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 Versión MOA: moa-release-2016.04

# 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, donde se tiene que cumplir los requisitos de que:
  + Cada objeto debe procesarse solo una vez durante el entrenamiento.
  + El sistema debe consumir una memoria y procesamiento limitados con independencia de la cantidad de datos procesada.
  + El aprendizaje se pueda pausar en cualquier momento y su precisión no debería ser peor que la de un clasificador entrenado offline sobre los datos vistos hasta el momento.
* Algoritmos de detección del desvío, donde:
  + Se podría conocer el rendimiento del modelo que se está aprendiendo.
  + Se detecta el desvío de concepto en función de la precisión del clasificador o mediante un análisis de distribución de la clase.
  + Estos algoritmos pueden ser ineficientes y, por tanto, incapaces de asumir una tasa de llegada de datos elevada.
* Soluciones basadas en instancias (ventana deslizante), donde:
  + Los algoritmos incorporan mecanismos para olvidar datos antiguos.
  + Se basan en asumir que los datos llegados recientemente son más relevantes porque contienen características del contexto actual y su relevancia disminuye con el paso del tiempo.
  + Sería conveniente que los datos sean aquellos que han llegado más recientemente, quedando un conjunto de datos con un contexto actual.
* Aprendizaje de múltiples modelos (ensemble), donde:
  + Son algoritmos que incorporan conjuntos de clasificadores elementales (combinación de modelos).
  + Se ha demostrado que una decisión colectiva puede incrementar la precisión porque el conocimiento distribuido entre los clasificadores puede ser más exhaustivo.
  + En entornos estáticos, la diversidad de los modelos, se puede referir al modelo de clasificador, el conjunto de características o las instancias usadas en el entrenamiento.
  + En un entorno cambiante, la diversidad de los modelos, puede referirse también al contexto.

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

Vamos a entrenar un clasificador Hoeffding Tree offline, por lo que usamos “LearnModel”, con “-l” para entrenar el modelo HoeffdinTree, y con “-s” para indicar el flujo de datos “WaveformGenerator” y “-i” para indicar la semilla, la cual la vamos a variar de 1 a 5. Este modelo lo vamos a tratar en un máximo de un millón de instancias con “-m”. Además, lo evaluaremos con “EvaluateModel” sobre el modelo descrito con “-m” y sobre un el flujo de datos proporcionados por “-s” que será un “WaveformGenerator” y semilla “-i” 4. Por lo que el comando ha sido:

EvaluateModel -m (LearnModel -l trees.HoeffdingTree -s (generators.WaveformGenerator -i SEMILLA) -m 1000000) -s (generators.WaveformGenerator -i 4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Semilla entrenamiento | Semilla evaluación | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 4 | 84,509 | 76,765 | 76,774 | 76,702 |
| 2 | 4 | 84,512 | 76,77 | 76,778 | 76,707 |
| 3 | 4 | 84,59 | 76,887 | 76,895 | 76,825 |
| 4 | 4 | 84,666 | 77,001 | 77,009 | 76,939 |
| 5 | 4 | 84,481 | 76,723 | 76,731 | 76,66 |
| Media | | 84,5516 | 76,8292 | 76,8374 | 76,7666 |
| Varianza | | 0,0057 | 0,0129 | 0,0129 | 0,0131 |
| Desviación típica | | 0,0757 | 0,1138 | 0,1136 | 0,1143 |

### 2.1.2 Sección

La diferencia con respecto al apartado anterior es el clasificador, pasando a ser un “HoeffdingAdaptiveTree”. La orden ha sido:

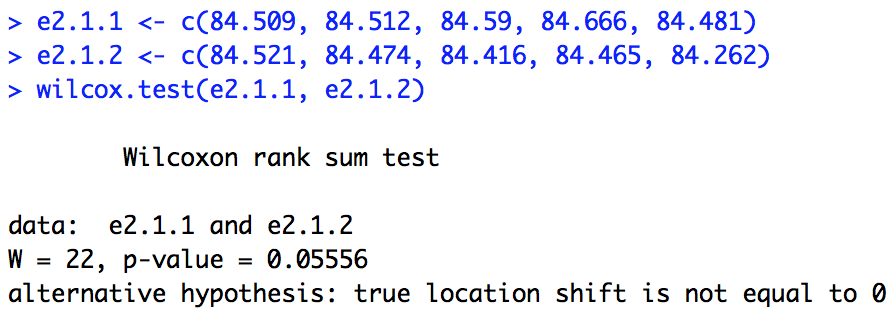
EvaluateModel -m (LearnModel -l trees. HoeffdingAdaptiveTree -s (generators.WaveformGenerator -i SEMILLA) -m 1000000) -s (generators.WaveformGenerator -i 4)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Semilla entrenamiento | Semilla evaluación | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 4 | 84,521 | 76,783 | 76,792 | 76,721 |
| 2 | 4 | 84,474 | 76,712 | 76,721 | 76,65 |
| 3 | 4 | 84,416 | 76,625 | 76,634 | 76,562 |
| 4 | 4 | 84,465 | 76,699 | 76,707 | 76,636 |
| 5 | 4 | 84,262 | 76,395 | 76,403 | 76,331 |
| Media | | 84,4276 | 76,6428 | 76,6514 | 76,5800 |
| Varianza | | 0,0100 | 0,0223 | 0,0224 | 0,0226 |
| Desviación típica | | 0,0998 | 0,1494 | 0,1498 | 0,1502 |

### 2.1.3 Sección

Como podemos ver en las tablas anteriores, la diferencia entre ambos modelos es mínima, ya que el primero obtiene de media un 84,55% de aciertos mientras el segundo obtiene un 84,42%. Podríamos decir que el primero es ligeramente mejor para este problema.

Si realizamos un test de Wilcoxon tenemos que:



Podríamos decir que no existen diferencias significativas entre ambos ya que solo tenemos un 94,4% de confianza en que sean distintos.

## 2.2 Sección

### 2.2.1 Sección

Vamos a entrenar un HoeffdingTree online, mediante el modo “EvaluateInterleavedTestThenTrain” y la opción “-l” Además tendremos que el máximo número de instancias “-i” es de un millón y que la frecuencia de muestreo “-f” será de 10000. El generador será un “WaveformGenerator” y la semilla “-i” la modificaremos entre 1 y 5. Con la orden:

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 83,8903 | 75,8362 | 75,8495 | 75,7872 |
| 2 | 83,7851 | 75,6774 | 75,6743 | 75,6440 |
| 3 | 83,8876 | 75,8295 | 75,8560 | 75,7898 |
| 4 | 84,0451 | 76,0694 | 76,0780 | 76,0048 |
| 5 | 83,8402 | 75,7599 | 75,7660 | 75,7110 |
| Media | 83,8897 | 75,8345 | 75,8448 | 75,7874 |
| Varianza | 0,0094 | 0,0214 | 0,0225 | 0,0184 |
| Desviación típica | 0,0969 | 0,1462 | 0,1498 | 0,1357 |

### 2.2.2 Sección

La diferencia con respecto al anterior será que hemos cambiado el modelo por un “HoeffdingAdaptiveTree”. Con la orden:

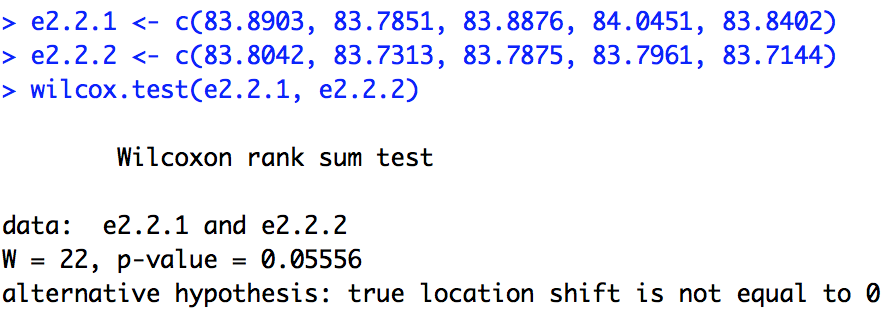
EvaluateInterleavedTestThenTrain -l moa.classifiers.trees.HoeffdingAdaptiveTree -s (generators.WaveformGenerator -i SEMILLA) -i 1000000 -f 10000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 83,8042 | 75,7072 | 75,7204 | 75,6577 |
| 2 | 83,7313 | 75,5968 | 75,5936 | 75,5632 |
| 3 | 83,7875 | 75,6792 | 75,7060 | 75,6393 |
| 4 | 83,7961 | 75,6960 | 75,7047 | 75,6303 |
| 5 | 83,7144 | 75,5712 | 75,5774 | 75,5219 |
| Media | 83,7667 | 75,6501 | 75,6604 | 75,6025 |
| Varianza | 0,0017 | 0,0038 | 0,0047 | 0,0033 |
| Desviación típica | 0,0409 | 0,0618 | 0,0689 | 0,0575 |

