Series Temporales y Minería de flujos de datos

Trabajo Guiado: MOA (*Massive Online Analysis*)

Parte II: Minería de flujos de datos

Manuel Pegalajar Cuéllar (manupc@decsai.ugr.es)
Dpto. Ciencias de la Computación e I.A., Universidad de Granada

Índice

- MOA por línea de comandos.
- Clasificación.
- Concept Drift.
- Entrega del ejercicio guiado.

Índice

- **MOA por línea de comandos.**
- Clasificación.
- Concept Drift.
- Entrega del ejercicio guiado.

MOA por línea de comandos

- En el seminario sobre MOA se han explicado varios problemas que se pueden abordar con el software MOA.
- Se recomienda hacer uso de la línea de comandos para abarcar todas las posibilidades de MOA, pudiendo ser capaces de replicar la experimentación tantas veces como sea necesario.
- Para hacer uso de la línea de comandos, hay que ejecutar MOA de la siguiente forma:

java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask "*Tarea*"

Donde "**tarea**" **(comillas incluidas)** es la tarea a realizar.

MOA por línea de comandos

- Se proporciona en PRADO el manual de MOA, como copia local descargable desde: https://moa.cms.waikato.ac.nz/
- El manual es una herramienta vital para conocer todas las tareas de MOA y sus parámetros (ejemplos incluidos).
- Ejemplo: *EvaluatePrequential*

Parameters:

- Same parameters as EvaluateInterleavedTestThenTrain
- -e : Classification performance evaluation method
  - WindowClassificationPerformanceEvaluator
    - -w : Size of sliding window to use with WindowClassification-PerformanceEvaluator
  - FadingFactorClassificationPerformanceEvaluator
    - -a : Fading factor to use with FadingFactorClassificationPerfor-manceEvaluator
  - EWMAFactorClassificationPerformanceEvaluator
    - -a : Fading factor to use with FadingFactorClassificationPerfor-manceEvaluator

5.7 EvaluatePrequential

Evaluates a classifier on a stream by testing then training with each example in sequence. It may use a sliding window or a fading factor forgetting mecha-nism.

This evaluation method using sliding windows and a fading factor was presented in

[C] João Gama, Raquel Sebastião and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *KDD'09*, pages 329–338.

The fading factor α is used as following:

$$E_i = \frac{S_i}{B_i}$$

with

$$S_i = L_i + \alpha \times S_{i-1}$$

$$B_i = n_i + \alpha \times B_{i-1}$$

where n_i is the number of examples used to compute the loss function L_i. $n_i = 1$ since the loss L_i is computed for every single example.

Examples:

```
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask \
  "EvaluatePrequential -l trees.HoeffdingTree \
  -e (WindowClassificationPerformanceEvaluator -w 10000) \
  -s generators.WaveformGenerator \
  -i 100000000 -f 1000000" > htresult.csv
```

MOA por línea de comandos

- Ejemplo: ¿Qué hace la siguiente tarea?

```
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask \
  "EvaluateInterleavedTestThenTrain -l trees.HoeffdingTree \
  -s generators.WaveformGenerator \
  -i 100000000 -f 1000000" > htresult.csv
```

MOA por línea de comandos

5.6 EvaluateInterleavedTestThenTrain

Evaluates a classifier on a stream by testing then training with each example in sequence. Example:

```
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask \
  "EvaluateInterleavedTestThenTrain -l trees.HoeffdingTree \
  -s generators.WaveformGenerator \
  -i 100000000 -f 1000000" > htresult.csv
```

Parameters:

- -l : Classifier to train
- -s : Stream to learn from
- -e : Classification performance evaluation method
- -i : Maximum number of instances to test/train on (-1 = no limit)
- -t : Maximum number of seconds to test/train for (-1 = no limit)
- -f : How many instances between samples of the learning performance
- -b : Maximum size of model (in bytes). -1 = no limit
- -q : How many instances between memory bound checks
- -d : File to append intermediate csv results to
- -O : File to save the final result of the task to

MOA por línea de comandos

- Interpretación de resultados: El fichero del ejemplo.

