Minería de Flujo de Datos: Trabajo Final

Juanjo Sierra

25 de abril de 2019

Ejercicio 1: Entrenamiento offline (estacionario) y evaluación posterior.

Entrenar un clasificador HoeffdingTree offline (estacionario, aprender modelo únicamente), sobre un total de 1.000.000 de instancias procedentes de un flujo obtenido por el generador WaveFormGenerator con semilla aleatoria igual a 2. Evaluar posteriormente (sólo evaluación) con 1.000.000 de instancias generadas por el mismo tipo de generador, con semilla aleatoria igual a 4. Repita el proceso varias veces con la misma semilla en evaluación y diferentes semillas en entrenamiento, para crear una población de resultados. Anotar como resultados los valores de porcentajes de aciertos en la clasificación y estadístico Kappa.

El modelo se entrena con el siguiente comando en la consola de MOA.

```
java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask "EvaluateModel -m \
(LearnModel -l trees.HoeffdingTree -s (generators.WaveformGenerator -i 1) -m \
1000000) -s (generators.WaveformGenerator -i 4) -i 1000000"
```

Realizamos una población de tamaño 30 utilizando 30 semillas distintas para comprobar cómo se comporta este algoritmo en general. Almacenamos los datos de accuracy y Kappa para las 30 ejecuciones en un dataframe.

```
accuracy = array(dim = 30)
kappa = array(dim = 30)
for (i in 1:30) {
  archivo =
    paste(c(paste(c("./Datos/Ejercicio1.1/hoeffdingEstacionario",i),
                  collapse = ""),".csv"),collapse="")
  con = file(archivo,open="r")
  datos = readLines(con)
  accuracyFinal =
    as.double(gsub(",", ".", gsub(".* = ", "", datos[2])))
  accuracy[i] = accuracyFinal
  kappaFinal =
    as.double(gsub(",", ".", gsub(".* = ", "", datos[3])))
  kappa[i] = kappaFinal
  close(con)
}

hoeffdingEst = as.data.frame(cbind(accuracy, kappa))
hoeffdingEst
```

```
##    accuracy  kappa
## 1    84.509 76.765
## 2    84.512 76.770
## 3    84.590 76.887
## 4    84.666 77.001
## 5    84.481 76.723
## 6    84.342 76.514
## 7    84.799 77.200
```

```
## 8    84.153 76.231
## 9    84.641 76.963
## 10   84.578 76.869
## 11   84.539 76.810
## 12   84.457 76.688
## 13   84.369 76.555
## 14   84.547 76.822
## 15   84.648 76.974
## 16   84.626 76.940
## 17   84.513 76.772
## 18   84.434 76.653
## 19   84.605 76.910
## 20   84.568 76.853
## 21   84.518 76.779
## 22   84.657 76.988
## 23   84.543 76.815
## 24   84.754 77.132
## 25   84.646 76.971
## 26   84.586 76.880
## 27   84.488 76.734
## 28   83.529 75.294
## 29   84.591 76.888
## 30   84.505 76.759
```

Ahí se pueden ver los valores que ha obtenido para accuracy y Kappa el modelo estacionario.

Repetir el paso anterior, sustituyendo el clasificador por HoeffdingTree adaptativo.

Para ello utilizamos el mismo comando anterior pero sustituyendo `trees.HoeffdingTree` por `trees.HoeffdingAdaptiveTree`. De nuevo generamos una población de tamaño 30 mediante un script.

```
accuracy = array(dim = 30)
kappa = array(dim = 30)
for (i in 1:30) {
  archivo =
    paste(c(paste(c("./Datos/Ejercicio1.2/hoeffdingAdaptativo",i),
                  collapse = ""),".csv"),collapse="")
  con = file(archivo,open="r")
  datos = readLines(con)
  accuracyFinal =
    as.double(gsub(",", ".", gsub(".* = ", "", datos[2])))
  accuracy[i] = accuracyFinal
  kappaFinal =
    as.double(gsub(",", ".", gsub(".* = ", "", datos[3])))
  kappa[i] = kappaFinal
  close(con)
}

hoeffdingAd = as.data.frame(cbind(accuracy, kappa))
hoeffdingAd
```

```
##    accuracy  kappa
## 1    84.521 76.783
```

```
## 2    84.474 76.712
## 3    84.416 76.625
## 4    84.465 76.699
## 5    84.262 76.395
## 6    84.368 76.554
## 7    84.271 76.408
## 8    84.243 76.367
## 9    84.478 76.719
## 10   84.326 76.491
## 11   84.371 76.558
## 12   84.416 76.627
## 13   84.498 76.749
## 14   84.415 76.624
## 15   84.229 76.345
## 16   84.328 76.494
## 17   84.358 76.539
## 18   84.456 76.685
## 19   84.451 76.679
## 20   84.459 76.690
## 21   84.553 76.831
## 22   84.432 76.650
## 23   84.686 77.030
## 24   84.485 76.729
## 25   84.589 76.886
## 26   84.372 76.559
## 27   84.476 76.716
## 28   83.978 75.968
## 29   84.379 76.570
## 30   84.222 76.335
```

