

Series Temporales y Minería de flujos de datos

Seminario: Introducción al software MOA
(*Massive Online Analysis*)

Parte II: Minería de flujos de datos

- Prerrequisitos
- Introducción a MOA.
- Configuración de Tareas.
- Primeros pasos: Clasificación.
- Concept Drift en MOA.
- Clustering.
- Ejecución por línea de comandos.

- **Prerrequisitos**
- Introducción a MOA.
- Configuración de Tareas.
- Primeros pasos: Clasificación.
- Concept Drift en MOA.
- Clustering.
- Ejecución por línea de comandos.

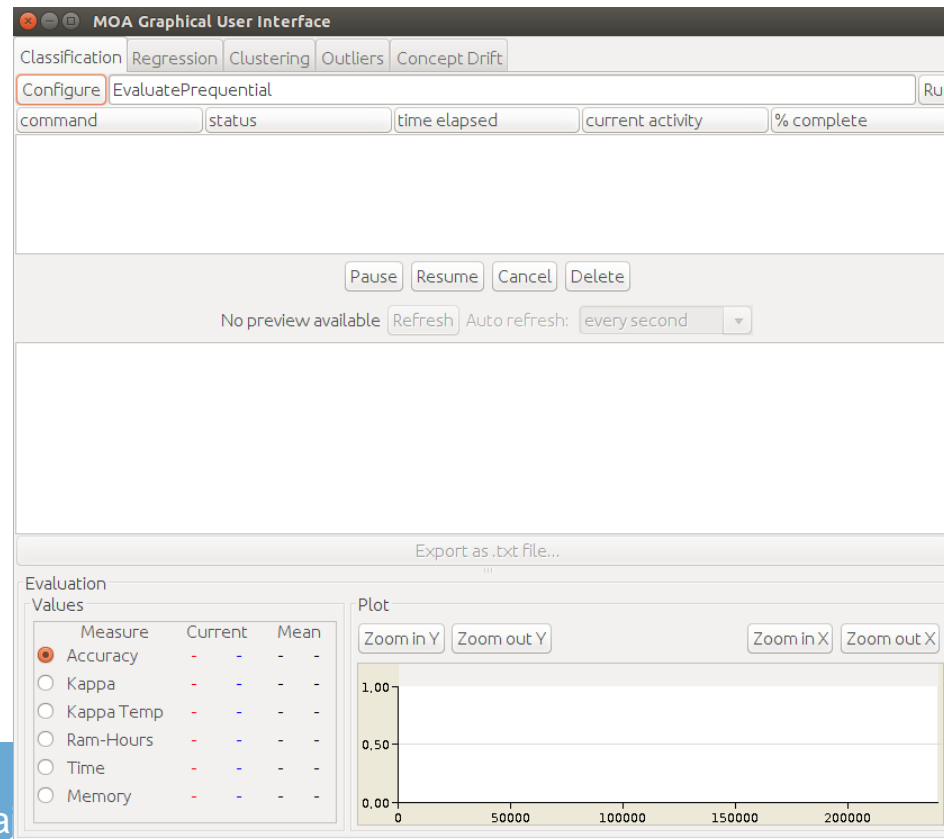
Prerrequisitos

- En este seminario se asume lo siguiente:
 - El alumno conoce las técnicas de minería de flujos de datos impartidas en clase de teoría.
 - El alumno tiene instalado en el PC el Entorno de Ejecución de Java (JRE).
 - El alumno tiene instalado en el PC el software MOA (<http://moa.cms.waikato.ac.nz/>).

- Prerrequisitos
- **Introducción a MOA.**
- Configuración de Tareas.
- Primeros pasos: Clasificación.
- Concept Drift en MOA.
- Clustering.
- Ejecución por línea de comandos.

Introducción a MOA

- MOA (*Massive Online Analysis*) es una herramienta básica para análisis de datos online.
- Dispone de interfaz gráfica y por línea de comandos.



Introducción a MOA

- MOA dispone de una API en Java para acceder a los algoritmos directamente.
- A través de la API, también se pueden incorporar nuevos algoritmos al sistema, o construir plug-ins.

Available Extensions

FREQUENT PATTERN MINING

- **MOA-IncMine** IncMine is an extension for MOA to compute frequent itemsets from data streams. It implements the method proposed by Yiping Ke and Wilfred Ng "Maintaining frequent closed itemsets in a sliding window" Journal of Intelligent Information Systems, 2010, Pages 191-215, using a version of the CHARM algorithm to compute frequent closed itemsets over a batch of transactions.
- **MOA-AdaGraphMiner** AdaGraphMiner is a framework for frequent pattern mining on time-varying streams. It contains three novel algorithms to mine closed subgraphs. All methods work on coresets of closed subgraphs, compute compact representations of graph sets, and maintain these sets in a batch-incremental manner, but use different approaches to address potential concept drift. [Website](#)
- **MOA-Moment** Moment is a closed frequent itemset miner over a stream sliding window. Implemented by Maciek Jarka (www.admire-project.eu). It was presented in: Yun Chi, Haoyun Wang, Philip S. Yu, Richard D. Muntz: Moment: Maintaining

CLASSIFICATION AND REGRESSION

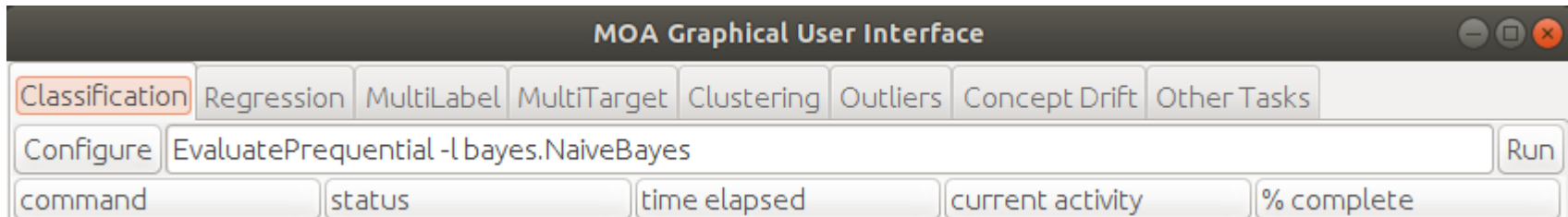
- **IBLStreams** (Instance Based Learner on Streams): is an instance-based learning algorithm for classification and regression problems on data streams. By Ammar Shaker, Eyke Hüllermeier and Jürgen Beringer. [Website](#)
- **MOA-TweetReader** MOA-TweetReader reads and converts tweets from the Twitter Streaming API to MOA instances, so that it is possible to perform data stream mining with them. [Website](#)
- **Classifiers & Drift Detection Methods** This extension provides several published ensemble classifiers and drift detection methods.

STREAM SENTIMENT ANALYSIS

- **Framework for Sentiment Analysis of a Stream of Texts (2012 Harvest Project)** This project aims at building an online, real-time system able to analyze an

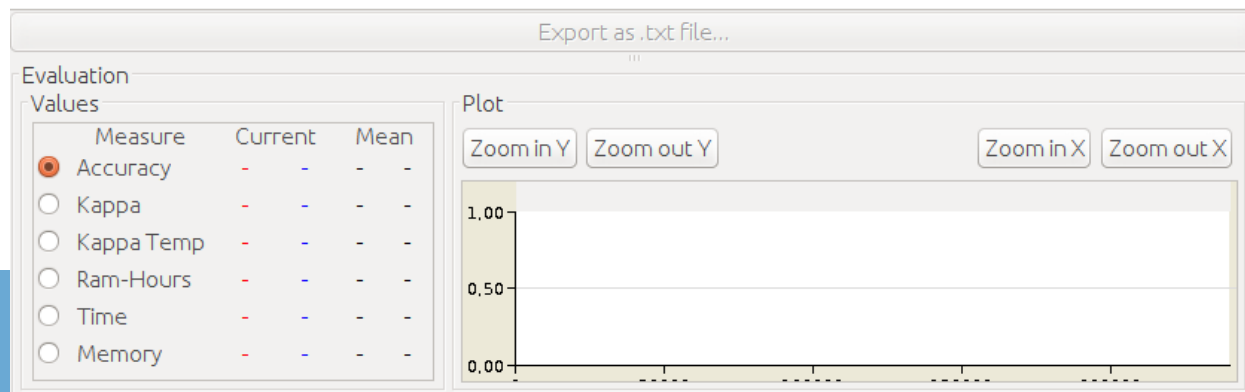
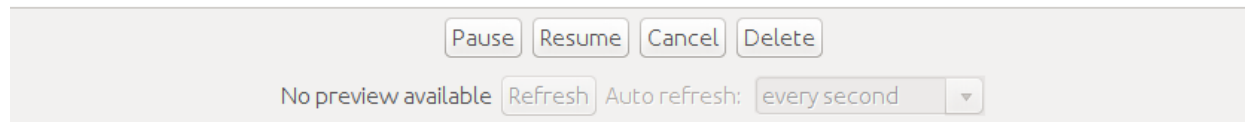
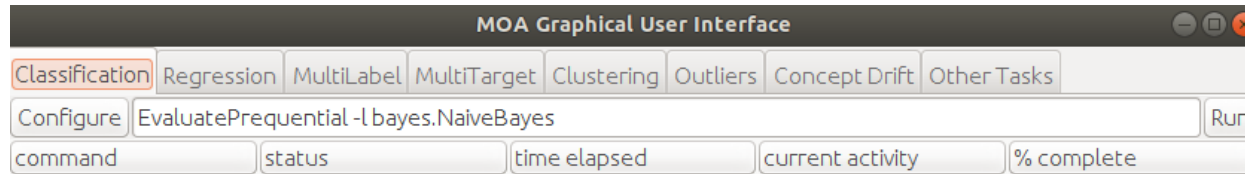
Introducción a MOA