Indicadores y estadísticos de resultados

C	D	E	F	
model cost (RAM-Hours)	classified instances	classifications correct (percent)	Kappa Statistic (percent)	Kappa Temporal Statistic (p
5.09660082264079E-9	10000.0	80.13	70.2168965376655	70.2856288320622
7.76859924732604E-9	20000.0	79.69000000000001	69.56353753157502	69.6843047988656
9.921100750608992E-9	30000.0	79.93333333333334	69.92213044421285	69.87288559703734
1.345498114138221E-8	40000.0	80435	70.671149968273	70.62091748629776
1.807697379042291E-8	50000.0	80.44200000000001	70.67389768519415	70.62481225593272
2.474783655923688E-8	60000.0	80.62833333333333	70.95389467819486	70.79721615034799
3.126964359832928E-8	70000.0	80.72714285714285	71.09984132739656	
3.94291513209045E-8	80000.0	80.94125	71.41897599560595	
4.714997567315897E-8	90000.0	81.19444444444444	71.79944484978635	
5.5688736104269834E-8	100000.0	81378	72.0716202784353	
6.598806380541376E-8	110000.0	81.53727272727272	72.3096371199828	72.18250054788517
7.667745171122461E-8	120000.0	81.65333333333334	72.48258952591267	72.36080597577052
8.604374586112178E-8	130000.0	81.82923076923076	72.74577849911535	72.63533473118403
9.587690560288108E-8	140000.0	81.85142857142857	72.77959838943461	72.67192978682213
1.0505067195169008E-7	150000.0	81972	72.96079395755515	72.88288559309287
1.1604608298067832E-7	160000.0	82.023125	73.03762001955002	72.96151459888323
1.2573999944287664E-7	170000.0	82.09058823529412	73.13871756920211	73.07998373092363
1.3761246103368907E-7	180000.0	82.15444444444444	73.23358591025666	73.17107801785698
1.5087543344557492E-7	190000.0	82.27	73.40700480097253	73.35521632523927
1.641075277563888E-7	200000.0	82352	73.52963602665398	73.48238971030172
1.7635881223246867E-7	210000.0	82.41761904761906	73.62842700743597	73.57262999677916
1.9003288189866888E-7	220000.0	82.4909090909091	73.73805128561865	73.67629773392014
2.036843860799943E-7	230000.0	82.53086956521739	73.7978867278709	73.74745177983377
2.181720127899075E-7	240000.0	82.58375	73.87734673863424	73.8426012841212
2.3398674042392517E-7	250000.0	82.6524	73.98001105275999	73.9588086946079

Resultados obtenidos a lo largo del tiempo, a ser analizados.

Manuel Pegalajar Cuéllar (manupc@decsai.ugr.es)
Dpto. Ciencias de la Computación e I.A., Universidad de Granada

MOA por línea de comandos

- Concatenación de tareas:
  - La línea de comandos permite concatenar tareas en un pipeline, haciendo que la salida de una tarea sea entrada a otra.
- Ejemplo:

```
java -cp moa.jar -javaagent:sizeofag.jar moa.DoTask \
  "EvaluateModel -m (LearnModel -l trees.HoeffdingTree \
  -s generators.WaveformGenerator -m 1000000) \
  -s (generators.WaveformGenerator -i 2) -i 1000000"
```

- El ejemplo primero entrena un modelo (estacionario), que luego es evaluado en otro conjunto de datos generado con semilla diferente.

Índice

- MOA por línea de comandos.
- **Clasificación.**
- Concept Drift.
- Entrega del ejercicio guiado.

- **Ejercicio:**
  - Se pide comparar la eficacia de un Hoeffding Tree con un clasificador Naïve Bayes, para un flujo de datos de 1.000.000 de instancias generadas con un generador RandomTreeGenerator, suponiendo una frecuencia de muestreo de 10.000 y con el método de evaluación Interleaved Test-Then-Train.

- **Solución:**
  - Hay que mirar el manual para ver cómo evaluar con Test-Then-Train, cada uno de los clasificadores y generadores, y luego aplicarlo para Hoeffding Trees y Naïve Bayes.

5.6 EvaluateInterleavedTestThenTrain

Evaluates a classifier on a stream by testing then training with each example in sequence. Parameters:

- -l : Classifier to train
- -s : Stream to learn from
- -e : Classification performance evaluation method
- -i : Maximum number of instances to test/train on (-1 = no limit)
- -t : Maximum number of seconds to test/train for (-1 = no limit)
- -f : How many instances between samples of the learning performance
- -b : Maximum size of model (in bytes). -1 = no limit
- -q : How many instances between memory bound checks
- -d : File to append intermediate csv results to
- -O : File to save the final result of the task to

6.2.7 generators.RandomTreeGenerator

Generates a stream based on a randomly generated tree.
This generator is based on that proposed in

- -r: Seed for random generation of tree
- -i: Seed for random generation of instances
- -c: The number of classes to generate
- -o: The number of nominal attributes to generate
- -u: The number of numeric attributes to generate
- -v: The number of values to generate per nominal attribute
- -d: The maximum depth of the tree concept
- -l: The first level of the tree above maxTreeDepth that can have leaves
- -f: The fraction of leaves per level from firstLeafLevel onwards

7.1.1 NaiveBayes

Performs classic bayesian prediction while making naive assumption that all inputs are independent.