### 2.2.3 Sección

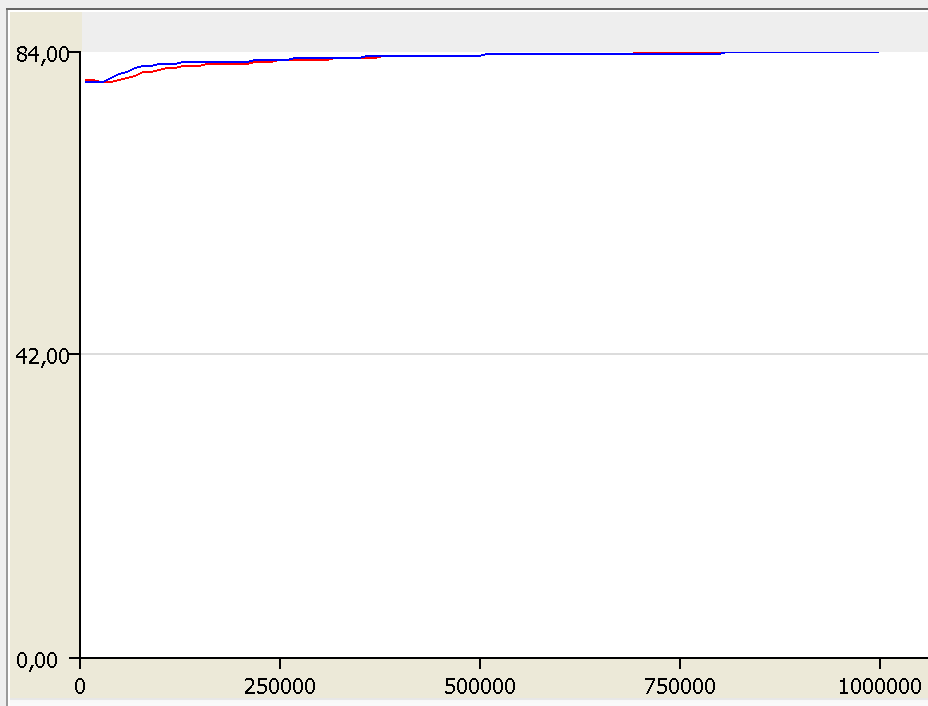
Vemos como pasaba en la sección 2.1 que la diferencia entre ambos es mínima, y que podría ser ligeramente mejor, para este problema, el modelo “HoeffdingTree” al tener un porcentaje de aciertos superior.

Si realizamos un test de Wilcoxon tenemos que:



Podríamos decir que no existen diferencias significativas entre ambos ya que solo tenemos un 94,4% de confianza en que sean distintos.

Si observamos su gráfica (semilla igual a 5), podemos ver como ambos modelos se comportan de igual forma:



## 2.3 Sección

### 2.3.1 Sección

Realizamos un “EvaluateInterleavedTestThenTrain” sobre un clasificador “-l” que es el “HoeffdingTree” con un número de instancias “-i” de dos millones y un flujo de datos “-s” que es el “RandomRBFGeneratorDrift” en el que la velocidad de cambio del centroide “-s” es de 0.001, el número de centroides con drift “-k” es 3, una semilla del modelo “-r” que va variando entre 1 y 5 como también la semilla de las instancias “-i” y que tiene un número de atributos “-a” igual a 7 y el número de centroides en el modelo “-n” igual a 3. La orden sería:

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 77,4545 | 54,7757 | 54,8526 | 52,9810 |
| 2 | 83,9565 | 29,3917 | 43,1454 | 5,5781 |
| 3 | 73,4734 | 43,0588 | 43,9611 | 30,7641 |
| 4 | 73,0467 | 44,9855 | 44,8941 | 36,9974 |
| 5 | 72,0616 | 43,1521 | 43,3551 | 36,3311 |
| Media | 75,9985 | 43,0728 | 46,0417 | 32,5303 |
| Varianza | 24,0095 | 81,9488 | 24,7211 | 295,5432 |
| Desviación típica | 4,9000 | 9,0526 | 4,9720 | 17,1914 |