- Por defecto, MOA tiene algoritmos para resolver 7 tipos de problemas: Clasificación (binaria), regresión, clasificación multi-etiqueta, clasificación multidimensional (multi-objetivo), agrupamientos, detección de outliers, detección de cambios de concepto.



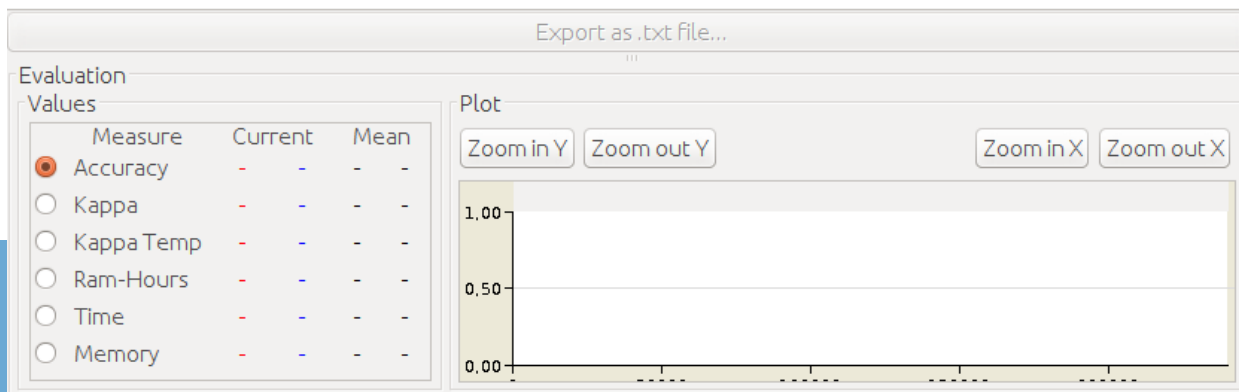
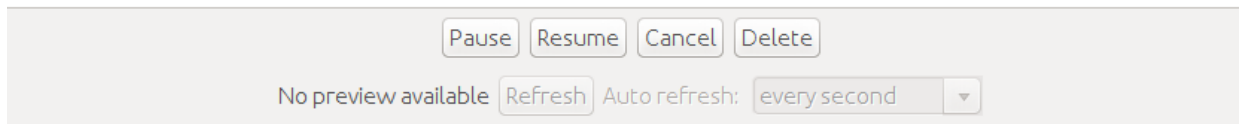
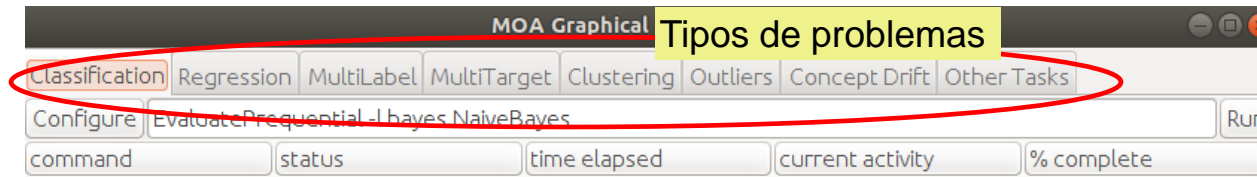
Introducción a MOA

- Un vistazo a la ventana principal de MOA:



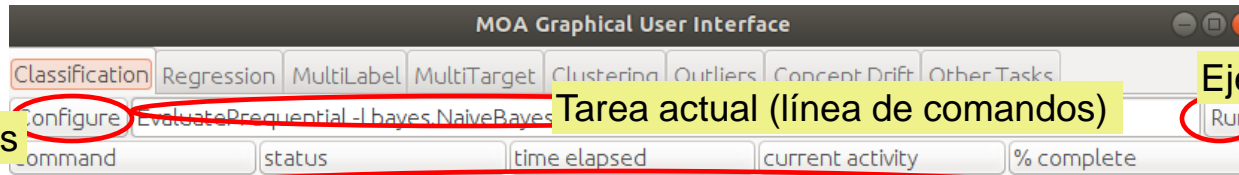
Introducción a MOA

- Un vistazo a la ventana principal de MOA:



Introducción a MOA

- Un vistazo a la ventana principal de MOA:

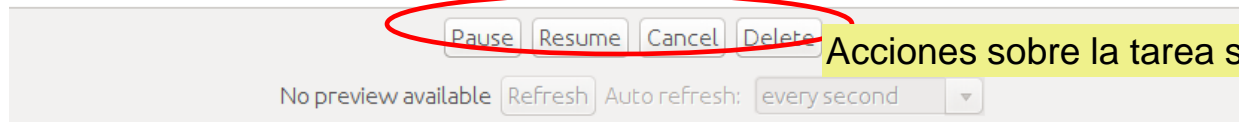


Configurar tareas

Tarea actual (línea de comandos)

Ejecutar tarea actual

Lista de tareas ejecutadas y en curso

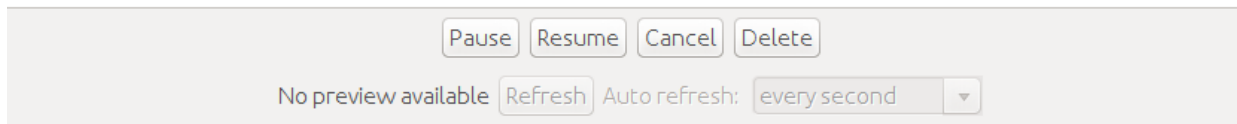
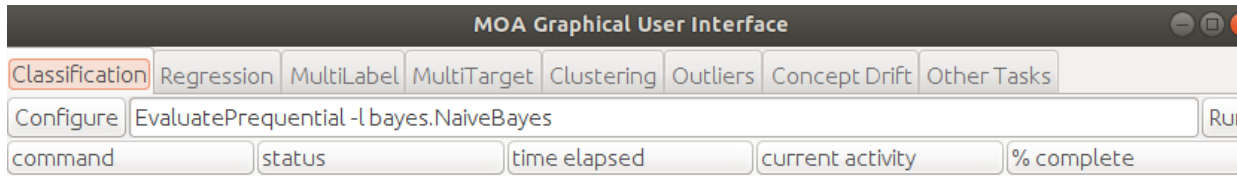


