UNIVERSIDAD DE GRANADA

MÁSTER DE CIENCIA DE DATOS E INGENIERÍA DE COMPUTADORES

CURSO ACADÉMICO 2019-2020

SERIES TEMPORALES Y MINERÍA DE FLUJO DE DATOS

Ejercicio guiado: Minería de Flujo de datos.

Análisis de flujos de datos con diferentes algoritmos y modelos y comparación de modelos.

Nicolás Cubero

22 de Abril de 2020

Índice

1. Comparación de *Hoeffding Tree* y *Naïve Bayes*

En este primer ejercicio se comparará el rendimiento del clasificador ***Hoeffding Tree*** con el clasificador ***Naïve Bayes***.

Esta comparación se basarán en las métricas de rendimiento medidas para ambos algoritmos sobre 30 ejecuciones evaluadas haciendo mediante el evaluador ***Interleaved Test-Then-Train*** de **MOA**.

En cada ejecución, se someterá cada clasificador a un flujo de datos de 1.000.000 de instancias generadas mediante el generador ***RandomTreeGenerator*** también implementado en **MOA** y, tomando diferentes semillas inicializadoras.

Se lanzan las ejecuciones con **MOA** desde la línea de comandos con ayuda del siguiente *script* en *bash*:

```
#!/bin/bash

BASEDIR="/home/nico/moa-release-2019.05.0" #`dirname $0`/..
#BASEDIR=`(cd "$BASEDIR"; pwd)`
MEMORY=512m

seeds=(23 17 2246 76 9257 8 349 5 12165 51 \
    43 104 1073 7624 2 866 1355 1274 9543 4421 \
    5378 1007 1743 5888 1921 4211 3422 56 1020 370)

# Crear directorios si no existen
if [ ! -d "naive-bayes_results" ]
  then mkdir naive-bayes_results
fi

if [ ! -d "hoeffding-tree_results" ]
  then mkdir hoeffding-tree_results
fi

# Crear ficheros donde se almacenarán los resultados finales
echo -n "" > nb.txt
echo -n "" > ht.txt


for i in $(seq 0 29)
do
  # Ejecución con Naive Bayes
  java -Xmx$MEMORY -cp "$BASEDIR/lib/moa-2019.05.0:$BASEDIR/lib/*" \
  -javaagent:$BASEDIR/lib/sizeofag-1.0.4.jar moa.DoTask \
  "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes " \
  "-s (generators.RandomTreeGenerator -i ${seeds[$i]} ) " \
    " -i 1000000 -f 10000" > "naive-bayes_results/nb${i}.txt"

```

```
  # Almacenar últimos valores de columna "Correct Classifier (Percent)"
  cat "naive-bayes_results/nb${i}.txt" | tail -n 1 | \
    sed -re 's/([^,]*,){4}([^,]*),.*/\2/' >> nb.txt

  # Ejecución con Hoeffding Trees
  java -Xmx$MEMORY -cp "$BASEDIR/lib/moa-2019.05.0:$BASEDIR/lib/*" \
  -javaagent:$BASEDIR/lib/sizeofag-1.0.4.jar moa.DoTask \
  "EvaluateInterleavedTestThenTrain -l trees.HoeffdingTree " \
  "-s (generators.RandomTreeGenerator -i ${seeds[$i]} ) " \
    " -i 1000000 -f 10000" > "hoeffding-tree_results/ht${i}.txt"

  # Almacenar últimos valores de columna "Correct Classifier (Percent)"
  cat "hoeffding-tree_results/ht${i}.txt" | tail -n 1 | \
    sed -re 's/([^,]*,){4}([^,]*),.*/\2/' >> ht.txt
done
```

Script 1: Script bash para la ejecución de las 30 experimentaciones para Naive Bayes y Hoeffding tree

En cada ejecución y para cada modelo el evaluador anota entre otras medidas: las instancias del flujo sobre las que se evalúa el modelo, el tiempo de evaluación (segundos de CPU), el coste del modelo en memoria RAM (horas de uso de RAM), el número de instancias evaluadas, el porcentaje de clasificaciones correctas, etc.

El anterior *script* extrae de las anteriores medidas el **porcentaje de clasificaciones correctas** al final de cada flujo en cada una de las 30 ejecuciones, tanto para el clasificador *Hoeffding Tree* como el **Naïve Bayes**, y almacena esta anotación en los ficheros `ht.txt` y `nb.txt` respectivamente.

Para comparar ambos clasificadores, nos basamos en estos **porcentajes de clasificaciones correctas**, mediante la aplicación de un test de hipótesis de muestras pareadas para determinar si existen diferencias significativas entre los porcentajes evaluados para cada clasificador.