Parameters:

- -r : Seed for random behaviour of the classifier

7.2.2 HoeffdingTree

Decision tree for streaming data.

A *Hoeffding tree* is an incremental, anytime decision tree induction algorithm that is capable of learning from massive data streams, assuming that the distribution generating examples does not change over time. Hoeffding trees exploit the fact that a small sample can often be enough to choose an optimal splitting attribute. This idea is supported mathematically by the Hoeffding bound, which quantifies the number of observations (in our case, examples)

...

6.2.7 generators.RandomTreeGenerator

Generates a stream based on a randomly generated tree.
This generator is based on that proposed in

- -r: Seed for random generation of tree
- -i: Seed for random generation of instances
- -c: The number of classes to generate
- -o: The number of nominal attributes to generate
- -u: The number of numeric attributes to generate
- -v: The number of values to generate per nominal attribute
- -d: The maximum depth of the tree concept
- -l: The first level of the tree above maxTreeDepth that can have leaves
- -f: The fraction of leaves per level from firstLeafLevel onwards

- **Una ejecución para Naïve Bayes:**

```
manupc@aquiles:~/Escritorio/MOASoftware/moa-release-2017.06b/moa-release-2017.06
b$ java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask \
>  "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes \
> -s generators.RandomTreeGenerator -i 1000000 -f 10000"
```

- **Una ejecución para Hoeffding Trees:**

```
manupc@aquiles:~/Escritorio/MOASoftware/moa-release-2017.06b/moa-release-2017.06
b$ java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask \
>  "EvaluateInterleavedTestThenTrain -l trees.HoeffdingTree \
> -s generators.RandomTreeGenerator -i 1000000 -f 10000"
```

- **Ejercicio (continuación):**
  - Para saber si hay diferencias significativas, tendríamos que generar una población de resultados y escoger una medida de eficacia para comparar.
  - Para ello, escogeremos 30 semillas diferentes y ejecutaremos 30 veces el mismo método (en total 30 ejecuciones para Naïve Bayes y otras 30 para Hoeffding Trees).
  - Escogeremos los resultados del porcentaje de aciertos en la clasificación, y las compararemos con un test estadístico.

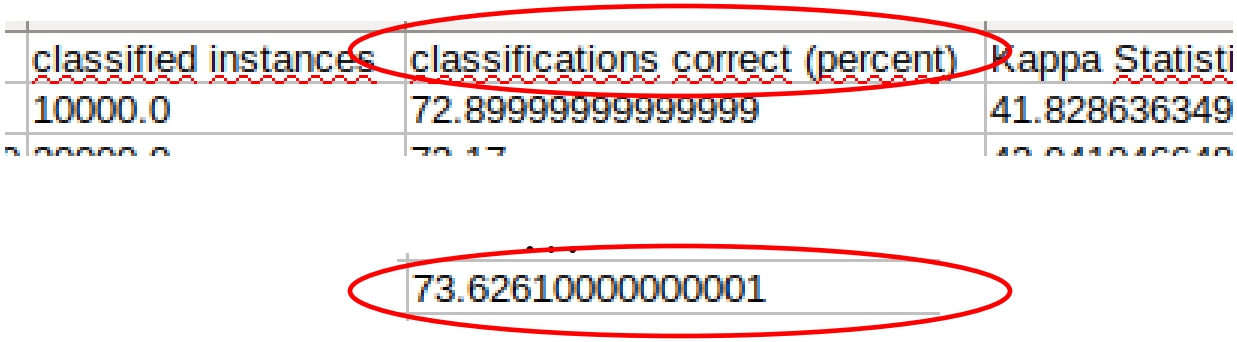
- **Script con la ejecución con varias semillas para el generador aleatorio (Naïve Bayes):**

```
java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask \
 "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes \
-s (generators.RandomTreeGenerator -i 1) -i 1000000 -f 10000" > nb1.txt

java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask \
 "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes \
-s (generators.RandomTreeGenerator -i 2) -i 1000000 -f 10000" > nb2.txt

java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask \
 "EvaluateInterleavedTestThenTrain -l bayes.NaiveBayes \
-s (generators.RandomTreeGenerator -i 3) -i 1000000 -f 10000" > nb3.txt
```

...