Acciones sobre la tarea seleccionada

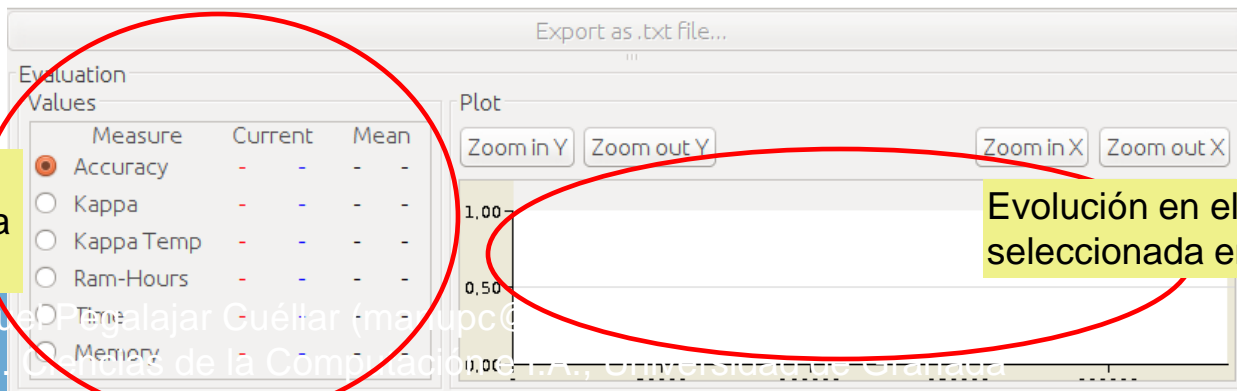


Introducción a MOA

- Un vistazo a la ventana principal de MOA:



Ventana de resultados de la tarea actual



Métricas de evaluación de la tarea actual

Evolución en el tiempo de la métrica seleccionada en la tarea actual

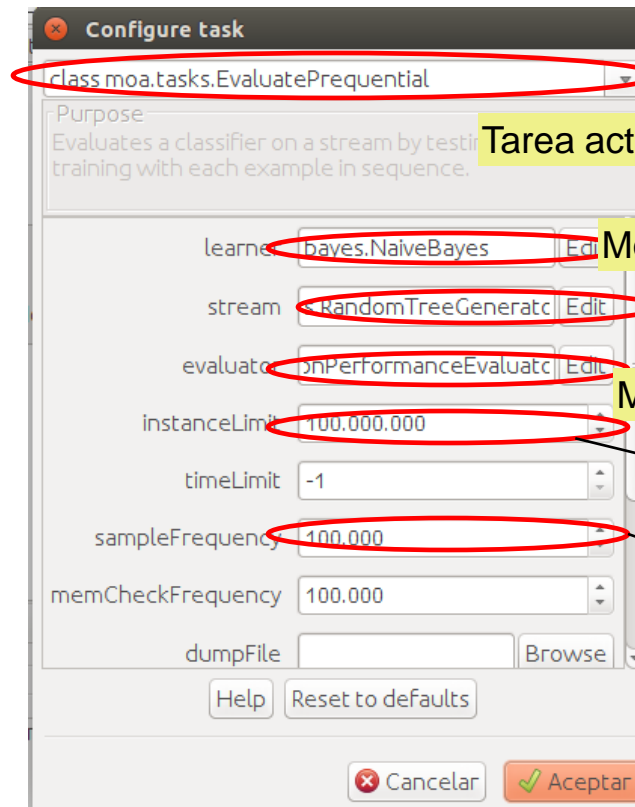
- Prerrequisitos
- Introducción a MOA.
- **Configuración de Tareas.**
- Primeros pasos: Clasificación.
- Concept Drift en MOA.
- Clustering.
- Ejecución por línea de comandos.

Configuración de tareas

- Una tarea consiste en ejecutar un algoritmo concreto para resolver un problema particular. **Ejemplo para clasificación:**



Botón de configuración.
La ventana para configurar la tarea dependerá del problema seleccionado en las pestañas superiores.



Tarea actual a configurar

Modelo de aprendizaje

Flujo de datos

Modelo de evaluación

Instancias a evaluar

Frecuencia de muestreo

Configuración de tareas

- Configuraciones principales:
 - **Evaluación:**
 - **Prequential evaluation:** Cada individuo se utiliza para evaluar el sistema. Inmediatamente después, se usa para entrenarlo.
 - **Heldout:** Se reserva un número de datos para test, se entrena con el resto.

Configuración de tareas

- Configuraciones principales:
 - **Entrenamiento (aprender modelos):** Utiliza todos los datos para entrenar un modelo.
 - **Validación (evaluar modelos):** Utiliza todos los datos para evaluar un modelo previamente aprendido.
 - **Guardar datos a fichero (formato ARFF):** Utiliza los generadores de datos para escribir los resultados a un fichero de salida.

- Prerrequisitos
- Introducción a MOA.
- Configuración de Tareas.
- **Primeros pasos: Clasificación.**
- Concept Drift en MOA.
- Clustering.
- Ejecución por línea de comandos.

Primeros pasos: Clasificación

- **Ejemplo (planteamiento):**
 - Vamos a abordar el problema de **clasificación** de datos online. Para ello, vamos a suponer que tenemos 2 modelos de aprendizaje que queremos validar:
 - **Clasificador de Naïve Bayes, Hoeffding trees**
 - Además, queremos comprobar el comportamiento de ambos en 2 modelos de evaluación:
 - **Prequential, Periodic Heldout**

Primeros pasos: Clasificación

- Ejemplo (planteamiento, continuación):
 - Para probar los modelos, usaremos datos sintéticos generados por un árbol aleatorio (**Random Tree Generator**).
 - Generaremos un total de **1.000.000** de instancias, y haremos el muestreo con una frecuencia de **10.000** instancias.
 - Para Heldout: 999.000 datos de entrenamiento, 1.000 de test.
 - Los datos se analizarán/evaluarán usando una **ventana deslizante de tamaño 1.000**.
 - Finalmente, los criterios de valoración del algoritmo en los que estamos interesados es la precisión (**accuracy**).

Primeros pasos: Clasificación

- **Ejemplo (solución): Tarea *Naïve Bayes-Prequential***
 - 1. Abrir MOA y pulsar “Configure”.
 - 2. Seleccionar la tarea “moa.task.EvaluatePrequential”
 - 3. Seleccionar el learner “moa.classifiers.bayes.NaiveBayes”
 - 4. Seleccionar el stream “... generators.RandomTreeGenerator” (dejar los parámetros por defecto).
 - 5. Seleccionar el evaluador “... BasicClassificationPerformanceEvaluator”
 - 6. Establecer 1.000.000 datos, frecuencia de muestreo 10.000.
 - 7. Dejar el ancho de ventana a 1.000.
 - 8. Pulsar “Aceptar”, “Run”.
-
- **Resultado:** Última precisión “73,63%”. En media: “73,54%”

Primeros pasos: Clasificación

- Ejemplo (solución): Tarea *Naïve Bayes-Prequential*

The screenshot shows the Orange3 software interface. At the top, the 'Classifications' tab is selected. Below it, the 'EvaluatePrequential' widget is configured with the command: `EvaluatePrequential -e BasicClassificationPerformanceEvaluator -i 1000000 -F 1000`. The widget's status is 'completed'. A yellow text box points to the command line, stating: 'Todo por defecto salvo el evaluador, el número de instancias y la frecuencia de muestreo.'

Below the widget, the 'Final result' section shows a list of data points. The 'Export as .txt file...' button is visible.

The 'Evaluation' section displays a table of values:

Measure	Current	Mean
Accuracy	73,63	73,54
Kappa	44,38	44,15
Kappa Temp	45,91	45,65
Ram-Hours	0,00	0,00
Time	1,37	0,70
Memory	0,00	0,00

The 'Plot' section shows a line graph with a red line representing the accuracy over time. The y-axis ranges from 0.00 to 74.00. The x-axis represents time. The graph shows a steady increase in accuracy, reaching a plateau around 73.5.

Primeros pasos: Clasificación

- **Ejemplo (solución): Tarea *Naïve Bayes-Held Out***
- 1. Abrir MOA y pulsar “Configure”.
- 2. Seleccionar la tarea “moa.task.EvaluatePeriodicHeldOutTest”
- 3. Seleccionar el learner “moa.classifiers.bayes.NaiveBayes”
- 4. Seleccionar el stream “... generators.RandomTreeGenerator” (dejar los parámetros por defecto).
- 5. Seleccionar el evaluador “... BasicClassificationPerformanceEvaluator”
- 6. Establecer 999.000 datos de entrenamiento, 1.000 de test, frecuencia de muestreo 10.000.
- 8. Pulsar “Aceptar”, “Run”.
- **Resultado:** Última precisión “71,60%”. En media: “73,50%”



Primeros pasos: Clasificación

- Ejemplo (solución): Tarea *Naïve Bayes-Held Out*

The screenshot displays the Orange3 software interface for a Naïve Bayes classification task. The top section shows the 'Classification' tab selected, with the 'Configure' button and the command 'EvaluatePeriodicHeldOutTest -l bayes.NaiveBayes -n 1000 -i 999000 -f 10000'. Below this is a table showing the status of the evaluation process.

command	status	time elapsed	current activity	% complete
EvaluatePeriodicHeld...	completed	0,51s		100,00
EvaluatePrequential -...	completed	1,37s		100,00

Below the table are buttons for 'Pause', 'Resume', 'Cancel', and 'Delete'. The 'Final result' section shows a 'Refresh' button and an 'Auto refresh' dropdown set to 'every second'. The main area displays a large text output of the evaluation results, including various statistical measures and their values.

