

Ciencia de Datos: Un Enfoque Práctico en la Era del Big Data

Hadoop: Caso Práctico 2

Sara Del Río García

Departamento de Ciencias de la Computación e Inteligencia Artificial,
E.T.S. de Ingenierías Informática y de Telecomunicación,
srio@decsai.ugr.es



Contenido

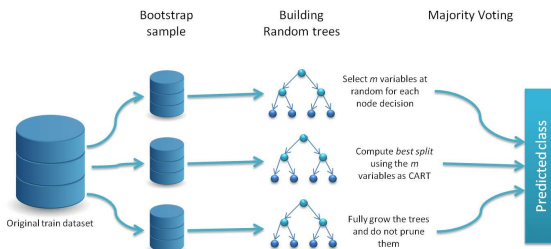
- 1 Ejemplo: Random Forest
- 2 Random Forest usando MapReduce
- 3 Referencias

Contenido

- 1 Ejemplo: Random Forest
- 2 Random Forest usando MapReduce
- 3 Referencias

Ejemplo: Random Forest

- Random Forest (RF) es un ensemble de árboles de decisión.
- La clase predicha se calcula mediante la agregación de las predicciones del conjunto a través de la votación por mayoría



Contenido

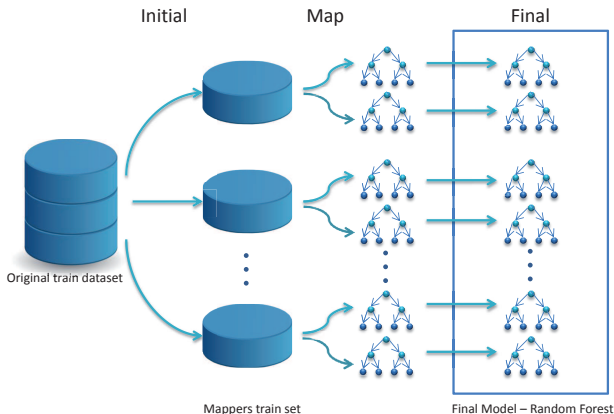
- 1 Ejemplo: Random Forest
- 2 Random Forest usando MapReduce
- 3 Referencias

Random Forest usando MapReduce

- La implementación Partial de la biblioteca Mahout (RF-BigData) es un algoritmo que construye múltiples árboles para diferentes porciones de los datos
- Este algoritmo consta de dos fases diferentes:
 - 1 Fase de Construcción del Modelo
 - 2 Fase de Clasificación
- De forma adicional, cada una de las fases constan de tres etapas:
 - 1 Inicial
 - 2 Map
 - 3 Final

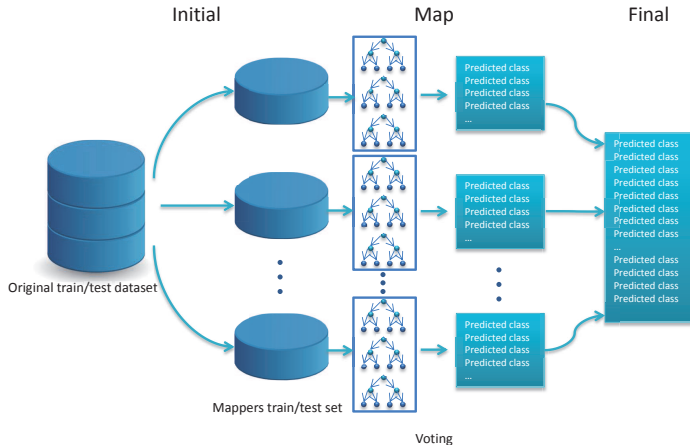
Random Forest usando MapReduce

- Fase de Construcción del Modelo



Random Forest usando MapReduce

- Fase de Clasificación



Random Forest usando MapReduce

Fase de Construcción del Modelo - Mapper Class

```
public class Step1Mapper extends MapredMapper<
    LongWritable, Text, TreeID, MapredOutput> {

    /** number of trees to be built by this mapper */
    private int nbTrees;

    /** will contain all instances if this mapper's split */
    private final List<Instance> instances = Lists.newArrayList();

    ...

    protected void setup(Context context) throws IOException
        , InterruptedException {
        super.setup(context);
        Configuration conf = context.getConfiguration();

        configure(Builder.getRandomSeed(conf), conf.getInt("mapred.
            task.partition", -1),
            Builder.getNumMaps(conf), Builder.getNbTrees(conf));
    }
```

