Metodos de Machine Learning sobre spikes neuronales

Tesis de Grado para la obtención
del título de Ingeniero en Informatica
realizada por

Matías Leonel Egea

Dirigida por el Doctor Ingeniero
Sergio Lew

Facultad de Ingeniería
Universidad de Buenos Aires

Marzo 2015

Metodos de Machine Learning sobre spikes neuronales

Tesis de grado

Ingeniería en Informatica

2015/3

Facultad de Ingeniería
Universidad de Buenos Aires

Marzo 2015

Índice

1. Introducción **1**
1.1. Introducción 1

2. Set de datos **2**
2.1. Introducción 2
2.2. Orígen de los Datos 2

3. Métodos empleados **4**
3.1. Introducción 4
3.2. Métodos 4
3.2.1. Análisis Discriminante 4
3.2.2. Random Forest 5
3.2.3. Support Vector Machine 6
3.2.4. Bayes Naive 7

4. Interpretación del problema **8**
4.1. Introducción 8
4.2. Interpretaciones 8
4.2.1. Analisis mono-neuronal 8
4.2.2. Análisis multi-neuronal 10

5. Enfoque por ventanas **12**
5.1. Introducción 12
5.2. Implementación 12
5.3. Resultados 14

6. Iteraciones y resultados **16**
6.1. Introducción 16
6.2. Análisis con LDA 17
6.3. Análisis con Bayes-Naive 18
6.4. Análisis con SVM 19
6.5. Análisis con Random Forest 20

7. Comparaciones y corridas apareadas **21**
7.1. Introducción 21
7.2. Comparación entre los métodos 22
7.3. Resultados apareados 26
7.3.1. Comparación 26
7.3.2. Conclusión 29

8. Conjunto reducido **30**
8.1. Introducción 30
8.2. Análisis comparativo 31
8.3. Conclusión 36

9. Tamaño de ventana **37**
9.1. Introducción 37
9.2. Exploración de tamaños 37
9.3. Conclusión 41

Apéndice A. Set de datos **42**
A.1. Formato y distribución de los datos 42
A.2. Observaciónes y análisis de los datos 42

Bibliografía **44**

Índice de figuras

3.1. Separación lineal de 2 clases. 5
3.2. Separacion con margen. Los vectores p1 y p3 son los llamados *support vectors* 6

4.1. Matriz de confusion con las cuatro clases. 9

5.1. Division de la neurona en ventanas 13
5.2. Composición de la matriz X_w y el vector Y_w 13
5.3. LDA - Performance en funcion de la ventana. 14
5.4. LDA - Performance en funcion del tiempo. 15

6.1. LDA - Performance en funcion del tiempo - Promedio de las 500 corridas 17
6.2. Bayes Naive - Performance en funcion del tiempo - Promedio de las 500 corridas 18
6.3. SVM - Performance en funcion del tiempo - Promedio de las 500 corridas 19
6.4. Random Forest - Performance en funcion del tiempo - Promedio de las 500 corridas 20

7.1. Comparación no apareada para todas las neuronas. 22
7.2. Comparación no apareada para todas las neuronas con zoom. 23
7.3. Comparación no apareada para todas las neuronas en barras a partir del milisegundo 5000. 24
7.4. Porcentaje en el que la media de un método es mejor que la de otro a partir del milisegundo 5000. 24
7.5. Comparación apareada para todas las neuronas. 26
7.6. Comparación apareada para todas las neuronas con zoom. . . 27
7.7. Comparación apareada para todas las neuronas en barras a partir del milisegundo 5000. 27
7.8. Porcentaje en el que la media de un método es mejor que la de otro a partir del milisegundo 5000. 28

8.1. Relación Porcentaje - Cantidad de neuronas. 31
8.2. Gráficos performance/tiempo. 32
8.3. Gráficos performance/tiempo con zoom. 33
8.4. Gráficos de barras a partir del milisegundo 5000. 34
8.5. Gráficos de barras a partir del milisegundo 5000. 35

9.1. Random Forest - Gráfico de la performance en función al tamaño de ventana, desde el milisegundo 4000 - 500 corridas . . 38
9.2. Random Forest - Gráfico en barras de la performance en función al tamaño de ventana, desde el milisegundo 4000 - 500 corridas 38
9.3. Random Forest - Gráfico en barras de la performance en función al tamaño de ventana, desde el milisegundo 4000 - 500 corridas - primeras 20 ventanas 39
9.4. Random Forest - Gráfico en barras de la performance en función a la ventana, cada ventana posee $300ms$ - 500 corridas . 40
9.5. Random Forest - Gráfico en barras de la performance en función a la ventana, cada ventana posee $300ms$ - 500 corridas - primeras 20 ventanas 40
9.6. Random Forest - Ventana de tamaño fijo en $300ms$ (Verde) vs Ventana ampliándose a medida que avanza el tiempo (Rojo). 41