### 2.3.2 Sección

El cambio con respecto al anterior es el clasificador ya que ahora tendremos un “HoeffdingAdaptiveTree”. La orden sería:

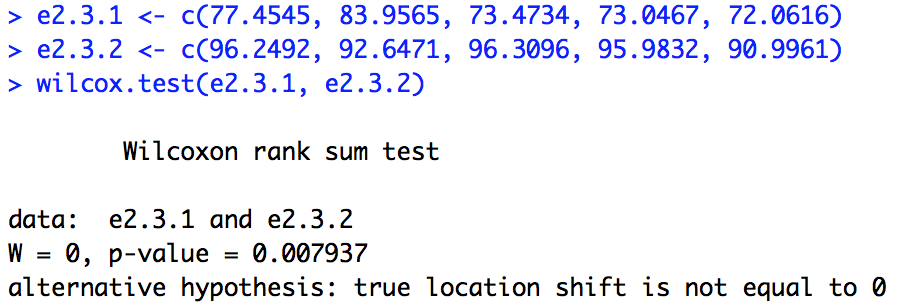
EvaluateInterleavedTestThenTrain -l moa.classifiers.trees. HoeffdingAdaptiveTree -s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -r SEMILLA -i SEMILLA -a 7 -n 3) -i 2000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 96,2492 | 92,4913 | 92,4891 | 92,1777 |
| 2 | 92,6471 | 73,9267 | 73,9430 | 56,7256 |
| 3 | 96,3096 | 92,1943 | 92,2040 | 90,3681 |
| 4 | 95,9832 | 91,7716 | 91,7876 | 90,6108 |
| 5 | 90,9961 | 81,6579 | 81,7447 | 79,4810 |
| Media | 94,4370 | 86,4084 | 86,4337 | 81,8726 |
| Varianza | 6,0562 | 69,4006 | 69,1582 | 223,2040 |
| Desviación típica | 2,4609 | 8,3307 | 8,3161 | 14,9400 |

### 2.3.3 Sección

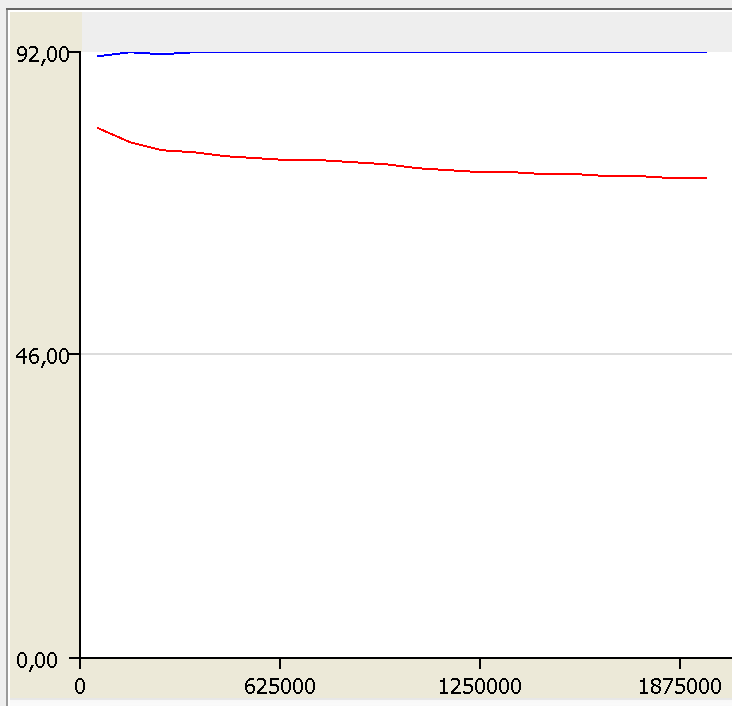
Si nos fijamos en los datos proporcionados por estos modelos, vemos como la media para el acierto de clasificación con el modelo normal es de 75% mientras que para el modelo adaptativo es del 94%, con lo que se podría decir que el modelo adaptativo, como era de esperar, funciona mejor en un tipo de problemas como este en el que tenemos un cambio de concepto.