Export as .txt file...

Evaluation Values

Measure	Current	Mean
<input checked="" type="radio"/> Accuracy	71,60	73,50
<input type="radio"/> Kappa	40,20	44,13
<input type="radio"/> Kappa Temp	41,08	45,62
<input type="radio"/> Ram-Hours	-	-
<input type="radio"/> Time	0,41	0,21
<input type="radio"/> Memory	0,00	0,00

Plot

Zoom in Y Zoom out Y Zoom in X Zoom out X

The plot shows a line graph with the Y-axis ranging from 0,00 to 78,00. The line is red and shows a fluctuating pattern, indicating the performance of the model over time.

Primeros pasos: Clasificación

- **Ejemplo (solución): Tarea *Hoeffding tree-Prequential***
- 1. Abrir MOA y pulsar “Configure”.
- 2. Seleccionar la tarea “moa.task.EvaluatePrequential”
- 3. Seleccionar el learner “moa.classifiers.trees.HoeffdingTree”
- 4. Seleccionar el stream “... generators.RandomTreeGenerator” (dejar los parámetros por defecto).
- 5. Seleccionar el evaluador “... BasicClassificationPerformanceEvaluator”
- 6. Establecer 1.000.000 datos, frecuencia de muestreo 10.000.
- 7. Dejar el ancho de ventana a 1.000.
- 8. Pulsar “Aceptar”, “Run”.
- **Resultado:** Última precisión “94,45%”. En media: “92,08%”



Primeros pasos: Clasificación

- Ejemplo (solución): Tarea *Hoeffding tree-Prequential*

Classification Regression Clustering Outliers Concept Drift

Configure EvaluatePrequential -l trees.HoeffdingTree -e BasicClassificationPerformanceEvaluator -i 1000000 -f 1000 Run

command	status	time elapsed	current activity	% complete
EvaluatePrequential -...	completed	2,85s		100,00
EvaluatePeriodicHeld...	completed	0,51s		100,00
EvaluatePrequential -...	completed	1,37s		100,00

Pause Resume Cancel Delete

Final result Refresh Auto refresh: every second

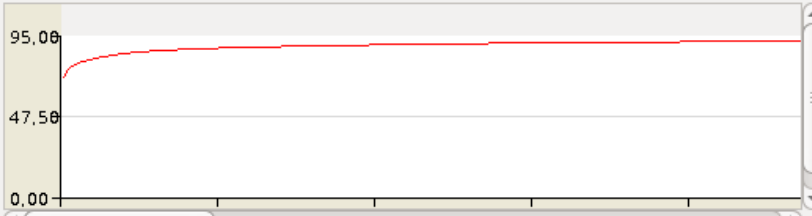
Export as .txt file...

Evaluation Values

Measure	Current	Mean
<input checked="" type="radio"/> Accuracy	94,45	92,08
<input type="radio"/> Kappa	88,61	83,68
<input type="radio"/> Kappa Temp	88,61	83,71
<input type="radio"/> Ram-Hours	0,00	0,00
<input type="radio"/> Time	2,85	1,56
<input type="radio"/> Memory	0,00	0,00

Plot

Zoom in Y Zoom out Y Zoom in X Zoom out X



Primeros pasos: Clasificación

- **Ejemplo (solución): Tarea *Hoeffding tree-Held Out***
- 1. Abrir MOA y pulsar “Configure”.
- 2. Seleccionar la tarea “moa.task.EvaluatePeriodicHeldOutTest”
- 3. Seleccionar el learner “moa.classifiers.trees.HoeffdingTree”
- 4. Seleccionar el stream “... generators.RandomTreeGenerator” (dejar los parámetros por defecto).
- 5. Seleccionar el evaluador “... BasicClassificationPerformanceEvaluator”
- 6. Establecer 999.000 datos de entrenamiento, 1.000 de test, frecuencia de muestreo 10.000.
- 8. Pulsar “Aceptar”, “Run”.
- **Resultado:** Última precisión “96,50%”. En media: “94,62%”

Primeros pasos: Clasificación

- Ejemplo (solución): Tarea *Hoeffding tree-Held Out*

Classification Regression Clustering Outliers Concept Drift

Configure EvaluatePeriodicHeldOutTest -n 1000 -i 999000 -f 10000 Run

command	status	time elapsed	current activity	% complete
EvaluatePeriodicHeld...	completed	1,94s		100,00
EvaluatePrequential -...	completed	2,85s		100,00
EvaluatePeriodicHeld...	completed	0,51s		100,00
EvaluatePrequential -...	completed	1,37s		100,00

Pause Resume Cancel Delete

Final result Refresh Auto refresh: every second

Export as .txt file...

Evaluation Values

Measure	Current	Mean
Accuracy	96,50	94,62
Kappa	92,87	88,96
Kappa Temp	92,74	88,94
Ram-Hours	-	-
Time	1,84	0,96
Memory	0,00	0,00