Clasificación

- **De cada uno de los ficheros de resultados, cogemos el último valor de la columna "Classification Correct (Percent)".**



classified instances	classifications correct (percent)	Kappa Statisti
10000.0	72.89999999999999	41.828636349

73.62610000000001

Con los 30 valores que obtenemos para cada clasificador, nos creamos 2 poblaciones de datos (una por cada clasificador) y las guardamos aparte, para compararlas con test estadísticos.

- Tests a usar: Si las dos distribuciones son normales, un test paramétrico. Si alguna no lo es, un test no paramétrico. El resultado del test nos indicará si los resultados son estadísticamente equivalentes o no.

Clasificación

- Los resultados del test, aplicados sobre el porcentaje de instancias correctamente clasificadas, nos dirá si hay diferencias significativas entre los porcentajes de clasificación correcta de un algoritmo y de otro.

- Si **NO** hay diferencias significativas: Ningún algoritmo es mejor que otro.

- Si **HAY** diferencias significativas: Uno de los algoritmos es mejor que el otro. **¿Cuál?** El que mejor porcentaje de aciertos promedio haya proporcionado (asumiendo normalidad), o el que mejor porcentaje de aciertos mediano haya proporcionado (asumiendo no normalidad).

Índice

- MOA por línea de comandos.
- Clasificación.
- **Concept Drift.**
- Entrega del ejercicio guiado.

- **Ejercicio:**
  - Se pide generar 100.000 instancias utilizando el generador de datos SEAGenerator, con un desvío de concepto centrado en la instancia 20.000 en una ventana de 100 instancias. Para simular el desvío de concepto, hacer que el simulador genere la función –f 2 al principio, y luego la función –f 3.
  - Usar un clasificador Naïve Bayes evaluado con una frecuencia de muestreo de 1.000 instancias, usando el método prequential para evaluación.
  - Inserte la configuración directamente en la GUI de MOA para visualizar la gráfica de la evolución de la tasa de aciertos (medida *accuracy*). ¿Qué se observa?

6.1.2 ConceptDriftStream

Generator that adds concept drift to examples in a stream.

Parameters:

- -s : Stream
- -d : Concept drift Stream
- -p : Central position of concept drift change
- -w : Width of concept drift change

5.7 EvaluatePrequential

Evaluates a classifier on a stream by testing then training with each example in sequence. It may use a sliding window or a fading factor forgetting mechanism.

Parameters:

- Same parameters as EvaluateInterleavedTestThenTrain
- -e : Classification performance evaluation method
  - WindowClassificationPerformanceEvaluator
    - -w : Size of sliding window to use with WindowClassificationPerformanceEvaluator
  - FadingFactorClassificationPerformanceEvaluator
    - -a : Fading factor to use with FadingFactorClassificationPerformanceEvaluator
  - EWMAFactorClassificationPerformanceEvaluator
    - -a : Fading factor to use with FadingFactorClassificationPerformanceEvaluator

6.2.8 generators.SEAGenerator

Generates SEA concepts functions. This dataset contains abrupt concept drift,

Parameters:

- -f: Classification function used, as defined in the original paper
- -i: Seed for random generation of instances
- -b: Balance the number of instances of each class
- -n: Percentage of noise to add to the data

- **Ejercicio (solución):**
  - La tarea es evaluar en prequential, sobre el modelo Naïve Bayes, generando 100.000 instancias con frecuencia de muestreo de 1.000. Dejaremos el flujo para la siguiente diapositiva.

