Escalabilidad en grandes conjuntos de datos

Ignacio Cordón Castillo

Características del ordenador

- SO: Linux Ubuntu 15.04, 64 bits con núcleo `4.4.0-040400-generic`
- Procesador: `Intel Core i7-4700HQ CPU, 2.40GHz × 8`
- RAM: `11.6 GiB`
- Versión de Java 64 bits: `openjdk version "1.8.0_45-internal"`
- Versión de `Weka`: 3.6.11
- Versión de `R`: `3.2.3, Wooden Christmas-Tree`
- Versión de `RWeka`: 0.4.27

Datasets

Para el desarrollo de la práctica se han empleado 4 datasets:

Covertype

581012 instancias, 12 características y 7 categorías.

Kddcup99

4898431 instancias, 41 características y 23 categorías.

Protein

1000000 instancias, 20 características y 2 categorías.

Pokerhand

1025010 instancias, 10 características y 10 categorías.

Estudio de escalabilidad

Consistía en dividir cada conjunto de datos en 20% de test y otro 80% de training. Se estudia la escalabilidad entrenando clasificadores `J48` y `Random Forest` con 50 árboles, sobre particiones del train del 20%, 40%, 60%, 80% y 100% para evaluar los resultados obtenidos sobre test, y efectuar una comparación en cuanto precisión, ejecución y tamaño del train.

Se ha usado como semilla aleatoria `12345678`

Se ha programado una función de `R`, disponible en `./bin/partitioning.R` que efectúa la división de un dataset parado como parámetro `data` al 20% test y 80% training, dividiendo a su vez training en 5 particiones disjuntas y estratificadas (test también se ha extraído con muestreo estratificado, conservando la distribución

de clases original). Para ello se han empleado las funciones `createDataPartition` y `createFolds` del paquete `caret` de `R`.

```
make.partition <- function(data, name){
  train.index <- createDataPartition(data$class, p = 0.8, list = F, times = 1)
  train <- data[ train.index, ]
  test  <- data[-train.index, ]

  folds <- createFolds(train$class, 5)

  # Returns map of folds to the original data
  partition <- list(
    train = lapply(folds, function(selected){
      train[selected, ]
    }),
    test = test)

  save (list = c('partition'), file = paste(name,".RData", sep = ""))
}
```

Se han leído los datasets y se ha aplicado la función anterior.

```
covertype <- read.arff("../data/covertype.arff")
kddcup <- read.arff("../data/kddcup99.arff")
protein <- read.arff("../data/protein.arff")
pokerhand <- read.arff("../data/pokerhand.arff")


datasets <- c(covertype, kddcup, protein, pokerhand)
datasets.names <- c("covertype", "kddcup", "protein", "pokerhand")


partitions <- lapply(1:length(datasets), function(i) {
  make.partition(datasets[i], datasets.names[i])
})
```

Una vez obtenidas las particiones, se han fusionado las dos primeras para obtener una con el 40% de train, las tres primeras para obtener otra con el 60% de train y se han escrito cada una de las particiones para cada dataset con la función `write.arff` del paquete `RWeka` en un dataset de nombre `./data/train{porcentaje}-{nombre-dataset}` o `./data/test-{nombre-dataset}` (p.e. `train20-covertype`, `test-covertype`).

A su vez, se han guardado las particiones correspondientes a un dataset en un archivo de la forma `{nombre-dataset}.RData` para liberar toda la memoria RAM posible y disponer de la mayor cantidad posible para la ejecución de algoritmos.

Se ha automatizado la ejecución de `Weka` sobre cada una de las particiones, con un script `bash` para obtener los resultados en ficheros homónimos en la carpeta `results`

```
#!/bin/bash


training=(train20 train40 train60 train80 train100)
datasets=(covertype kddcup protein pokerhand)
```

```
for train in ${training[*]}
do
    for d in ${datasets[*]}
    do
        echo "Haciendo J48 sobre ${train}-${d}"
        java -cp ~/weka-3-8-1/weka.jar -Xmx8g weka.classifiers.trees.J48 \
             -t ../data/${train}-${d}.arff -T ../data/test-${d}.arff \
             > ../results/J48-${train}-${d}
        echo "Haciendo Random Forest sobre ${train}-${d}"
        java -cp ~/weka-3-8-1/weka.jar -Xmx8g weka.classifiers.trees.RandomForest -I 50 \
             -t ../data/${train}-${d}.arff -T ../data/test-${d}.arff \
             > ../results/RF-${train}-${d}

    done
done
```