Capítulo 1

Introducción

Resumen: En este capítulo se brinda una introducción al trabajo realizado en esta Tesis

1.1. Introducción

En este trabajo se estudiará cuánta información es posible recopilar en base a los *spikes* neuronales provenientes de las neuronas del área tegmental ventral del cerebro de la rata.
Para ello, se utilizarán diferentes algoritmos estadísticos de *machine learning* con el objetivo de clasificar el set de *spikes* neuronales y evaluar si efectivamente existe información guardada en esta área del cerebro de la rata relativa a la actividad neuronal.

En el caso de tener éxito, esta información podría permitir predecir el comportamiento del animal conociendo solamente su actividad neuronal.

A su vez se pretende concluir cuales de los métodos estadísticos empleados permiten extraer mayor información, trabajando con *spikes* neuronales.

Como objetivo secundario se intentará descubrir cual es el menor número de neuronas que nos permiten clasificar los *spikes*.

Los datos utilizados para esta investigacion fueron cedidos por ****, en el marco de la investigación de ****.

Capítulo 2

Set de datos

Resumen: Descripción del set de datos.

2.1. Introducción

En esté capítulo se pretende explicar el origen del set de datos, así como su composición y estructura.
En el Apéndice A se realizará un análisis del mismo para poder entenderlo, evaluar sus fortalezas y sus debilidades.

2.2. Orígen de los Datos

El set de datos proporcionado corresponde al registro de los *spikes* neuronales correspondientes a cuatro ratas macho adulto Long Evans. Las mismas fueron sometidas a dos estímulos auditivos de frecuencias diferentes ($1KHz$ y $8KHz$), registrando la actividad neuronal del área tegmental ventral (VTA) del animal en el segundo posterior a la presentación del estímulo.
Esta actividad neuronal remanente es lo que habitualmente se considera memoria de corto plazo o memoria de trabajo según el tipo de paradigma conductual utilizado (Funahashi et al., October 1993).

Se realizaron 30 sesiones de registro, obteniéndose luego de un proceso de *spike* sorting 153 neuronas. (Quiroga et al., 2004)

Luego de someter al animal al estímulo, si el tono se trataba de la frecuencia asociada a GO debía sacar la lengua, si lo hacía correctamente recibiría una gota de agua, cabe aclarar que previamente eran privadas de agua por un determinado lapso de tiempo.
Si, por el contrario, el tono se trataba de la frecuencia asociada a NOGO el

animal debía esconder la lengua.
Con el objetivo de evitar sesgo con respecto al tono, para dos de los animales la frecuencia GO es $1KHz$ y la NOGO es $8KHz$. En contraste para los otros dos la frecuencia GO es $8KHz$ y la NOGO es $1KHz$.

Los trials de cada sesión se dividieron de acuerdo a la respuesta que el animal tuvo que realizar y de acuerdo a si esta respuesta fue o no correcta:

- **GOc**: trial GO correcto, el animal tuvo que sacar la lengua una vez (lick) y recibió agua
- **GOi**: trial GO incorrecto, el animal tuvo que sacar la lengua una vez (lick) y no lo hizo.
- **NOGOc**: trial NOGO correcto, el animal tuvo que esconder la lengua durante 1 segundo (no-lick) y así lo hizo.
- **NOGOi**: trial NOGO incorrecto, el animal tuvo que esconder la lengua durante 1 segundo (no-lick) y sin embargo la sacó.

Como cada tipo de trial tiene unívocamente asociado un estímulo auditivo determinado, la relación estímulo-respuesta es unívoca. Por este motivo hablar de trial GOc en el procesamiento de datos es equivalente a hablar de un estímulo auditivo determinado, presentado durante ese trial.

Para el análisis se tomó una ventana de $-4000\ ms$ a $+4001\ ms$, coincidiendo el $0\ ms$ con el inicio de la presentación del estímulo.

La información relativa al formato y distribución de los datos, junto con el análisis de los mismos se detalla en las secciones A.1 y A.2 del Apéndice A.

Capítulo 3

Métodos empleados

Resumen: En este capítulo se describirán los métodos de *machine learning* utilizados para el desarrollo de este trabajo.