Si realizamos un test de Wilcoxon tenemos que:



Podríamos decir que existen diferencias significativas entre ambos ya que tenemos un 99,2% de confianza en que sean distintos.

Si observamos su gráfica (semilla igual a 5), podemos ver como los modelos se comportan de una forma distinta, siendo mejor el model HoeffdingAdativeTree (azul):



## 2.4 Sección

### 2.4.1 Sección

Si vemos los cambios con respecto al apartado anterior de la sección 2.3, vemos como el cambio es el tipo de evaluación, de forma que tenemos un “EvaluatePrequential” con el que iremos olvidando instancias pasadas. La orden sería:

EvaluatePrequential -l trees.HoeffdingTree -s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -r $SEMILLA -i $SEMILLA -a 7 -n 3) -i 2000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 76,6 | 52,5587 | 53,6633 | 51,0460 |
| 2 | 86,5 | 39,8256 | 48,0769 | 14,5569 |
| 3 | 52,9000 | -0,10626 | 0,8421 | -25,59999 |
| 4 | 65,3 | 29,6037 | 33,0115 | 21,4932 |
| 5 | 63,3 | 25,2844 | 24,6406 | 16,2100 |
| Media | 68,9200 | 29,4332 | 32,0469 | 15,5412 |
| Varianza | 167,3420 | 383,1530 | 438,4382 | 747,5077 |
| Desviación típica | 12,9361 | 19,5743 | 20,9389 | 27,3406 |

Y con adaptativo cambiando solamente el clasificador “HoeffdingAdaptiveTree”:

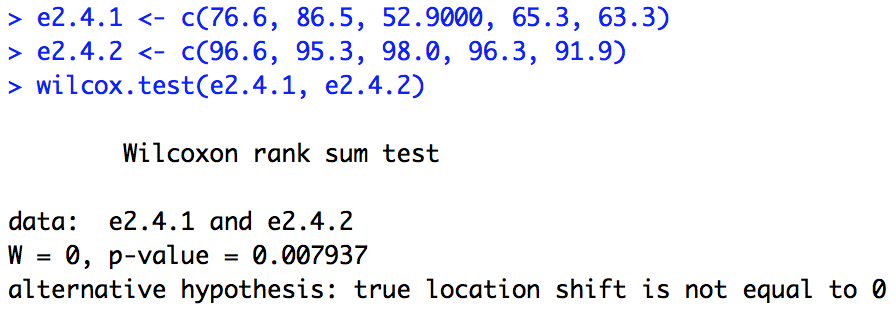
EvaluatePrequential -l trees. HoeffdingAdaptiveTree -s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -r $SEMILLA -i $SEMILLA -a 7 -n 3) -i 2000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 96,6 | 93,1907 | 93,2673 | 92,8870 |
| 2 | 95,3 | 82,1983 | 81,9230 | 70,2531 |
| 3 | 98,0 | 95,7559 | 95,7894 | 94,6666 |
| 4 | 96,3 | 92,4689 | 92,8571 | 91,6289 |
| 5 | 91,9 | 83,4006 | 83,3675 | 81,5068 |
| Media | 95,6200 | 89,4029 | 89,4409 | 86,1885 |
| Varianza | 5,2570 | 38,0106 | 40,0051 | 105,5503 |
| Desviación típica | 2,2928 | 6,1653 | 6,3250 | 10,2738 |

