Trabajo autónomo II: Minería de Flujo de Datos

Nombre: Francisco Pérez Hernández

E-mail: [herpefran92@gmail.com](mailto:herpefran92@gmail.com)

Asignatura: Series temporales y minería de flujos de datos

Máster: Ciencia de Datos e Ingeniería de Computadores

Fecha de entrega: 05/05/2017

# 1 Parte teórica

## 1.1 Clasificación

### 1.1.1 Problema de clasificación

Tenemos el problema de clasificación, pero no como en el ámbito tradicional, sino que tenemos un flujo continuo de datos que va llegando de forma ordenada y, por tanto, solo se lee una vez, con la que tenemos que manejar esta información y extraer conocimiento de ella conforme van llegando los datos. Además, esta información presenta cambios en el tiempo.

### 1.1.2 Clasificadores utilizados

Tenemos que los dos clasificadores utilizados en esta práctica han sido el Árbol de Decisión Hoeffding y el Árbol de Hoeffding adaptativo. Son algoritmos capaces de construir árboles de decisión a partir de datos que llegan incrementalmente. Además, se basan en la desigualdad de Hoeffding de Teoría de la Probabilidad, la cual proporciona una cota superior a la probabilidad de que la suma de variables aleatorias se desvíe una cierta cantidad de su valor esperado. El criterio habitual para dividir un nodo suele ser por la ganancia de información, por el ratio de ganancia o por la minimización de la entropía.

Las principales características de estos árboles son que mantienen estadísticas en los nodos hoja, no todos los ejemplos son procesados, es incremental y el modelo está disponible en todo momento, no maneja atributos continuos, no contempla la capacidad de adaptarse a cambios en la dinámica de los datos y procesa 1.6 millones de datos en 20 minutos.

Para el modelo adaptativo tenemos que sólo los ejemplos más recientes se tienen en cuenta, mantiene un árbol alternativo por cada nodo, periódicamente evalúa la validad del modelo y sustituye un nodo por el árbol alternativo si mejora, no maneja atributos continuos y tarda entre 5 y 6 veces más que el modelo no adaptativo.

### 1.1.3 Diferentes modos de evaluación/validación en flujo de datos

Tenemos 3 modos principales:

* **TestThenTrain**: En el que cada individuo se utiliza para evaluar el sistema. Inmediatamente después, se usa para entrenarlo.
* **Prequential evaluation**: Es equivalente a TestThenTrain, pero utiliza una ventana deslizante o mecanismos para “olvidar” datos antiguos.
* **Heldout**: Se reserva un número de datos para test y se entrena con el resto.

## 1.2 Concept Drift

### 1.2.1 Problema de concept drift

El problema del desvío de concepto o concept drift es el cambio gradual o abrupto en la distribución de los datos, con lo que el modelo construido no es consistente para estos nuevos datos. El cambio puede deberse al ruido, la influencia del entorno, variación en características no contempladas en el modelo, alteraciones estacionales, etc.

### 1.2.2 Técnicas para resolverlo en clasificación

Tenemos distintas técnicas, donde algunas de ellas lo que hacen es basarse en algoritmos de aprendizaje incremental (online, secuencial) donde no se tienen todos los datos previos, no todos residen en memoria o el aprendizaje no finaliza al obtener el modelo, sino que se adapta continuamente. Principalmente se han dividido los algoritmos que tratan este cambio de concepto en:

* Aprendizaje online
* Algoritmos de detección del desvío
* Soluciones basadas en instancias (ventana deslizante)
* Aprendizaje de múltiples modelos (ensemble)

Tenemos diferentes algoritmos para solucionar este caso, estos algoritmos pueden ser: DDM, ADWIN, CUSUM, Page-Hinkley, etc.

# 2 Parte Práctica

## 2.1 Sección

### 2.1.1 Sección

Para la parte de entrenamiento he usado:

LearnModel -l trees.HoeffdingTree -s (generators.WaveformGenerator -i 2) -m 1000000

Para la parte de evaluación:

EvaluateModel -m trees.HoeffdingTree -s (generators.WaveformGenerator -i 4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Semilla entrenamiento | Semilla evaluación | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 2 | 4 | 33,466 | 0.0 | 0.243 | -0.06256 |
| 1 | 4 | 33,466 | 0.0 | 0,243 | -0.06256 |
| 3 | 4 | 33,466 | 0.0 | 0,243 | -0.06256 |
| 4 | 4 | 33,466 | 0.0 | 0,243 | -0.06256 |
| 5 | 4 | 33,466 | 0.0 | 0,243 | -0.06256 |

### 2.1.2 Sección

Para la parte de entrenamiento he usado:

LearnModel -l trees.HoeffdingAdaptiveTree -s (generators.WaveformGenerator -i 2) -m 1000000

Para la parte de evaluación:

EvaluateModel -m trees.HoeffdingAdaptiveTree -s (generators.WaveformGenerator -i 4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Semilla entrenamiento | Semilla evaluación | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 2 | 4 | 33,466 | 0.0 | 0.243 | -0.06256 |
| 1 | 4 | 33,466 | 0.0 | 0,243 | -0.06256 |
| 3 | 4 | 33,466 | 0.0 | 0,243 | -0.06256 |
| 4 | 4 | 33,466 | 0.0 | 0,243 | -0.06256 |
| 5 | 4 | 33,466 | 0.0 | 0,243 | -0.06256 |

### 2.1.3 Sección

¿Es normal que salgan los mismos resultados?

## 2.2 Sección

### 2.2.1 Sección

Con la orden:

EvaluateInterleavedTestThenTrain -l moa.classifiers.trees.HoeffdingTree -s (generators.WaveformGenerator -i 2) -i 1000000 -f 10000

Cambiaríamos el –i 2 por el valor que deseemos.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 2 | 83.7851 | 75.6775 | 75.6743 | 75.6440 |
| 1 | 83.8903 | 75.8362 | 75.8495 | 75.7872 |
| 3 | 83.8876 | 75.8295 | 75.8560 | 75.7898 |
| 4 | 84.0451 | 76.0694 | 76.0780 | 76.0048 |
| 5 | 83.8402 | 75.7600 | 75.7661 | 75.7110 |

### 2.2.2 Sección

Con la orden:

EvaluateInterleavedTestThenTrain -l moa.classifiers.trees.HoeffdingAdaptiveTree -s (generators.WaveformGenerator -i 2) -i 1000000 -f 10000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 2 | 83.7313 | 75.5967 | 75.5936 | 75.5632 |
| 1 | 83.8042 | 75.7072 | 75.7204 | 75.6577 |
| 3 | 83.7875 | 75.6792 | 75.7060 | 75.6393 |
| 4 | 83.7961 | 75.6960 | 75.7047 | 75.6303 |
| 5 | 83.7144 | 75.5713 | 75.5774 | 75.5219 |

### 2.2.3 Sección

## 2.3 Sección

### 2.3.1 Sección

La orden sería:

EvaluateInterleavedTestThenTrain -l moa.classifiers.trees.HoeffdingTree -s (generators.RandomRBFGeneratorDrift -s 0.001 -a 7 -n 3) -r 2 -i 2000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 2 | 77.4546 | 54.7758 | 54.8527 | 52.9810 |
| 1 | 77.4546 | 54.7758 | 54.8527 | 52.9810 |
| 3 | 77.4546 | 54.7758 | 54.8527 | 52.9810 |
| 4 | 77.4546 | 54.7758 | 54.8527 | 52.9810 |
| 5 | 77.4546 | 54.7758 | 54.8527 | 52.9810 |

### 2.3.2 Sección

La orden sería:

EvaluateInterleavedTestThenTrain -l moa.classifiers.trees.HoeffdingAdaptiveTree -s (generators.RandomRBFGeneratorDrift -s 0.001 -a 7 -n 3) -r 2 -i 2000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 2 | 96.2493 | 92.4914 | 92.4891 | 92.1777 |
| 1 | 96.2493 | 92.4914 | 92.4891 | 92.1777 |
| 3 | 96.2493 | 92.4914 | 92.4891 | 92.1777 |
| 4 | 96.2493 | 92.4914 | 92.4891 | 92.1777 |
| 5 | 96.2493 | 92.4914 | 92.4891 | 92.1777 |

### 2.3.3 Sección

## 2.4 Sección

### 2.4.1 Sección

La orden sería:

RunStreamTasks -t (EvaluatePrequential -l trees.HoeffdingTree -s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -n 3) -i 2000000)

Y con adaptativo:

RunStreamTasks -t (EvaluatePrequential -l trees.HoeffdingAdaptiveTree -s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -n 3) -i 2000000)

### 2.4.2 Sección

## 2.5 Sección

### 2.5.1 Sección

La orden sería:

RunStreamTasks -t (EvaluatePrequential -l (drift.SingleClassifierDrift -l trees.HoeffdingTree) -s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -n 3) -i 2000000)

### 2.5.2 Sección

La orden sería:

RunStreamTasks -t (EvaluatePrequential -l (drift.SingleClassifierDrift -l trees.HoeffdingAdaptiveTree) -s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -a 7 -n 3) -i 2000000)

### 2.5.3 Sección