Responda a la pregunta: ¿Cree que algún clasificador es significativamente mejor que el otro en este tipo de problemas? Razone su respuesta.

Para responder a esta pregunta de forma adecuada deberíamos evaluar la similitud de los resultados de ambos algoritmos en base a tests estadísticos. Para saber si utilizar tests paramétricos o no paramétricos comprobamos con el test de Shapiro-Wilk si los valores siguen una distribución normal.

```
shapiro.test(hoeffdingEst$accuracy)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  hoeffdingEst$accuracy
## W = 0.68496, p-value = 9.496e-07
```

```
shapiro.test(hoeffdingAd$accuracy)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  hoeffdingAd$accuracy
## W = 0.94445, p-value = 0.1199
```

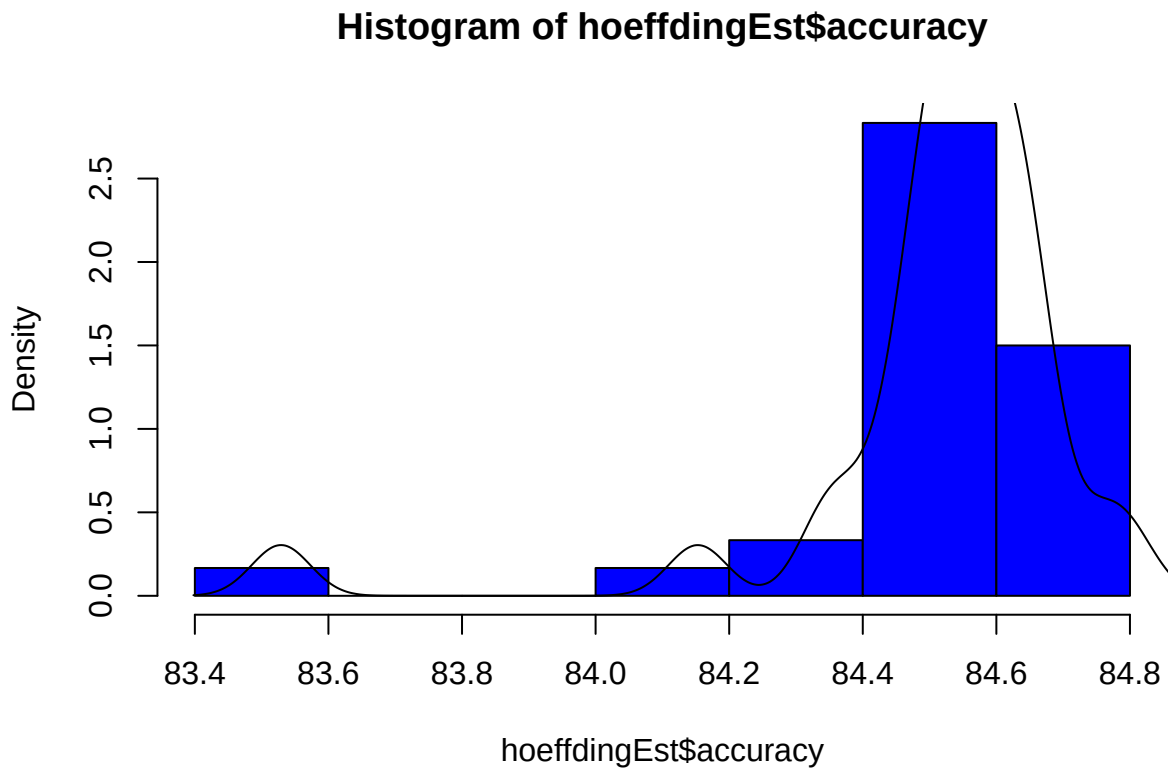
Como el accuracy del modelo estacionario no sigue una distribución normal según el test, para comparar esta variable tendré que usar un test no paramétrico como Wilcoxon.

```r
wilcox.test(hoeffdingEst$accuracy, hoeffdingAd$accuracy, exact = FALSE, alternative = )
```