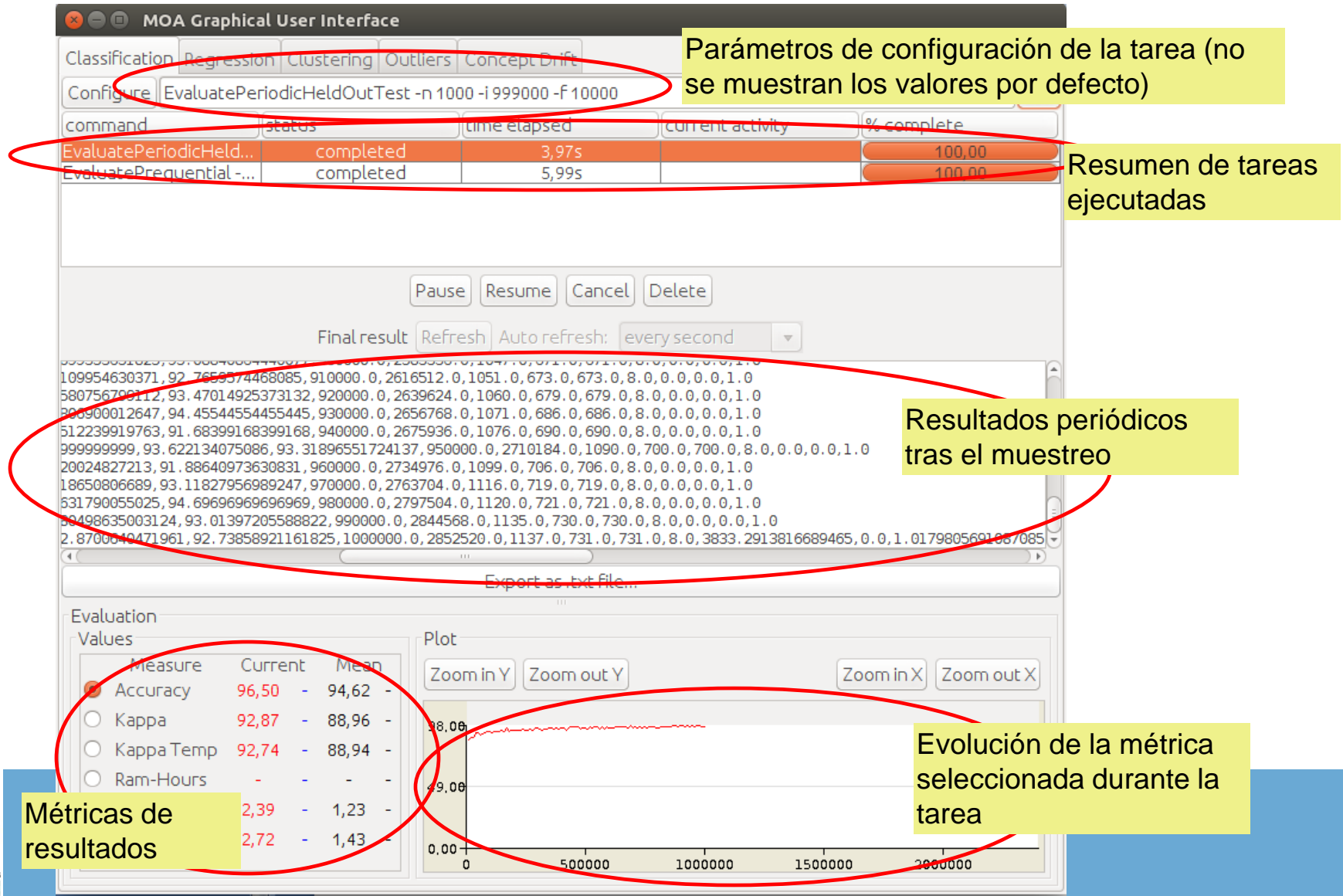
Plot

Zoom in Y Zoom out Y Zoom in X Zoom out X

98,00 49,00 0,00

Primeros pasos: Clasificación

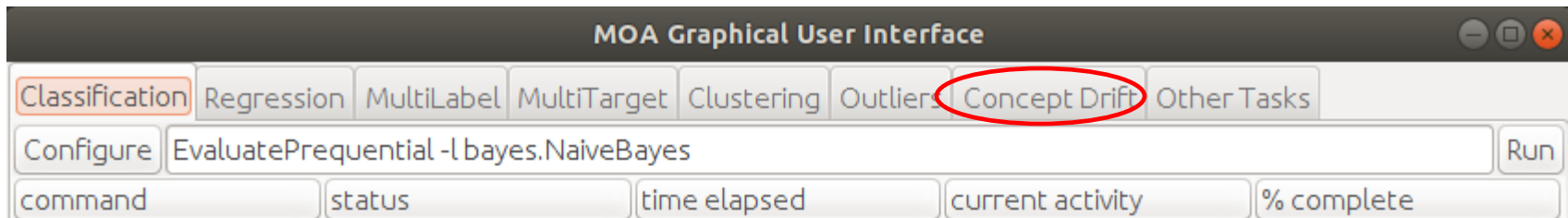
- Ejemplo (solución): Ejemplo de resultados



- Prerrequisitos
- Introducción a MOA.
- Configuración de Tareas.
- Primeros pasos: Clasificación.
- **Concept Drift en MOA.**
- Clustering.
- Ejecución por línea de comandos.

Concept Drift en MOA

- El término *Concept Drift* se utiliza para denominar aquellas situaciones donde las propiedades estadísticas de un fenómeno en estudio cambian a lo largo del tiempo.
- MOA incluye algoritmos y generadores de flujos para detección de concept drift.



- También es posible leer flujos desde fichero (sólo por línea de comandos).

Concept Drift en MOA

- **Ejemplo:** Vamos a generar los datos de un “supuesto” clasificador binario, y le vamos a aplicar un cambio de concepto cada 800 instancias. Lo guardaremos a fichero.
 - 1. Pulsar “Configure” y Seleccionar la tarea “RunStreamTasks”.
 - 2. Pulsar “Edit” en la opción “task”, y seleccionar la tarea “WriteToARFFFile”.
 - 3. Seleccionar como “stream” el “*Abrupt Change Generator*”, con **800** instancias a generar por cada concepto.
 - Establecer el nombre del fichero de salida.
 - 4. Seleccionar 5.000 instancias a generar y ejecutar.
- **BUG:** Existe un bug en la aplicación. Para generar correctamente el fichero, pulsar el botón “Cancel” tras ejecutar la tarea.

Concept Drift en MOA

- **Ejemplo (comprobación):**
 - Tras generar el fichero, se puede abrir con cualquier editor de textos.
 - Sólo nos interesan, para la comprobación, conocer las columnas 1 y 2.
 - La primera columna es la salida de un “supuesto” clasificador binario que hemos estado ejecutando, y para el que queremos comprobar si hay “concept drift”.
 - La segunda columna indica cuándo hay un concept drift real en el conjunto de datos, para validación.

Concept Drift en MOA

- Ejemplo (fichero de salida):

```
1 @relation 'generators.cd.AbruptChangeGenerator -p 800'  
2  
3 @attribute input {0,1}  
4 @attribute change {0,1}  
5 @attribute 'ground truth input' numeric  
6  
7 @data  
8  
9 0,0,0.2  
10 0,0,0.2  
11 0,0,0.2  
12 0,0,0.2  
-- -- -- --
```

```
405 0,0,0.2  
406 0,0,0.2  
407 0,0,0.2  
408 1,1,0.8  
409 1,0,0.8  
410 1,0,0.8  
411 0,0,0.8  
412 1,0,0.8  
413 1,0,0.8  
414 1,0,0.8  
415 1,0,0.8  
416 1,0,0.8  
417 1,0,0.8
```

Hasta la instancia 408 no se produce cambio de concepto. A partir de ahí, la distribución de clasificación del “supuesto” clasificador cambia.

Concept Drift en MOA

- La **detección** de concept drift es importante para conocer cuándo los datos que están siendo evaluados/aprendidos difieren sustancialmente de los previamente presentados.
- Tipos de generadores: Cambio abrupto, cambio paulatino, sin cambio.
- Algoritmos: DDM, ADWIN, CUSUM, Page-Hinkley, etc.
- La clave está en encontrar los parámetros óptimos del algoritmo de detección para evitar falsas alarmas de cambio de concepto.

Concept Drift en MOA

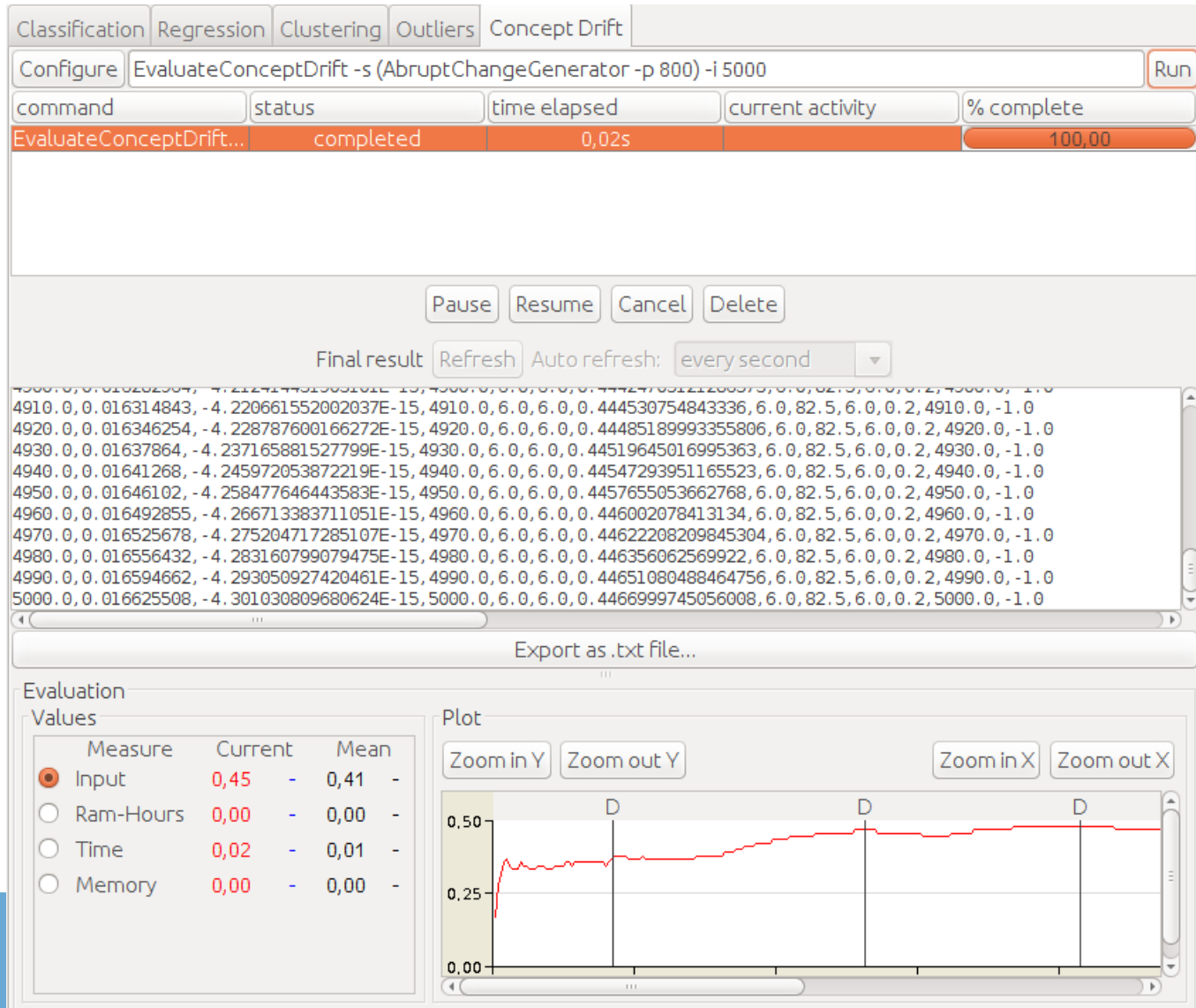
- Ejemplo:

- 1. Seleccionar “*Concept Drift*” desde la ventana principal. Seguidamente, “*Configure*” para establecer una tarea.
- 2. Seleccionar la tarea “*Evaluate Concept Drift*”.
- 3. Pulsar “*Edit*” en *learner*, y seleccionar la técnica *Drift Detection method (DDM)* con **30** instancias de espera antes de lanzar alarma.
- 4. Seleccionar como stream al generador de datos “*Abrupt Change Generator*”, con **800** instancias a generar por cada concepto.
- 5. Establecer **5.000** instancias a probar, con frecuencia de **10**.
- 6. Lanzar la tarea.



Concept Drift en MOA

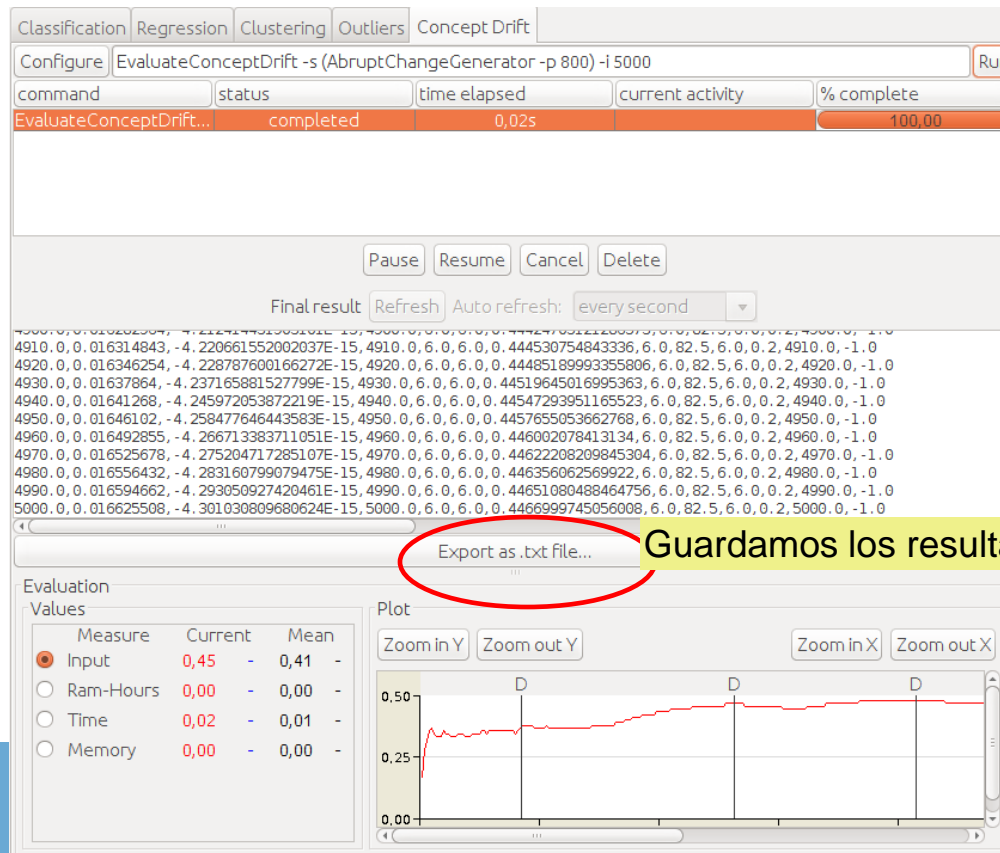
- Ejemplo (resultado):



Concept Drift en MOA

- Ejemplo (resultado):

- ¿Cómo saber el acierto de un método de concept drift? Al igual que en los clasificadores, debemos ir al fichero de resultados que se genera.



Guardamos los resultados a fichero.

Concept Drift en MOA

- Ejemplo (resultado):

- Entre todas las columnas de resultados, comprobamos dos de interés: “True Change” y “Detected Change”

1	model cost (RAM-Hours)	learned instances	detected changes	detected warnings	prediction error (average)	true changes	delay detection (average)	true changes detected	input values	model training instances	model serialized size (bytes)
2	-2.345070242881775E-17	10.0	0.0	0.0	0.15456349206349207	0.0	?	0.0	0.2	10.0	-1.0
3	-1.0146345529291366E-16	20.0	0.0	0.0	0.24874150273531076	0.0	?	0.0	0.2	20.0	-1.0
4	-1.1519451315204304E-16	30.0	0.0	0.0	0.276112880668223	0.0	?	0.0	0.2	30.0	-1.0
5	-1.3295897386140297E-16	40.0	0.0	0.0	0.3229322098270541	0.0	?	0.0	0.2	40.0	-1.0
6	-1.4600835533605685E-16	50.0	0.0	0.0	0.32844518096897185	0.0	?	0.0	0.2	50.0	-1.0
7	-1.5931824843088788E-16	60.0	0.0	0.0	0.3105039990652723	0.0	?	0.0	0.2	60.0	-1.0
8	-1.728208735585213E-16	70.0	0.0	0.0	0.30263503233935685	0.0	?	0.0	0.2	70.0	-1.0
9	-1.841799045602481E-16	80.0	0.0	0.0	0.2951418027622602	0.0	?	0.0	0.2	80.0	-1.0
10	-1.978928533693155E-16	90.0	0.0	0.0	0.2966263808556247	0.0	?	0.0	0.2	90.0	-1.0
11	-2.1087678356303111E-16	100.0	0.0	0.0	0.31687007342580137	0.0	?	0.0	0.2	100.0	-1.0
12	-2.2455118596553804E-16	110.0	0.0	0.0	0.3063463358578414	0.0	?	0.0	0.2	110.0	-1.0

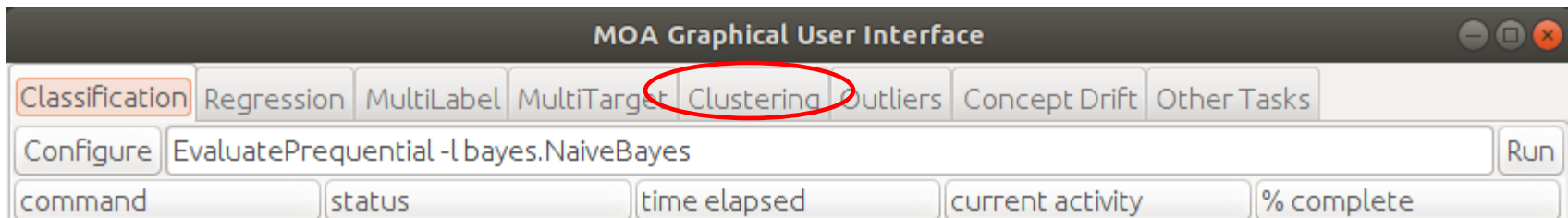
390.0	0.0	0.0	0.31010330743970237	0.0	?
400.0	0.0	0.0	0.3136994191087493	1.0	0.0
410.0	0.0	0.0	0.32403869707281246	1.0	0.0
420.0	0.0	1.0	0.3322711326519372	1.0	0.0
430.0	1.0	1.0	0.3399019169795975	1.0	30
440.0	1.0	1.0	0.33753311439312034	1.0	30
450.0	1.0	1.0	0.33848404020404046	1.0	30

Observaremos que DDM pudo detectar todos los cambios abruptos en el conjunto de datos, aunque con 30 instancias de retraso (tal y como se impuso en la configuración).

- Prerrequisitos
- Introducción a MOA.
- Configuración de Tareas.
- Primeros pasos: Clasificación.
- Concept Drift en MOA.
- **Clustering.**
- Ejecución por línea de comandos.