Random Forest usando MapReduce

Fase de Construcción del Modelo - Mapper Class

```
protected void map(LongWritable key, Text value, Context
    context) throws IOException, InterruptedException {
    instances.add(converter.convert(value.toString()));
}

protected void cleanup(Context context) throws IOException,
    InterruptedException {
    Data data = new Data(getDataset(), instances);
    Bagging bagging = new Bagging(getTreeBuilder(), data);
    TreeID key = new TreeID();

    for (int treeId = 0; treeId < nbTrees; treeId++) {
        Node tree = bagging.build(rng);
        key.set(partition, firstTreeId + treeId);
        if (isOutput()) {
            MapredOutput emOut = new MapredOutput(tree);
            context.write(key, emOut);
        }
    }
}
```

Random Forest usando MapReduce

Fase de Clasificación - Mapper Class

```
public static class CMapper extends Mapper<LongWritable, Text,
    DoubleWritable, Text> {
    private DataConverter converter;
    private DecisionForest forest;
    private final Text lvalue = new Text();
    private Dataset dataset;
    private final DoubleWritable lkey = new DoubleWritable();

    ...

    protected void setup(Context context) throws IOException,
        InterruptedException {
        super.setup(context);
        Configuration conf = context.getConfiguration();
        Path[] files = HadoopUtil.getCachedFiles(conf);
        dataset = Dataset.load(conf, files[0]);
        converter = new DataConverter(dataset);
        forest = DecisionForest.load(conf, files[1]);

        ...
    }
}
```

Random Forest usando MapReduce

Fase de Clasificación - Mapper Class

```
protected void map(LongWritable key, Text value, Context
    context) throws IOException, InterruptedException {
    ...

    String line = value.toString();
    if (!line.isEmpty()) {
        Instance instance = converter.convert(line);
        double prediction = forest.classify(dataset, rng,
            instance);
        lkey.set(dataset.getLabel(instance));
        lvalue.set(Double.toString(prediction));
        context.write(lkey, lvalue);
    }
}
```

Random Forest usando MapReduce

- Marco Experimental:
 - Conjunto de datos **Iris**
 - Características: 4
 - # Instancias: 150
 - # Clases: 3 {Iris-setosa, Iris-versicolor, Iris-virginica}
 - Disponible en *UCI Machine Learning Repository*
 - Esquema de validación cruzada en 5 particiones (usaremos la primera partición)

Random Forest usando MapReduce

- Caso de estudio: 5 maps - 10 árboles
- Uso:
 - 1 Crear el directorio de entrada `"/user/hadoop/datasets/iris"` en HDFS:

```
hadoop fs -mkdir /user/hadoop/datasets  
/user/hadoop/datasets/iris
```

- 2 Mover los conjuntos de datos al directorio creado previamente en HDFS:

```
hadoop fs -put *.arff datasets/iris
```

Random Forest usando MapReduce

- Caso de estudio: 5 maps - 10 árboles
- Uso:
 - ③ Generar el fichero que describe al conjunto de datos:

```
hadoop jar /home/hadoop/mahout-distribution-0.9.jar  
org.apache.mahout.classifier.df.tools.Describe  
-p datasets/iris/iris-5-1tra.arff  
-f datasets/iris/iris-5-1tra.info  
-d 4 N L
```

Random Forest usando MapReduce

- Caso de estudio: 5 maps - 10 árboles
- Uso:
 - ③ Generar el fichero que describe al conjunto de datos:

```
[srio@hadoop-master ~]$ hadoop jar /home/srio/mahout-distribution-0.9.jar  
p/kddcup_10_normal_versus_DOS-5-1tra.info -d N 3 C 37 N L  
15/04/01 13:06:22 INFO tools.Describe: Generating the descriptor...  
15/04/01 13:06:23 INFO tools.Describe: generating the dataset...  
15/04/01 13:06:26 INFO tools.Describe: storing the dataset description  
[srio@hadoop-master ~]$
```


Random Forest usando MapReduce

- Caso de estudio: 5 maps - 10 árboles
- Uso:
 - ④ Ejecutar la aplicación:

```
hadoop jar /home/hadoop/mahout-distribution-0.9.jar  
org.apache.mahout.classifier.df.mapreduce.BuildForest  
-Dmapreduce.input.fileinputformat.split.minsize=728  
-Dmapreduce.input.fileinputformat.split.maxsize=728  
-o output_iris_5maps  
-d datasets/iris/iris-5-1tra.arff  
-ds datasets/iris/iris-5-1tra.info  
-sl 3 -p -t 10
```