EvaluatePrequential -l bayes.NaiveBayes -s "Aún no sabemos" -i 100000 -f 1000

- **Ejercicio (solución):**
  - El flujo debe tener un desvío de concepto en la instancia 20.000, con una ventana de 100:

ConceptDriftStream -s "DatosAntesDelDrift" -d "DatosDespuésDelDrift" -p 20000 -w 100

  - El flujo de datos antes del drift debe proceder de la función 2 de SEAGenerator. El de después del drift, de la función 3:

generators.SEAGenerator -f 2

generators.SEAGenerator -f 3

Manuel Pegalajar Cuéllar (manupc@decsai.ugr.es)
Dpto. Ciencias de la Computación e I.A., Universidad de Granada

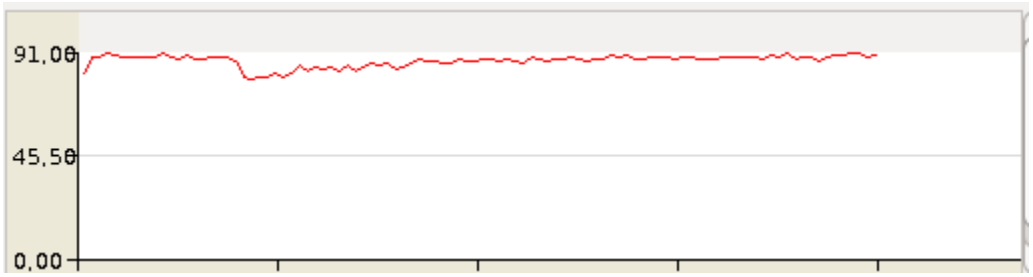
- **Ejercicio (solución):**
  - Por tanto, el flujo de datos del concept drift quedaría así:

ConceptDriftStream -s "generators.SEAGenerator -f 2" -d "generators.SEAGenerator -f 3" -p 20000 -w 100

  - Y la tarea completa, así:

EvaluatePrequential -l bayes.NaiveBayes -s (ConceptDriftStream -s (generators.SEAGenerator -f 2) -d (generators.SEAGenerator -f 3) -p 20000 -w 100) -i 100000 -f 1000

- **Ejercicio (solución):**
  - La gráfica de la evolución de la tasa de clasificación de MOA:



Se observa que, tras el fallo por el desvio de concepto, posteriormente el sistema trata de recuperarse aprendiendo los nuevos datos.

  - Con la precisión:

Evaluation

Values

Measure	Current	Mean
Accuracy	89,20 -	86,63 -
Kappa	72,84 -	68,62 -

Manuel Pegalajar Cuéllar (manupc@decsai.ugr.es)
Dpto. Ciencias de la Computación e I.A., Universidad de Granada

- **Ejercicio:**
  - Entrenar un modelo estático Naïve Bayes sobre 100.000 instancias de la función 2 del generador SEAGenerator.
  - Seguidamente, evaluarlo con un flujo de datos con desvío de concepto generado por el generador de datos SEAGenerator, con un desvío de concepto centrado en la instancia 20.000 en una ventana de 100 instancias. Para simular el desvío de concepto, hacer que el simulador genere la función –f 2 al principio, y luego la función –f 3.

- **Ejercicio (solución):**
  - Por partes. Primero:
    - Entrenar un modelo estático Naïve Bayes sobre 100.000 instancias de la función 2 del generador SEAGenerator.

LearnModel -l bayes.NaiveBayes -s (generators.SEAGenerator -f 2) -m 100000

  - Segundo: Evaluar el modelo sobre 100.000 instancias de un flujo con concept drift.

- **Ejercicio (solución):**
  - Segundo: Evaluar el modelo sobre 100.000 instancias de un flujo con concept drift.

EvaluateModel -m (LearnModel -l bayes.NaiveBayes -s (generators.SEAGenerator -f 2) -m 100000) -s "ElFlujo" -i 100000

El flujo de datos es el mismo que el utilizado en el ejercicio anterior. Por tanto, queda así:

EvaluateModel -m (LearnModel -l bayes.NaiveBayes -s (generators.SEAGenerator -f 2) -m 100000) -s (ConceptDriftStream -s (generators.SEAGenerator -f 2) -d (generators.SEAGenerator -f 3) -p 20000 -w 100) -i 100000

- **Ejercicio (solución):**
  - Resultados:

```
classified instances = 100.000
classifications correct (percent) = 80,344
Kappa Statistic (percent) = 57,301
Kappa Temporal Statistic (percent) = 54,583
Kappa M Statistic (percent) = 39,098
model training instances = 100.000
model serialized size (bytes) = 0.0
```

  - ¿Qué está pasando? Vemos que el porcentaje de clasificación, y también el estadístico Kappa, son inferiores en este segundo ejercicio en comparación con el primero.
  - Si hiciésemos múltiples ejecuciones y comparásemos las distribuciones de resultados, obtendríamos que hay diferencias significativas entre los dos métodos.