### 2.4.2 Sección

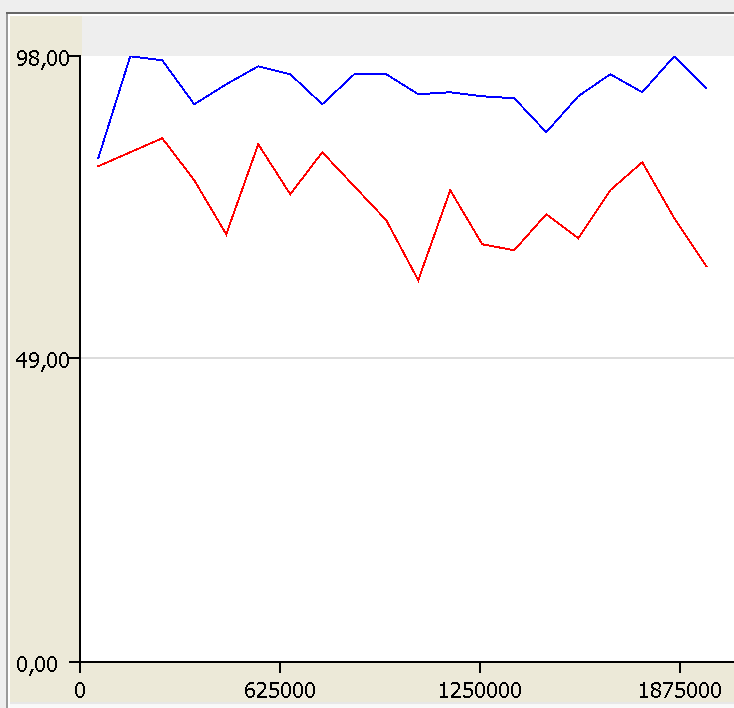
Podemos ver que, en el modelo normal, tenemos una variación de los resultados muy grande comparado con el modelo adaptativo con el que tenemos unos resultados más estables de media. Además, pasamos de un 68% de acierto de media, a un 95% de acierto con el modelo adaptativo, como era de esperar para este tipo de problema en el que vamos olvidando instancias y tenemos un cambio de concepto. Además, hemos pasado de un 94,43% de acierto en la sección 2.3, a un 95,62% de acierto en este modelo en el que olvidamos instancias, por lo que podríamos decir que, aparentemente, es mejor olvidar instancias pasadas.

Si realizamos un test de Wilcoxon tenemos que:



Podríamos decir que existen diferencias significativas entre ambos ya que tenemos un 99,2% de confianza en que sean distintos.

Si observamos su gráfica (semilla igual a 5), podemos ver como los modelos se comportan de una forma distinta, siendo mejor el model HoeffdingAdativeTree (azul):



## 2.5 Sección

### 2.5.1 Sección

Vamos a incluir mecanismos para reinicializar modelos tras la detección de cambios de concepto, por lo que hemos cambiado, con respecto al apartado 2.3, ya que ahora nuestro clasificador “-l” será un clasificador “SingleClassifierDrift” donde su clasificador es un “HoeffdingTree”. La orden sería:

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 97,1995 | 94,3919 | 94,3919 | 94,1595 |
| 2 | 90,5015 | 65,5687 | 66,3395 | 44,0979 |
| 3 | 97,0895 | 93,8475 | 93,8514 | 92,4034 |
| 4 | 96,5851 | 93,0011 | 93,0182 | 92,0177 |
| 5 | 92,5628 | 84,8316 | 84,9212 | 83,0514 |
| Media | 94,7877 | 86,3282 | 86,5044 | 81,1460 |
| Varianza | 9,4168 | 149,8218 | 141,9376 | 447,6166 |
| Desviación típica | 3,0687 | 12,2402 | 11,9138 | 21,1570 |

### 2.5.2 Sección

Hemos cambiado, con respecto al apartado anterior, el clasificador interno, que ahora es un “HoeffdingAdaptiveTree”. La orden sería:

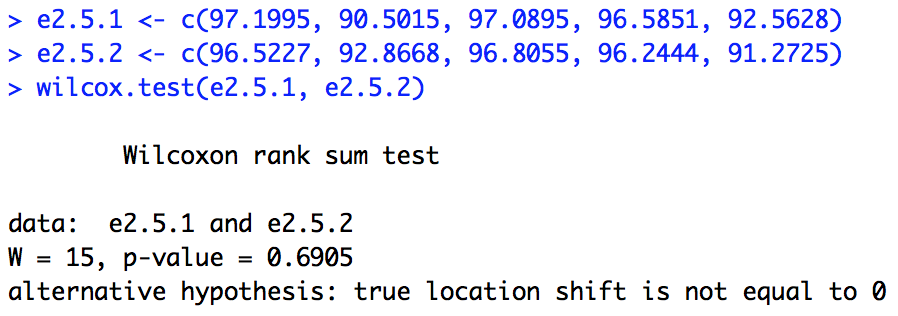
EvaluateInterleavedTestThenTrain -l (moa.classifiers.drift.SingleClassifierDrift -l trees. HoeffdingAdaptiveTree) -s (generators.RandomRBFGeneratorDrift -s 0.001 -k 3 -r $SEMILLA -i $SEMILLA -a 7 -n 3) -i 2000000