Primeramente, necesitamos conocer si los porcentajes evaluados para cada clasificador se ajustan normalmente, por lo que aplicamos el **test de *Shapiro-Wilks***:

```
# Test de shapiro-wilk sobre "nb"
shapiro.test(nb)

# Test de shapiro-wilk sobre "ht"
shapiro.test(ht)
```

Script 2: Sentencias para la aplicación del test de Shapiro Wilk sobre las mediciones de ambos clasificadores

```
Shapiro-Wilk normality test
```

```
data:  nb
W = 0.98247, p-value = 0.8867

Shapiro-Wilk normality test

data:  ht
W = 0.9598, p-value = 0.3061
```

Asumiendo una confianza del 95 %, los *p-value* de los tests nos llevan, para ambas mediciones, a aceptar la hipóteis nula del test y afirmar que **las mediciones de ambos clasificadores se distribuyen normalmente**.

Por consiguiente, resulta más idoneo aplicar un test T para muestras pareadas:

```r
# Prueba T para muestras pareadas
t.test(nb, ht, paired = TRUE)
```

Script 3: Sentencias para la aplicación del test T para muestras pareadas

```
Paired t-test

data:  nb and ht
t = -834.23, df = 29, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -20.93606 -20.83365
sample estimates:
mean of the differences
               -20.88485
```

Con una confianza del 95 %, el *p-value* del test nos lleva a rechazar la hipótesis nula del test y a considerar que **existen diferencias significativas en los rendimientos de ambos clasificadores**.

Por último, comparamos las medias de las medidas de ambos clasificadores para conocer cuál clasificador ofrece mejores rendimientos:

```r
# Mostrar la media de cada medida
cat('Media de las medidas del clasificador Naïve-Bayes: ', mean(nb),
    fill=T)
cat('Media de las medidas del clasificador Hoeffding Tree: ', mean(ht),
     fill=T)
```

Script 4: Sentencias para el cálculo de las medias de las medidas de ambos clasificadores

```
Media de las medidas del clasificador Naïve-Bayes:  73.6732
Media de las medidas del clasificador Hoeffding Tree:  94.55805
```

Concluímos, por tanto, que el clasificador *Hoeffding Tree* ofrece mejores rendimientos en la clasificación de los datos de los flujos de datos simulados.

2. Evaluación del clasificador Naïve-Bayes sobre un flujo con Concept Drift

A continuación, se evaluará el rendimiento de un clasificador *Naïve-Bayes* sobre un flujo de datos con *ConceptDriff* generado con el generador *SEAGenerator*.

Se modelará el cambio de concepto centrado en la instancia 20.000 en una ventana de 100 instancias simulando, primeramente, una función 2 y, posteriormente una función 3.

Por último, la evaluación del clasificador se realizará con una frecuencia de muestreo de 1000 instancias y con el evaluador prequencial.

Programamos esta evaluación desde la interfaz gráfica de *MOA*:

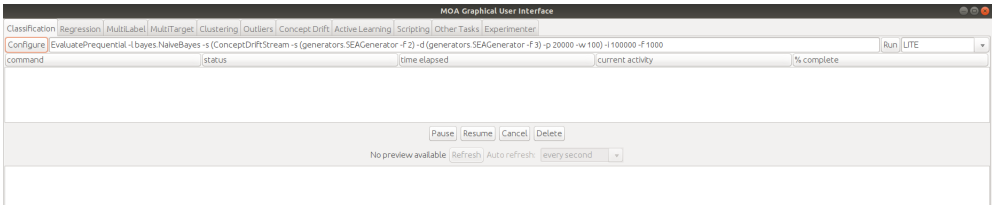

Figura 1: Captura de la interfaz gráfica de MOA donde se programa la evaluación del clasificador Naïve-Bayes ante el flujo con cambio de concepto.

Al ejecutar y costruir el modelo programado, la evolución de las métricas *accuracy, Kappa* y *Kappa Temp* a lo largo del flujo de datos muestran los resultados de las siguientes gráficas:

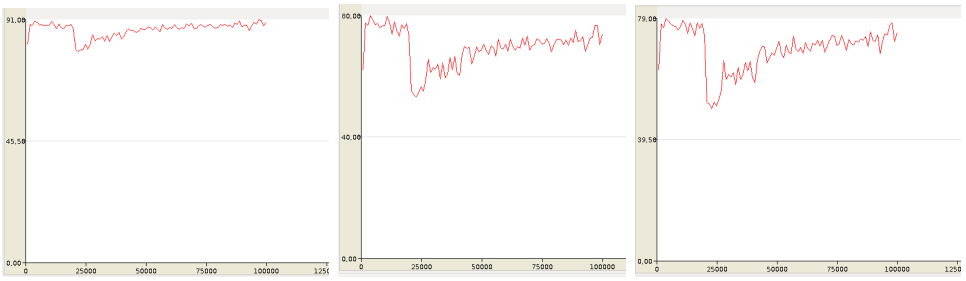

Figura 2: Captura de la evolución de las métricas accuracy, kappa y kappa temp, respectivamente, a lo largo del flujo de datos.

En las anteriores gráficas, las métricas evaluadas reflejan un descenso brusco en el rendimiento del clasificador desde su rendimiento máximo en torno a la instancia 25000. A partir de esta instancia, comienza un ascenso creciente y lento hasta alcanzar nuevamente, el rendimiento máximo.

