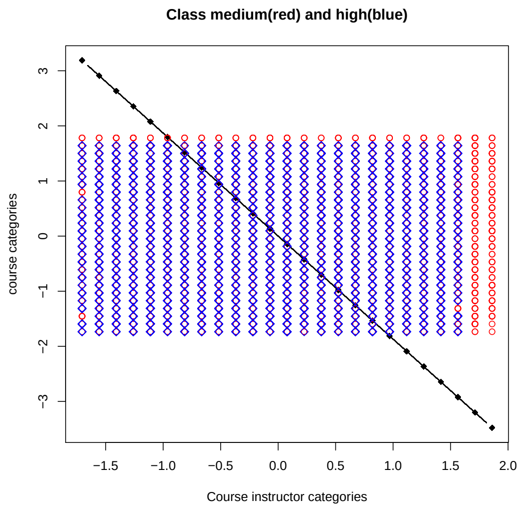


Figure 30: *Classification in SVM*

5.3 Scaling Up Machine Learning

Along with the popularity of social networking, data keeps growing faster, scalability becomes a big issue for machine learning. There are several methods to deal with this problem: Principal component analysis (PCA), feature hashing, stochastic gradient descent (SGD) and so on.

The main idea of PCA is to reduce dimension using orthogonal transformation and get more information from some perspective. In Figure 31, in the first sight, ignore u-v coordinate, the ellipse is lopsided, and it is hard to discriminate data inside ellipse according to x-y coordinate since the data points variation in x-y coordinate is not so big. But if look the ellipse corresponding to u-v coordinate, we can get more information based on x axis, thus it is easy to separate data according to x axis.

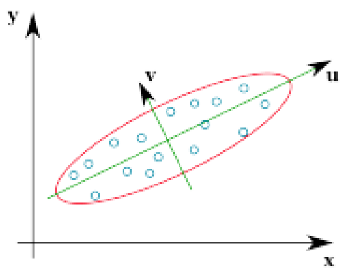


Figure 31: *PCA*

Feature hashing is applied in many areas, one typical example is document classification, which can be used to illustrate how feature hashing works. In the past, a term-document matrix is regarded as the input in machine learning algorithm, each row of the matrix represents a single document and each column of it stands for one word [18]; consequently, a dictionary which represents vocabularies in the training data set is built. However, this method has some disadvantages: one is the dictionary costs a lot of storage space, the other is not flexible for the new coming words, since the word is fixed. Feature hashing overcomes the two shortages. Instead of building a dictionary, feature hashing generates a vector of length N, applies a hash function to the features to get hash values h, then by using $h mod N$ as indices of the vector to control its length to N. When coming a new word, the vector will be updated by applying hash function to it, without increasing its length. That's the core idea of feature hashing.

6 Conclusion

In conclusion, we chose two data sets to observe, illustrated the detail procedure of Hadoop installation, tested some simple MapReduce on it with R language; what's more, we learned several ML methods, completed SVM implementation, and applied it to our data sets. It performanced pretty well on Iris data set, but not well on TAE data set.

Acknowledgement

In the end, I'm using this opportunity to thank UNCG, which is my school. It afford me enough materials and good study environment so that I can take this cutting edge class: Big Data. Thanks to my teacher Dr.Shan, the processor who ever visited in University of California, Berkeley, brought us these fresh knowledge, explained clearly for us to understand and gave me a hand whenever I needed help. Finally, thanks to my classmate Stephen, who helped me a lot during discussion.

References

[1] S.Suthaharan. Big data classification: Problems and challenges in network intrusion prediction with machine learning. *ACM SIGMETRICS Performance Evaluation Review*, 41(4):70–73, 2014.

[2] L.Bottou and O.Bousquet. The Tradeoffs of Large Scale Learning, *Advances in Neural Information Processing Systems*, 20:161-168, 2008.

[3] K.Weinberger, A.Dasgupta,J.Langford, et al. Feature hashing for large scale multitask learning,*Proceedings of the 26th Annual International Conference on Machine Learning*, Pages 1113-1120. ACM, 2009.

[4] B.Dalessandro. Bring the noise. *Big Data*, 1(2):110-112, June 2013.

[5] Iris Data Set. Available from: `https://archive.ics.uci.edu/ml/datasets/~Iris`

[6] W.Loh and Y.Shih. Split Selection Methods for Classification Trees. *Statistica Sinica*, 815–840, 1997.

[7] Teaching Assistant Evaluation Data Set. Available from: `https://archive.ics.uci.edu/ml/datasets/Teaching+Assistant+Evaluation`

[8] P.Agarwal. Available from: `http://blog.impiyush.com/2013/11/hadoop-220-installation-steps-for.html`

[9] Michael G.Noll. Available from: `http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-`node-cluster/

[10] M.A.Hearst, et al. Support Vector Machines. *IEEE Intelligent Systems*,13(4):18-28, July 1998.

[11] T.Hastie, R.Tibshirani el at. The Elements of Statistical Learning:Data Mining, Inference, and Prediction(2nd. ed.). *Springer*, New York, 2009.

[12] O.L.Mangasarian and D.R.Musicant. Lagrangian Support Vector Machines. *Journal of Machine Learning Research*,1:161-177, September 2001.

[13] Figures from: `http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html`

[14] Wikipedia, from: `http://en.wikipedia.org/wiki/Decision_tree`

[15] L.Breiman and A.Cutler, from `https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#workings`

[16] L.Wan, M.Zeiler et al. Regularization of Neural Networks using DropConnect. *Sanjoy Dasgupta and David Mcallester*, 28(3):1058-1066, 2013.

[17] Figures from: `http://cs.nyu.edu/~wanli/dropc/`

[18] Wikipedia, from: `http://en.wikipedia.org/wiki/Feature_hashing`