Random Forest usando MapReduce

- Caso de estudio: 5 maps - 10 árboles
- Uso:
 - 4 Ejecutar la aplicación:

```

15/04/01 13:15:32 INFO mapreduce.Job: Counters: 30
  File System Counters
    FILE: Number of bytes read=13415
    FILE: Number of bytes written=135495
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=59448653
    HDFS: Number of bytes written=262746
    HDFS: Number of read operations=25
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=10
  Job Counters
    Launched map tasks=5
    Rack-local map tasks=5
    Total time spent by all maps in occupied slots (ms)=394639
    Total time spent by all reduces in occupied slots (ms)=0
    Total time spent by all map tasks (ms)=56377
    Total vcore-seconds taken by all map tasks=56377
    Total megabyte-seconds taken by all map tasks=394639000
  Map-Reduce Framework
    Map input records=388495
    Map output records=100
    Input split bytes=755
    Spilled Records=0
    Failed Shuffles=0
    Merged Map outputs=0
    GC time elapsed (ms)=386
    CPU time spent (ms)=50090
    Physical memory (bytes) snapshot=3682967552
    Virtual memory (bytes) snapshot=3593168960
    Total committed heap usage (bytes)=6913261568
  File Input Format Counters
    Bytes Read=59447898
  File Output Format Counters
    Bytes Written=262746
15/04/01 13:15:32 INFO common.HadoopUtil: Deleting hdfs://hadoop-master/user/srio/output
15/04/01 13:15:32 INFO mapreduce.BuildForest: Build Time: 0h 0m 23s 868
15/04/01 13:15:32 INFO mapreduce.BuildForest: Forest num Nodes: 1141
15/04/01 13:15:32 INFO mapreduce.BuildForest: Forest mean num Nodes: 141
15/04/01 13:15:32 INFO mapreduce.BuildForest: Forest mean max Depth: 8
15/04/01 13:15:32 INFO mapreduce.BuildForest: Storing the forest in: output/forest.seq

```

Random Forest usando MapReduce

- Caso de estudio: 5 maps - 10 árboles
- Uso:
 - 5 Usar el modelo generado en el paso anterior para clasificar nuevos datos:

```
hadoop jar /home/hadoop/mahout-distribution-0.9.jar  
org.apache.mahout.classifier.df.mapreduce.TestForest  
-i datasets/iris/iris-5-1tst.arff  
-ds datasets/iris/iris-5-1tra.info  
-m output_iris_5maps  
-a -mr  
-o predictions_iris_5maps
```

Random Forest usando MapReduce

- Caso de estudio: 5 maps - 10 árboles
- Uso:
 - ⑤ Usar el modelo generado en el paso anterior para clasificar nuevos datos:

```
15/04/01 13:21:17 INFO common.HadoopUtil: Deleting predictions/mappers
15/04/01 13:21:17 INFO mapreduce.TestForest:
=====
Summary
-----
Correctly Classified Instances      :      97117      99.9969%
Incorrectly Classified Instances    :           3      0.0031%
Total Classified Instances          :      97120
=====
Confusion Matrix
-----
a      b      <--Classified as
77732   1      |   77733      a      = negative
2      19385   |   19387      b      = positive
=====
Statistics
-----
Kappa                                -0.5406
Accuracy                             99.9969%
Reliability                           66.6628%
Reliability (standard deviation)      0.5773
```

Random Forest usando MapReduce

- Caso de estudio: 5 maps - 10 árboles
- Uso:
 - ⑥ Comprobar la salida:

```
hadoop fs -cat predictions_iris_5maps/iris-5-1tst.arff.out |  
head
```

Comprobar el estado de las ejecuciones a través de la siguiente consola web:

<http://hadoop.ugr.es:8088/cluster>

Repetir los pasos 4 - 6 con los dos últimos casos de estudio

Contenido

- 1 Ejemplo: Random Forest
- 2 Random Forest usando MapReduce
- 3 Referencias

Referencias

- Apache Mahout:

<http://mahout.apache.org/>

- Random Forest MapReduce implementation in Mahout:

<http://mahout.apache.org/users/classification/partial-implementation.html>

- UCI Machine Learning Repository - Iris dataset:

<https://archive.ics.uci.edu/ml/datasets/Iris>

Happy Hadooping!



Ciencia de Datos: Un Enfoque Práctico en la Era del Big Data

Hadoop: Caso Práctico 2

Sara Del Río García

Departamento de Ciencias de la Computación e Inteligencia Artificial,
E.T.S. de Ingenierías Informática y de Telecomunicación,
srio@decsai.ugr.es