La razón a este fenómeno se explica por el cambio de concepto centrado en la instancia 20.000 especificado al generador y, por el cual, se aplica un cambio en la función usada por el generador *SEA*, lo que provocaría que, el clasificador prediga erróneamente los datos inmediatamente posteriores al cambio de concepto al haber sido desarrollado, hasta el momento con datos de la misma función 2.

Conforme llegan nuevas instancias tras el cambio de concepto, el clasificador comienza a ajustarse a los nuevos datos generados con la nueva función. El rendimiento del modelo crece de forma progresiva, más lentamente que en las primeras instancias del flujo de datos hasta alcanzar su rendimiento máximo.

3. Evaluación del clasificador Naïve-Bayes estático sobre un flujo con Concept Drift

Como tercera ejecución, entrenamos un clasificador *Naïve-Bayes* con un flujo del generador *SEAGenerator* con función 2 y con una frecuencia de muestreo de 100.000 y evaluamos su rendimiento ante el flujo de datos generado en la sección anterior.

Se programa esta ejecución en la interfaz gráfica de *MOA* y se procede a ejecutar:

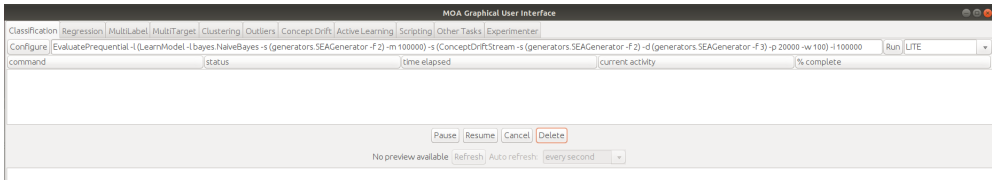

Figura 3: Captura de la interfaz gráfica de MOA donde se programa la evaluación del clasificador Naïve-Bayes estático.

El clasificador obtiene las siguientes evaluaciones de las métricas *accuracy, Kappa* y *Kappa Temp* para el flujo de datos con *ConceptDriff*:

Métrica	Naïve-Bayes estático	Naïve-Bayes dinámico
Accuracy	83.3	89.2
Kappa Statistic	61.230232061437306	72.84480807816674
Kappa Temporal Statistic	60.14319809069212	74.22434367541767
Kappa M Statistic	44.518272425249165	64.11960132890367

Cuadro 1: Métricas de accuracy, estadístico Kappa, estadístico Kappa temporal y estadístico Kappa M sobre el modelo Naïve-Bayes estático ante el flujo de datos con ConceptDriff y su comparación con las métricas evaluadas para el anterior modelo Naïve-Bayes

En la tabla, se refleja claramete que el rendimiento del modelo *Naïve-Bayes* estático es inferior (en todas las métricas) al modelo *Naïve-Bayes* dinámico: Al constituir un modelo estático ,no se re-entrena al cambiar la función del generador *SEAGenerator* en el cambio de concepto y, por lo tanto, no puede ajustarse a este nuevo conjunto de datos.

4. ¿Qué ocurriría si pudiésemos detectar un cambio de concepto y re-entrenar un modelo estacionario?

En este caso, no se estaría entrenando un modelo propiamente estacionario, sino que se desarrollaría un modelo diferente tras cada cambio de concepto

Para verificar el rendimiento de este tipo de modelo, se propone desarrollar otro clasificador *Naïve-Bayes* que se reentrene tras cada cambio de concepto, para el cual se aplicará entrenador *SingleClassifierDrift* con el método *DDM*.

Configuramos la ejecución en *MOA*, en esta ocasión, haciendo uso nuevamente de la línea de comandos y proporcionando el siguiente conjunto de sentencias:

```
EvaluateInterleavedTestThenTrain -l
(moa.classifiers.drift.SingleClassifierDrift -l bayes.NaiveBayes -d DDM)
-s (ConceptDriftStream -s (generators.SEAGenerator -f 2) -d
 (generators.SEAGenerator -f 3) -p 20000 -w 100) -i 100000
```

El modelo conjunto obtiene las siguientes métricas sobre el flujo de datos:

Métrica	Naïve-Bayes reajuste	Naïve-Bayes estático	Naïve-Bayes dinámico
Accuracy	88.086	83.3	89.2
Kappa Statistic	71.236	61.2302	72.8448
Kappa Temporal Statistic	72.4716	60.1432	74.2243
Kappa M Statistic	63.086	44.5183	64.1196

Cuadro 2: Métricas de accuracy, estadístico Kappa, estadístico Kappa temporal y estadístico Kappa M sobre el modelo Naïve-Bayes que reentrena tras cada cambio de concepto y su comparación con las métricas evaluadas para el modelo Naïve-Bayes original dinámico

Observamos que el rendimiento de este modelo conjunto es ligeramente inferior al modelo Naïve-Bayes original pero supera significativamente el rendimiento del modelo estático.