3.1. Introducción

El objetivo de la utilización de métodos estadísticos de clasificación es poder discriminar los trials de tipo GOc de los NOGOc. Tanto los trials de tipo GOi como NOGOi fueron descartados por no contar con un número de observaciones relevante.

3.2. Métodos

3.2.1. Análisis Discriminante

El primer método utilizado fue el análisis discriminante, el cual es una generalización del metodo descripto por (Fisher, 1936). Este método consiste en una técnica simple que permite encontrar una función dependiente de las variables que intenta separar las clases. Asimismo, el mismo asume que las clases tendrán distribuciones Gaussianas separables entre

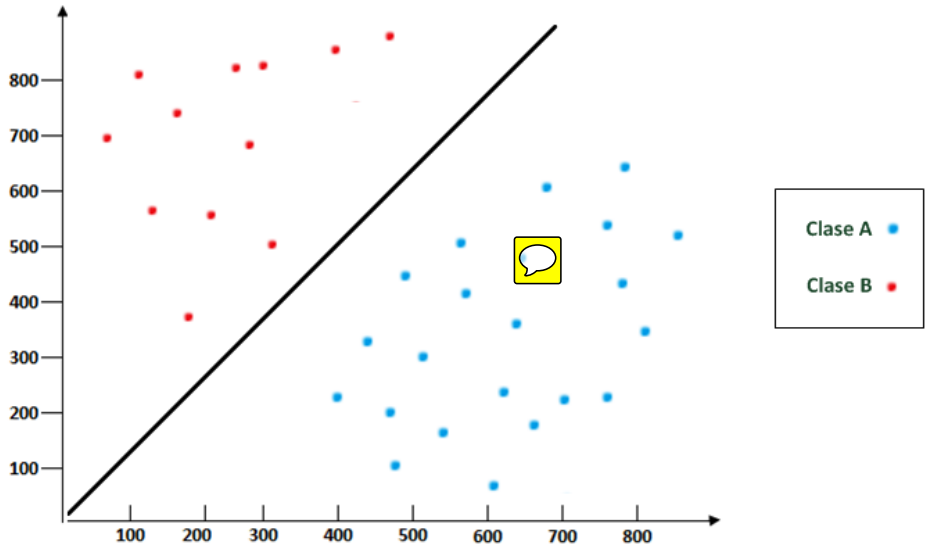


Figura 3.1: Separación lineal de 2 clases.

El método descripto por (Fisher, 1936) separa solamente dos clases. El LDA que se utiliza actualmente es una modificación del mismo que permite la separación de multiples clases. Este Multiclass LDA se enuncia en (Rao, 1948).

En este caso particular, se aplicó la version pseudolineal ya que la técnica del análisis discriminante requiere demasiado volumen de información para ajustar los modelos gaussianos con matrices de covarianza inversibles. En cambio, el modelo pseudolineal utiliza matrices de covarianza pseudoinversas, consiguiendo ajustar el modelo con menos volúmen de información.

3.2.2. Random Forest

El método de Random Forest utiliza la técnica divide and conquer. Es un ensamble de árboles de decisión entrenados con partes del corpus, de esta manera se logra, mediante un sistema de votos, asignar la clase correspondiente a cada una de las muestras. Cada árbol depende de los valores de un vector de muestras aleatorias, relevadas de forma independiente y con la misma distribución para todos los árboles.
Los arboles de decision son predictores sin sesgo, pero tienen alta varianza. Random Forest busca reducir la varianza promediando un alto número de arboles de decisión. El error de generalización convergirá a un límite cuanto mayor sea el número de árboles.
Este método se describe en (Breiman, 2001).

3.2.3. Support Vector Machine

El método Support Vector Machine en adelante llamado SVM se basa en la idea de proyectar los datos en un espacio dimensional mayor, con el fin de encontrar un hiperplano que permita separar las clases. Las características del hiperplano aseguran una alta capacidad de generalización de este método de machine learni
Una vez encontrado el hiperplano, se busca un márgen m que maximice el espacio entre las clases. Los vectores que se encuentran a distancia m del hiperplano son llamados *support vectors* (vectores de soporte).