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Semilla | Aciertos clasificación | Estadístico Kappa | Estadístico Kappa temporal | Estadístico Kappa M temporal |
| 1 | 96,5227 | 93,0386 | 93,0367 | 92,7481 |
| 2 | 92,8668 | 74,7934 | 74,7214 | 58,0183 |
| 3 | 96,8055 | 93,2458 | 93,2514 | 91,6621 |
| 4 | 96,2444 | 92,3027 | 92,3218 | 91,2215 |
| 5 | 91,2725 | 82,2160 | 82,3050 | 80,1108 |
| Media | 94,7424 | 87,1193 | 87,1273 | 82,7522 |
| Varianza | 6,3100 | 68,8525 | 69,1423 | 217,4440 |
| Desviación típica | 2,5120 | 8,2977 | 8,3152 | 14,7460 |

### 2.5.3 Sección

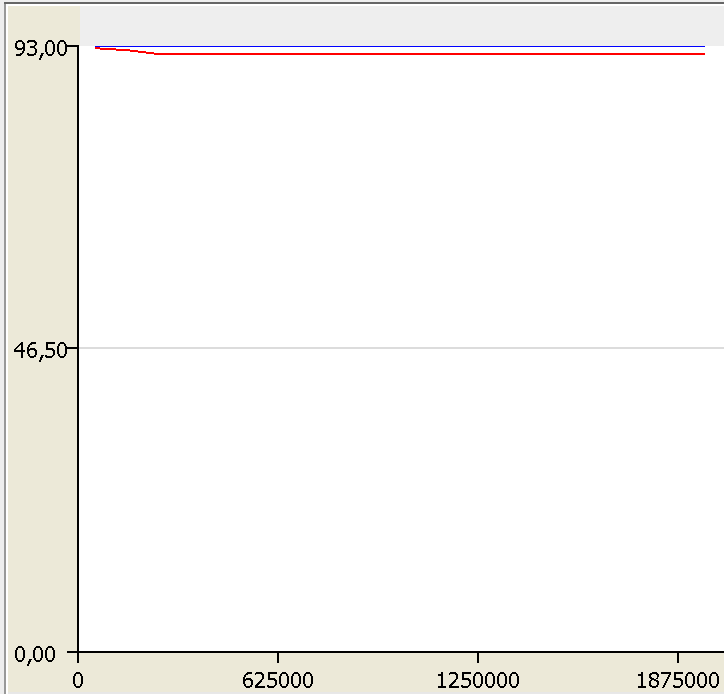
Si observamos los resultados de la sección 2.5 vemos como no hay cambios significativos entre ambos modelos, ya que como se produce reinicialización no hay mucha diferencia entre los modelos, siendo un poco difícil elegir un modelo como mejor que otro, pero si nos fiamos en las varianzas y desviaciones típicas, podríamos decir que el modelo adaptativo se comporta algo mejor y más estable que el normal.

Si realizamos un test de Wilcoxon tenemos que:



Podríamos decir que no existen diferencias significativas entre ambos ya que solo tenemos un 30,9% de confianza en que sean distintos.

Si observamos su gráfica (semilla igual a 5), podemos ver como ambos modelos se comportan de igual forma:



Con respecto a los apartados 2.3, 2.4 y 2.5 tenemos diferencias. En el 2.3 tenemos que el problema es de cambio de concepto y el método adaptativo funciona mejor ya que consigue adaptarse a este cambio. Con respecto al 2.4, ocurre lo mismo, pero funcionando un poco mejor que el 2.3 ya que al olvidar instancias pasadas no se condiciona el modelo. En estos dos apartados, el modelo adaptativo ha funcionado mejor que el modelo normal, pero en el apartado 2.5 vemos cómo funcionan ambos modelos casi igual. El problema es que, en este último apartado tenemos que los modelos se reinicializan tras el cambio de concepto, por lo que puede ser que en algunas situaciones nos perjudique el no saber lo que paso anteriormente a la hora de construir el modelo, pero en situaciones como las vistas durante este trabajo, los resultados no se ven afectados para estas técnicas.