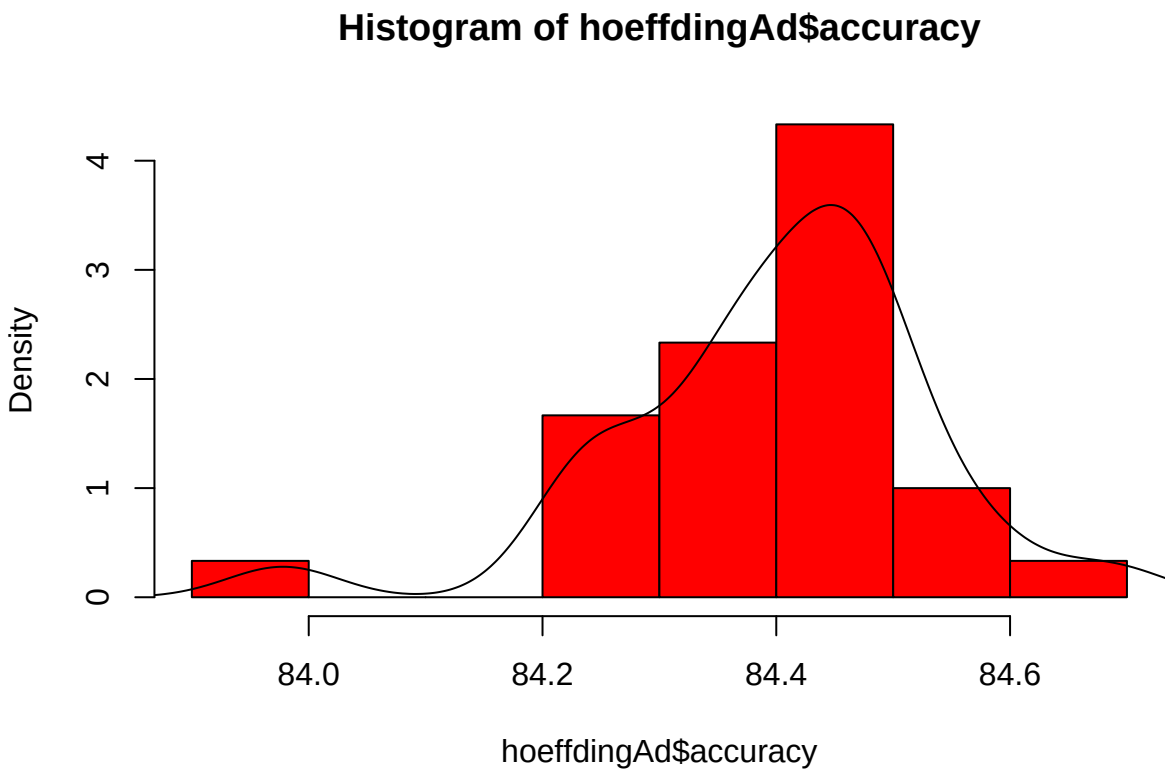
```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  hoeffdingEst$accuracy and hoeffdingAd$accuracy
## W = 720, p-value = 6.763e-05
## alternative hypothesis: true location shift is not equal to 0
```

El test dice que hay diferencias significativas entre ambas poblaciones. Observemos los histogramas de los valores para entender qué está ocurriendo.

```r
hist(hoeffdingEst$accuracy, col="blue", prob=TRUE)
lines(density(hoeffdingEst$accuracy))
```




```r
hist(hoeffdingAd$accuracy, col="red", prob=TRUE)
lines(density(hoeffdingAd$accuracy))
```



Como se puede observar, el Hoeffding adaptativo (en color rojo) sigue una distribución más normal que el estacionario (en color azul). El estacionario no ha sido clasificado como distribución normal debido a la oblicuidad hacia la derecha que sufre. Además, comprobamos que la gran mayoría de los resultados que obtiene el estacionario están a la derecha de 84.4, por lo que se entiende que salga elegido por Wilcoxon como el mejor algoritmo.

En un flujo que no tiene cambios de concepto es asumible pensar que el algoritmo estacionario se mantenga estable y funcione mejor de forma general a lo largo del tiempo que un modelo adaptativo que puede estar aprendiendo cosas erróneas con los distintos momentos del flujo.