Partial Implementation

Introduction

This quick start page shows how to build a decision forest using the partial implementation. This tutorial also explains how to use the decision forest to classify new data.
Partial Decision Forests is a mapreduce implementation where each mapper builds a subset of the forest using only the data available in its partition. This allows building forests using large datasets as long as each partition can be loaded in-memory.

Steps

Download the data

- The current implementation is compatible with the UCI repository file format. In this example we'll use the NSL-KDD dataset because its large enough to show the performances of the partial implementation.
  You can download the dataset here http://nsl.cs.unb.ca/NSL-KDD/
  You can either download the full training set "KDDTrain+.ARFF", or a 20% subset "KDDTrain+_20Percent.ARFF" (we'll use the full dataset in this tutorial) and the test set "KDDTest+.ARFF".
- Open the train and test files and remove all the lines that begin with '@'. All those lines are at the top of the files. Actually you can keep those lines somewhere, because they'll help us describe the dataset to Mahout
- Put the data in HDFS:

```
$HADOOP_HOME/bin/hadoop fs -mkdir testdata
$HADOOP_HOME/bin/hadoop fs -put <PATH TO DATA> testdata
```

Build the Job files

- In $MAHOUT_HOME/ run:

```
mvn clean install -DskipTests
```

Generate a file descriptor for the dataset:

run the following command:

```
$HADOOP_HOME/bin/hadoop jar $MAHOUT_HOME/core/target/mahout-core-<VERSION>-job.jar
org.apache.mahout.classifier.df.tools.Describe -p testdata/KDDTrain+.arff -f
testdata/KDDTrain+.info -d N 3 C 2 N C 4 N C 8 N 2 C 19 N L
```

The "N 3 C 2 N C 4 N C 8 N 2 C 19 N L" string describes all the attributes of the data. In this cases, it means 1 numerical(N) attribute, followed by 3 Categorical(C) attributes, ...L indicates the label. You can also use 'I' to ignore some attributes

Run the example

```
$HADOOP_HOME/bin/hadoop jar
$MAHOUT_HOME/examples/target/mahout-examples-<version>-job.jar
org.apache.mahout.classifier.df.mapreduce.BuildForest -Dmapred.max.split.size=1874231
-d testdata/KDDTrain+.arff -ds testdata/KDDTrain+.info -sl 5 -p -t 100 -o nsl-forest
```

which builds 100 trees (-t argument) using the partial implementation (-p). Each tree is built using 5 random selected attribute per node (-sl argument) and the example outputs the decision tree in the "nsl-forest" directory (-o).
The number of partitions is controlled by the -Dmapred.max.split.size argument that indicates to Hadoop the max. size of each partition, in this case 1/10 of the size of the dataset. Thus 10 partitions will be used.
IMPORTANT: using less partitions should give better classification results, but needs a lot of memory. So if the Jobs are failing, try increasing the number of partitions.

- The example outputs the Build Time and the oob error estimation

```
10/03/13 17:57:29 INFO mapreduce.BuildForest: Build Time: 0h 7m 43s 582
10/03/13 17:57:33 INFO mapreduce.BuildForest: oob error estimate :
0.002325895231517865
10/03/13 17:57:33 INFO mapreduce.BuildForest: Storing the forest in:
nsl-forest/forest.seq
```

Using the Decision Forest to Classify new data

run the following command:

```
$HADOOP_HOME/bin/hadoop jar
$MAHOUT_HOME/examples/target/mahout-examples-<version>-job.jar
org.apache.mahout.classifier.df.mapreduce.TestForest -i nsl-kdd/KDDTest+.arff -ds
nsl-kdd/KDDTrain+.info -m nsl-forest -a -mr -o predictions
```

This will compute the predictions of "KDDTest+.arff" dataset (-i argument) using the same data descriptor generated for the training dataset (-ds) and the decision forest built previously (-m). Optionally (if the test dataset contains the labels of the tuples) run the analyzer to compute the confusion matrix (-a), and you can also store the predictions in a text file or a directory of text files(-o). Passing the (-mr) parameter will use Hadoop to distribute the classification.

- The example should output the classification time and the confusion matrix

```
10/03/13 18:08:56 INFO mapreduce.TestForest: Classification Time: 0h 0m 6s 355
10/03/13 18:08:56 INFO mapreduce.TestForest:
=======================================================
Summary
-------------------------------------------------------
Correctly Classified Instances          :      17657    78.3224%
Incorrectly Classified Instances        :       4887    21.6776%
Total Classified Instances              :      22544

=======================================================
Confusion Matrix
-------------------------------------------------------
a     b     <--Classified as
9459  252    |  9711   a     = normal
4635  8198   |  12833  b     = anomaly
Default Category: unknown: 2
```

If the input is a single file then the output will be a single text file, in the above example 'predictions' would be one single file. If the input if a directory containing for example two files 'a.data' and 'b.data', then the output will be a directory 'predictions' containing two files 'a.data.out' and 'b.data.out'

Known Issues and limitations

The "Decision Forest" code is still "a work in progress", many features are still missing. Here is a list of some known issues:

- For now, the training does not support multiple input files. The input dataset must be one single file. Classifying new data does support multiple input files.
- The tree building is done when each mapper.close() method is called. Because the mappers don't refresh their state, the job can fail when the dataset is big and you try to build a large number of trees.