- **Ejercicio (solución):**
  - ¿Qué ocurre?
    - El modelo estacionario se ha entrenado con la función f2 de SEA Generator. Falla con la función f3.
    - Al producirse el cambio de concepto, el modelo estacionario no se re-entrena. El modelo dinámico sí.
    - Esto provoca que el modelo dinámico vaya adaptándose con el tiempo a las nuevas condiciones de los datos y que, globalmente, al final proporcione una mejor tasa de aciertos.

- **Ejercicio:**
  - ¿Qué ocurriría si pudiésemos detectar un cambio de concepto y re-entrenar un modelo estacionario?
  - El resultado no sería un modelo "estacionario", sino múltiples de ellos entrenados tras detectar cambio de concepto.
  - Se pide: Evaluar, y entrenar online con el método TestThenTrain, un modelo estacionario Naïve Bayes que se adapta (re-entrena) tras la detección de un cambio de concepto mediante el método DDM (función SingleClassifierDrift). Usar el flujo de datos del ejercicio anterior.

- **Ejercicio (solución):**
  - La tarea es evaluar con TestThenTrain un modelo, en el flujo de datos del ejercicio anterior:

EvaluateInterleavedTestThenTrain –l "modelo" -s (ConceptDriftStream -s (generators.SEAGenerator -f 2) -d (generators.SEAGenerator -f 3) -p 20000 -w 100) -i 100000

  - El modelo se obtiene tras reentrenar un Naïve Bayes cuando se detecta un cambio de concepto con DDM: Tarea SingleClassifierDrift.

- **Ejercicio (solución):**

8.1.2 SingleClassifierDrift

Class for handling concept drift datasets with a wrapper on a classifier.

The drift detection method (DDM) proposed by Gama et al. controls the number of errors produced by the learning model during prediction. It compares the statistics of two windows: the first one contains all the data, and the second one contains only the data from the beginning until the number of errors increases. Their method doesn't store these windows in memory. It keeps only statistics and a window of recent errors.

Example:

```
SingleClassifierDrift -d EDDM -l trees.HoeffdingTree
```

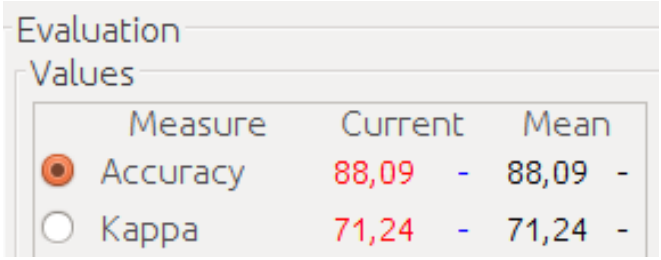
Parameters:

- -l : Classifier to train
- -d : Drift detection method to use: DDM or EDDM

- **Ejercicio (solución):**
  - Por lo tanto, entrenar un Naïve Bayes tras la detección de un cambio de concepto con DDM se realizaría de la siguiente forma:
  - Y la tarea definitiva quedaría así:

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

- **Ejercicio (solución):**
  - Resultados:



  - Si construyésemos una población de resultados e hiciésemos tests estadísticos de comparación entre este modelo y el anterior:
    - Vemos que el porcentaje de aciertos y el estadístico Kappa mejora sustancialmente.
    - Es el resultado esperado: Tras un cambio de concepto, un modelo estacionario deja de funcionar. Se debe reentrenar para ser usado en el nuevo contexto.

Índice

- MOA por línea de comandos.
- Clasificación.
- Concept Drift.
- **Entrega del ejercicio guiado.**

Entrega del trabajo guiado

- **Entrega:**
  - Por PRADO, en el apartado de entrega del trabajo guiado de Minería de Flujos de Datos.
  - La entrega consiste en un PDF que contenga las soluciones a los ejercicios (tareas por línea de comandos), y las conclusiones de cada apartado. Si es necesario, incluir gráficas para aquel o aquellos apartados que lo requieran.
  - El PDF deberá contener el nombrer y apellidos del alumno, así como su e-Mail y DNI/NIE/Pasaporte.

Series Temporales y Minería de flujos de datos

Trabajo Guiado: MOA (*Massive Online Analysis*)

Parte II: Minería de flujos de datos

Manuel Pegalajar Cuéllar (manupc@decsai.ugr.es)
Dpto. Ciencias de la Computación e I.A., Universidad de Granada