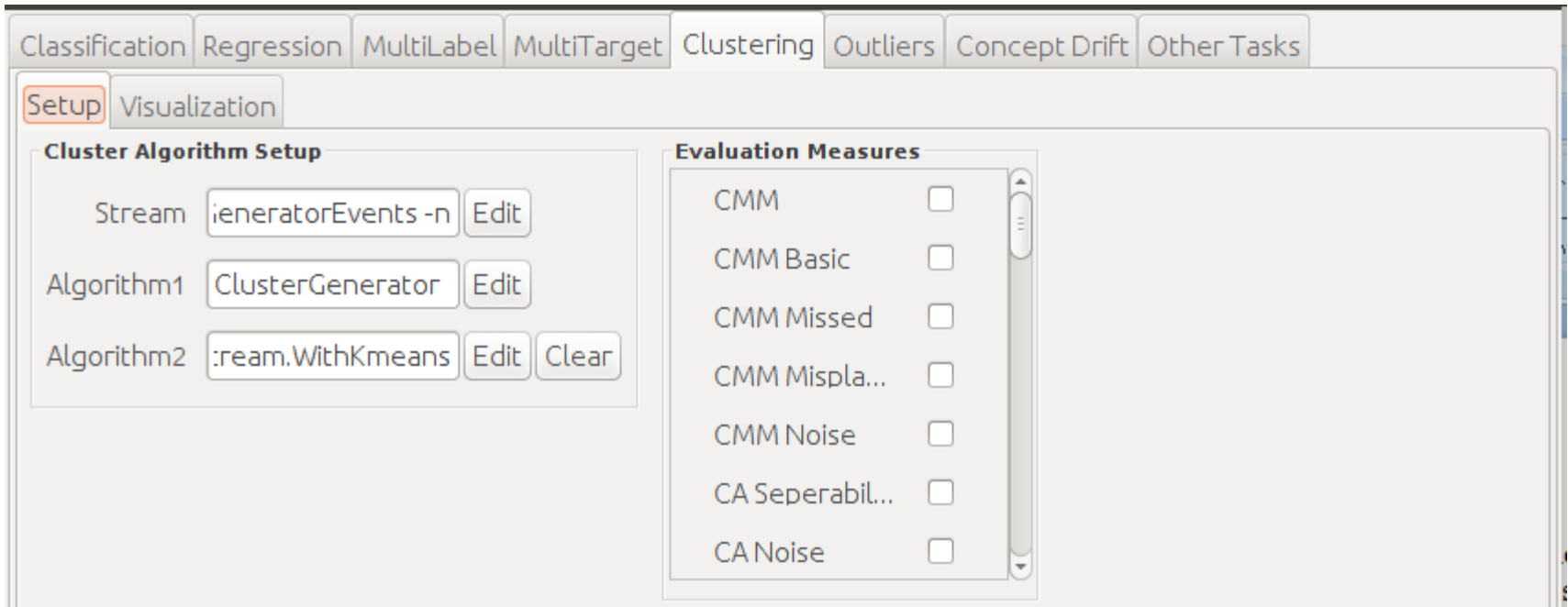
Clustering

- Las tareas de **Clustering**, en análisis de flujos de datos, se orientan a la detección de agrupamientos dentro de un flujo y adaptar los modelos de clusters online, según la evolución de los datos presentados en el flujo.
- MOA presenta una interfaz para análisis de clusters en este tipo de escenarios.



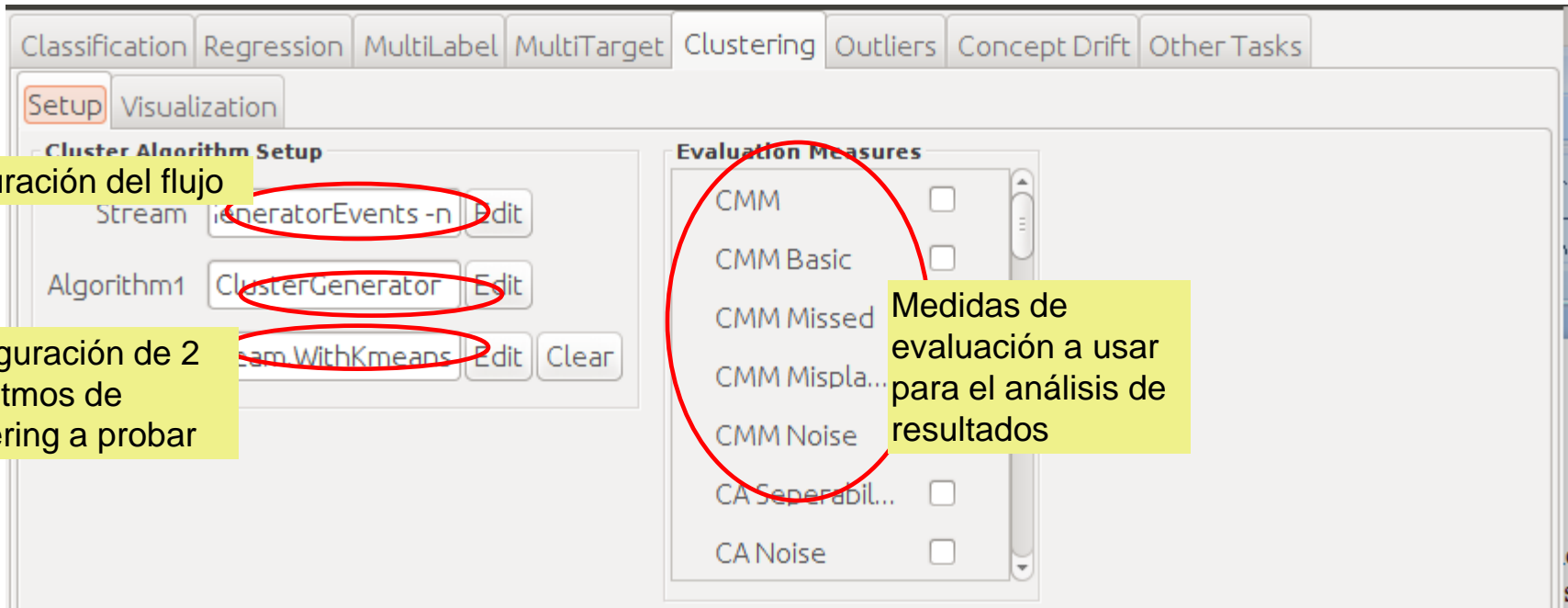
Clustering

- En este caso, y a diferencia de las demás tareas, no existe la opción “Configure” para generar una tarea.
- En su lugar, hay 2 pestañas: Una para configuración de la experimentación, y otra para visualización de resultados.