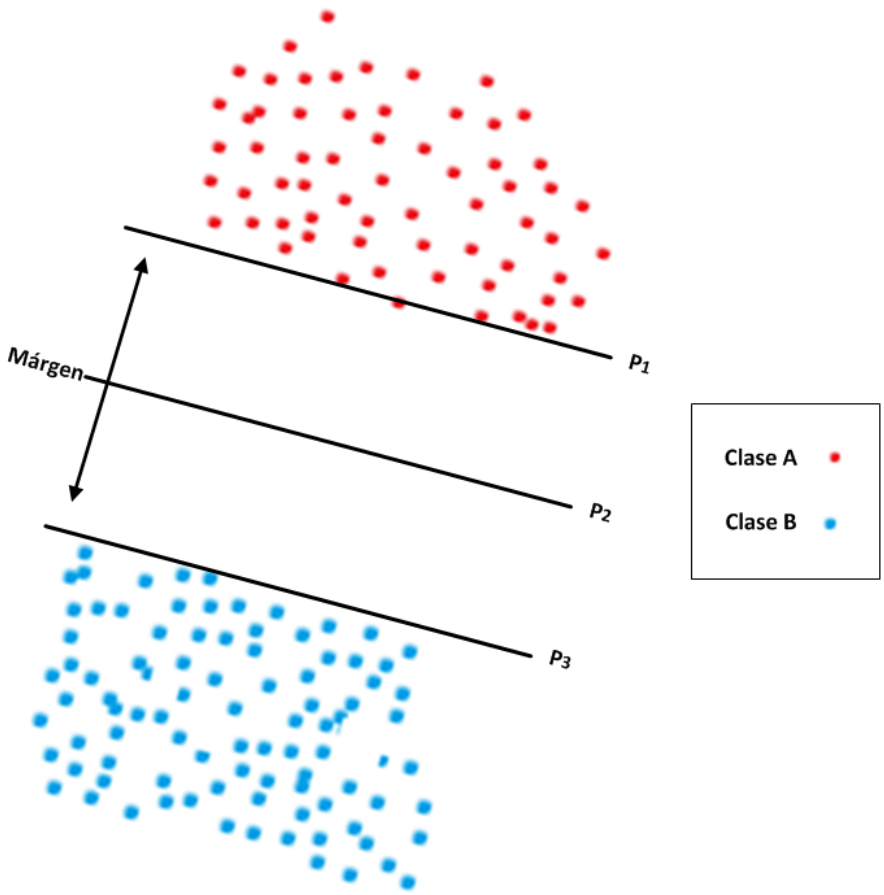


Figura 3.2: Separacion con margen. Los vectores p1 y p3 son los llamados *support vectors*

Dado que es necesario calcular productos internos entre vectores, es fácil inferir que en espacios de grandes dimensiones los mismos resultan compu-

tacionalmente costosos. Por ello, se utilizan funciones de kernel que simplifican el cálculo. Los kernels son funciones que permiten calcular el producto interno de dos vectores pertenecientes a una dimensión n en una dimensión m, con $m > n$, trabajando en la misma dimensión n.
Existen diferentes funciones de kernel y este es uno de los parámetros mas importantes a definir a la hora de utilizar SVM. Este método se describe en (Cortes y Vapnik, 1995).

3.2.4. Bayes Naive

Este clasificador toma como hipótesis que todas las variables son independientes. Utilizando alguna funcion de distribución a priori, Bayes Naive se basa en el teorema de Bayes para poder predecir la probabilidad a posteriori.
Este método es un clasificador simple basado en aplicar el teorema de Bayes con el agregado ingenuo (naive) de considerar a las variables independientes. Determinar su autor y la epoca de creación resulta complejo ya que dicho teorema data de 1763.
En (Kononenko, 1993) se describe al método como rápido, incremental y apto para el trabajo con atributos discretos y continuos. Sus ventajas son la excelente performance para resolver problemas de la vida real, con la posibilidad de explicar sus decisiones como la suma de la información obtenida. Sin embargo, su ingenuidad puede derivar en un bajo rendimiento cuando se traten dominios con fuertes dependencias entre atributos.

Capítulo 4

Interpretación del problema

Resumen: Se comienza a interpretar el problema y se ven los primeros resultados

4.1. Introducción

Existen muchos enfoques y caminos posibles a tomar para analizar el set de datos descripto en el capítulo 2.
En este capítulo se presentan algunos enfoques tomados y los resultados obtenidos.

4.2. Interpretaciones

La técnica utilizada para entrenar los algorítmos estadísticos elegidos es el aprendizaje supervisado, detallado en (Mohri et al., 2012).
Esta técnica consiste en predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto una serie de ejemplos (datos de entrenamiento).
Para el aprendizaje supervisado se tomaron en forma aleatoria el 70 % de los datos como datos de entrenamiento y el 30 % como datos de testeo.
Elegido el método de aprendizaje, queda por definir con que información se van a entrenar los algoritmos.

