UNIVERSIDAD
DE GRANADA

DECSAI

Cloud Computing: Servicios y Aplicaciones

Curso 2016 - 2017

Guión de prácticas 4

Ciencia de Datos con Hadoop

El **objetivo** de esta práctica es conocer las alternativas para realizar experimentaciones de Ciencia de Datos. Para ello, haremos uso del entorno que se ha convertido en un estándar de facto como es **Hadoop**, utilizando *HDFS* como sistema de archivos distribuido y *Hadoop-MapReduce* como mecanismo de ejecución. Por último, aplicaremos la biblioteca Mahout para lanzar algoritmos de clasificación sobre conjuntos tipo Big Data.

Para constatar el manejo de la herramienta anterior, el alumno deberá realizar las **tareas** que se describen a continuación y entregar documentación describiendo las tareas realizadas.

Tareas: Realizar los objetivos que aparecen a lo largo del documento de prácticas

Documentación: Es necesario entregar un informe con la siguiente estructura:

1. Portada con:
    a. Nombre de la asignatura
    b. Nombre de la práctica
    c. Nombre del alumno
    d. Dirección de correo electrónico
2. Salida resumida para los diferentes cálculos realizados.

El documento se entregará en formato PDF. El código fuente Java asociado con las diferentes tareas MapReduce se adjuntará también en una carpeta a tal efecto.

Tanto el código fuente como el documento, se empaquetará en un único fichero .zip con el nombre "`practicaBigData`" y se entregará a través de la plataforma docente de decsai.

Fecha límite para la entrega:

12 de Junio de 2017 a las 23:59h.

Utilizando la herramienta Apache Mahout[1],

INDICE.

[1] http://mahout.apache.org/

Introducción

Vamos a familiarizarnos con los procedimientos de Ciencia de Datos, en concreto con las tareas de clasificación automática con algoritmos de Machine Learning. Para ello, haremos uso de una de las bibliotecas más utilizadas en este contexto de trabajo: Apache Mahout.

Como sabemos, la tarea de clasificación requiere dos etapas: (1) aprendizaje del modelo; (2) etiquetado de nuevos ejemplos mediante el sistema generado.

Existen multitud de algoritmos para realizar la clasificación. En nuestro caso, vamos a utilizar uno de los más robustos, el método RandomForest. Consiste en realizar un "ensemble" (agrupación) de árboles de decisión sobre distintos conjuntos de atributos, buscando los mejores puntos de corte con respecto a dichos atributos. A la hora de tomar la decisión de etiquetar un ejemplo a una clase, todos los árboles son preguntados y la salida será un consenso entre los mismos.

Para realizar nuestras pruebas, nos basaremos en el problema "Heart" del repositorio UCI, pero donde muchos ejemplos se han replicado de manera aleatoria con un modelo gaussiano. El conjunto de datos se encuentra en el directorio `hdfs://user/ahilario/datasets/BNG_heart/`

Adicionalmente, se ha puesto disponible un fichero JAR simplificado de mahout, utilizado en la asignatura SIGE. Se puede encontrar en el directorio `/tmp/mahout-distribution-sige.jar`

Objetivos

Las tareas a realizar serán las siguientes:

1. Ejecutar el algoritmo "Random Forest" sobre el conjunto de datos BNG_heart y comprueba el rendimiento alcanzado de acuerdo a los siguientes casos:
    a. Número de Maps:
        i. 64, 128, 256
    b. Número de árboles:
        i. 10, 100, 1000 árboles
2. Del punto anterior, obtener una tabla que indique los siguientes datos:
    a. Características del modelo: número de nodos (total y promedio), profundidad máxima del árbol.
    b. Tiempo de ejecución para entrenamiento.
    c. Medidas de calidad Accuracy estándar y media geométrica tanto para la partición de entrenamiento como para test.

Ejecución con Mahout

Para ejecutar Mahout, serán necesarios los siguientes pasos:

1. Si procede, importar los conjuntos de datos en un directorio HDFS con la orden "*put*"
2. Indicar el tamaño de los datos para cada partición. Para ello seguiremos la siguiente secuencia de etapas:
    a. Obtener el tamaño del fichero de entrenamiento:

    ```
    FILE_SIZE=( `hadoop fs -ls <NOMBRE.TRAIN> | awk '{print $5}'`)
    ```

    b. Calcular el tamaño de cada partición en función del número de Maps elegido:

    ```
    MAPS = 64
    BYTES_BY_PARTITION=$((FILE_SIZE/$MAPS))
    ```

    c. Crear una nueva variable para el máximo valor del Split (minimo + 1):

    ```
    MAX_BYTES_BY_PARTITION=$((BYTES_BY_PARTITION+1))
    ```

    d. Generar los descriptores del conjunto de datos necesarios para la ejecución de los métodos. Para ello, necesitamos conocer la cabecera de los mismos, es decir, cuántos atributos hay y de qué tipo es cada uno.

    En particular, para BNG_heart tenemos dicha información en el fichero BNG_heart.header: tenemos 1 numérico, 3 categóricos, seguido de 1 numérico, 2 categóricos, 1 numérico, 1 categórico, 1 numérico, 3 categóricos, y finalmente etiqueta de clase

    Para crear el descriptor se utiliza la siguiente llamada:

    hadoop jar /tmp/mahout-distribution-sige.jar org.apache.mahout.classifier.df.tools.Describe -p <RUTA_DATOS> -f <RUTA-DESCRIPTOR-SALIDA.info> -d <TIPO DE ATRIBUTOS>

    por ejemplo:

    ```
    #Generate a file descriptor for the dataset:
    DATAPATH="/user/ahilario/datasets"
    DATASET="BNG_heart"

    hadoop jar /tmp/mahout-distribution-sige.jar \
         org.apache.mahout.classifier.df.tools.Describe \
         -p $DATAPATH/$DATASET/$DATASET-5-1tra.dat \
         -f $DATASET.info -d N C 3 N 2 C N C N 3 C L;
    ```

3. Ejecutar el clasificador con los parámetros asociados:

```
hadoop jar /tmp/mahout-distribution-sige.jar \
      org.apache.mahout.classifier.df.mapreduce.BuildForest\
      -Dmapreduce.input.fileinputformat.split.minsize=
$BYTES_BY_PARTITION \
      -Dmapreduce.input.fileinputformat.split.maxsize=
$MAX_BYTES_BY_PARTITION \
      -o output_RF_100 \
      -d $DATAPATH/$DATASET/$DATASET-5-1tra.dat \
      -ds $DATASET.info \
      -sl 13 -p -t 100;
```

4. Obtener la salida para el conjunto de test:

```
hadoop jar /tmp/mahout-distribution-sige.jar \
      org.apache.mahout.classifier.df.mapreduce.TestForest \
      -i $DATAPATH/$DATASET/$DATASET-5-1tst.dat \
      -ds $DATASET.info \
      -m output_RF_100 \
      -a -mr -o output_RF_predict_out_100;

#=======================================================
#Summary
#-------------------------------------------------------
#Correctly Classified Instances         :     178762     89.381%
#Incorrectly Classified Instances       :      21238     10.619%
#Total Classified Instances             :     200000

#=======================================================
#Confusion Matrix
#-------------------------------------------------------
#a          b     <--Classified as
#77407      11361  |  88768   a     = 1
#9877       101355 |  111232  b     = 0
```