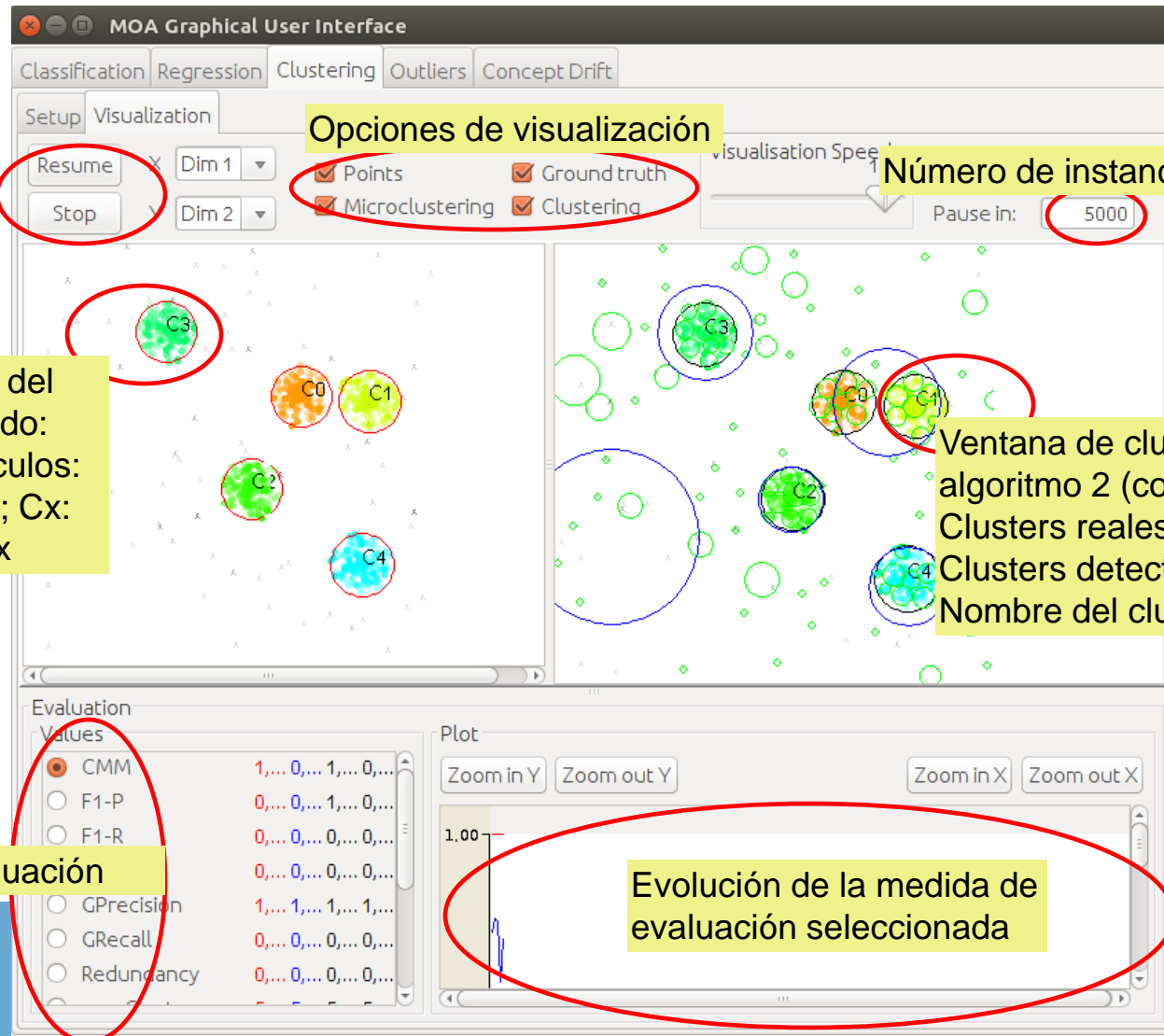
Clustering

- Pestaña de configuración de experimentación con Clustering:



Clustering

- Pestaña de visualización con Clustering:



Opciones para lanzar/parar experimento

Opciones de visualización

Número de instancias del flujo

Ventana de clusters del algoritmo 1 (coloreado: Clusters reales; Círculos: Clusters detectados; Cx: Nombre del cluster x

Ventana de clusters del algoritmo 2 (coloreado: Clusters reales; Círculos: Clusters detectados; Cx: Nombre del cluster x

Medidas de evaluación

Evolución de la medida de evaluación seleccionada

- (Algunas) medidas para evaluación de clusters incluidas en MOA:
 - **CMM** (Cluster Mapping Measure), tiene en cuenta las faltas cometidas en cuanto a elementos no incluidos en el cluster, elementos mal colocados y el ruido.
 - **Pureza**: Porcentaje de la clase mayoritaria de un cluster entre todos los elementos del cluster.
 - **F1-P/F1-R**: Dos formas diferentes de calcular el F-Score. El F-score es una medida de la precisión del clustering, utilizando los valores de Precision y Recall para su cálculo.
 - **Precision**: Valores que pertenecen a la clase entre todos los valores catalogados como de la clase.
 - **Recall**: Valores que pertenecen a la clase entre todos los valores que deberían pertenecer a la clase (aunque no se hayan catalogado como tales).

- (Algunas) medidas para evaluación de clusters incluidas en MOA:
 - **SSQ**: Sum of Squared Distances, distancia de los puntos al centro del cluster.
 - **RAM-Hours**: GB de RAM desplegados en 1h.
 - **Homogeneidad**: Todas las instancias de un cluster pertenecen a la misma clase.
 - **Compleitud**: Todas las instancias de una clase están asignadas al mismo cluster.
 - ...
 - Etc.

- Prerrequisitos
- Introducción a MOA.
- Configuración de Tareas.
- Primeros pasos: Clasificación.
- Concept Drift en MOA.
- Clustering.
- **Ejecución por línea de comandos.**

Ejecución por línea de comandos

- MOA es un sistema soportado por la comunidad y en constante desarrollo.
- Al estar en constante desarrollo, podemos encontrar bugs. No obstante, el código es abierto y existe la posibilidad entrar a modificar errores.
- Pese a estas limitaciones, MOA está muy extendido como marco de referencia para realización de experimentación con flujos de datos.
- La GUI de MOA no es completa. El sistema permite ejecutar más tareas de las que la GUI posibilita.
 - Un ejemplo: La GUI no deja cargar datos desde fichero, salvo que estén en un formato ARFF generado por el propio MOA.

Ejecución por línea de comandos

- Todas las tareas de MOA se pueden ejecutar por línea de comandos (las presentes en la GUI y otras adicionales).

- **Uso de la línea de comandos:**

java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask
“Tarea”

- El texto ***“Tarea”*** debe sustituirse por la tarea a realizar.

- **Ejemplo:**

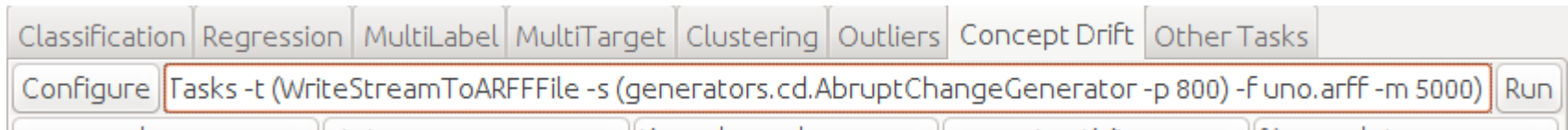
- 1. Lanzar una tarea cualquiera desde la GUI.
- 2. Copiar el texto de la tarea desde la ventana principal.
- 3. Lanzar la misma tarea desde la línea de comandos.

Ejecución por línea de comandos

- **Ejemplo (desde la GUI): Ejercicio con concept drift**
 - 1. Pulsar “Configure” y Seleccionar la tarea “RunStreamTasks”.
 - 2. Pulsar “Edit” en la opción “task”, y seleccionar la tarea “WriteToARFFFile”.
 - 3. Seleccionar como “stream” el generador “*Abrupt Change Generator*”, con **800** instancias. Usar el detector ADWIN con parámetros por defecto.
 - 4. Seleccionar 5.000 instancias a generar al fichero de salida “uno.arff” y ejecutar.
 - 5. Ejecutar el mismo código en línea de comandos, cambiando el fichero de salida a “uno.arff”.
 - **Se generan los mismos datos en 2 ficheros diferentes.**

Ejecución por línea de comandos

- **Ejemplo (desde la GUI): Ejercicio con concept drift**
 - Con interfaz:



- Por línea de comandos:

```
manupc@aquiles: ~/Escritorio/MOASoftware/moa-release-2017.06b/moa-release-2017.06b
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
b$ java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask "RunStreamTasks -t
(WriteStreamToARFFFile -s (generators.cd.AbruptChangeGenerator -p 800) -f uno.ar
ff -m 5000)"

{M}assive {0}nline {A}nalysis
Version: 17.06 June 2017
Copyright: (C) 2007-2017 University of Waikato, Hamilton, New Zealand
Web: http://moa.cms.waikato.ac.nz/
```

Ejecución por línea de comandos

- De especial interés es la opción para obtener flujos de datos desde ficheros externos (en formato **ARFF**), con el fin de evaluar los algoritmos en datos reales u obtenidos desde fuera de MOA.
- **Ejemplo de tarea** para aprender un Hoeffding Tree (para clasificación binaria) desde un fichero externo:
- `LearnModel -l trees.HoeffdingTree -s (ArffFileStream -f RutaFichero.arff) -O RutaFicheroSalidaModelo.moa`

- La introducción a MOA que hemos visto en este seminario es simple.
- Se han estudiado diferentes funcionalidades, dentro de la GUI.
- Para sacar el mayor partido de MOA, se debería utilizar la API de programación en Java.
- Utilizando la API, tenemos total libertad de acción sobre todas las acciones. Podríamos, por ejemplo, ensamblar la salida de un clasificador y su error con un método de detección de concept drift, de modo que tengamos el clasificador y el modelo de detección ejecutándose al mismo tiempo (muy útil para evaluar problemas reales).

Series Temporales y Minería de flujos de datos

Seminario: Introducción al software MOA
(*Massive Online Analysis*)

Parte II: Minería de flujos de datos