4.2.1. Analisis mono-neuron

Como primera aproximación se trató de conseguir separar las clases con una sola neurona.

Usando solamente una neurona, las opciones para entrenar el clasificador son:

- Tomar cada milisegundo como una variable, y enseñarle al clasificador todos los trials como X y las respuestas (GOc, GOi, NOGOc, NOGOi) como Y.
- Tomar la suma de los milisegundos como X (teniendo una sola variable) y las respuestas como Y.
- Hacer una mezcla de los dos ítems anteriores y tomar la suma de N milisegundos como una variable, teniendo asi 8001 ms / N variables como X y las respuestas como Y.
- Buscar patrones dentro de los 8001ms y alimentar al clasificador con los datos de alguno de los items anteriores mas los patrones encontrados. Por ejemplo, se buscaron picos de *spikes* en una ventana de tiempo.

El primer problema que se observó es que los trials de GOi y NOGOi son tan pocos que los clasificadores no encontraban patrón para poder diferenciarlos de los demás.

Supervised Learning LDA

Classifier performances

Error rate			0,3338					
Values prediction			Confusion matrix					
Value	Recall	1-Precision		GOc	NOGOi	NOGOc	GOi	Sum
GOc	0,6185	0,2595	GOc	214	0	131	1	346
NOGOi	0,0000	0,0000	NOGOi	9	0	19	0	28
NOGOc	0,7855	0,3765	NOGOc	62	0	260	9	331
GOi	0,4500	0,5263	GOi	4	0	7	9	20
			Sum	289	0	417	19	725

Figura 4.1: Matriz de confusion con las cuatro clases.

Se decidió de aquí en adelante utilizar solamente los trials de GOc y NOGOc para simplificar el trabajo de los clasificadores.

La performance de clasificacion con LDA no superó el 70 % en ninguna de las pruebas. Por ende se concluyó que con una sola neurona no hay suficiente información para diferenciar un trial GOc de uno NOGOc.

4.2.2. Análisis multi-neuronal

4.2.2.1. Análisis

Al analizar varias neuronas, debemos al igual que en el caso anterior, definir las variables de entrada. Queda en evidencia que las respuestas de los trials son las salidas esperadas.
Aprovechando el análisis mononeuronal realizado en la sección anterior, se puede elaborar un análisis similar para el caso multi-neuronal teniendo en cuenta que debemos definir una entrada compuesta de variables para cada respuesta. Por lo tanto, una de las opciones es tomar como variables a las neuronas y como valor la suma de los *spikes* para cada uno de los trials.
Otra opción también factible es seguir usando cualquiera de los items mencionados en el análisis mononeuronal alimentando con más información (mas cantidad de neuronas =>mas cantidad de trials) a los clasificadores.

Se decidió entonces, optar por la primer opción, generando una matriz X y un vector Y.

4.2.2.2. Preparación

La matriz X enseñada al clasificador tendrá a las neuronas como columnas y los distintos trials de GOc y NOGOc como filas. El vector Y tendrá las respuestas esperadas.

Para conseguir dicha matriz, se busco el máximo tamaño de trials GOc y NOGOc que aparecía mínimamente en todas las neuronas. Se encontraron 31 trials GOc y 23 trials NOGOc disponibles como mínimo en todas las neuronas. Se tomó entonces el mínimo valor de los dos (23) para conseguir que la matriz tenga la misma cantidad de trials de uno y de otro.

Se tiene la matriz X con 153 columnas (cantidad total de neuronas VTA que se registraron en las 30 sesiones) y 46 filas que representan 23 trials GOc y 23 trials NOGOc. También se cuenta con el vector Y con 46 posiciones donde en cada posición se colocará un 1 o un 2, dependiendo si la fila de X correspondiente a este número de posición corresponde a la clase 1 (GOc) o a la clase 2 (NOGOc).
A su vez, se tomará el 70 % de la matriz X para entrenamiento y el 30 % restante para medir la performance de clasificación.

4.2.2.3. Resultados

Se entrenó un LDA con las matrices mencionadas anteriormente y se consiguió una performance alrededor del 90 %.
Estos resultados resultan satisfactorios, mas adelante se analizarán las implicancias de los mismos.

Capítulo 5

Enfoque por ventanas

Resumen: Se describe en este capítulo el análisis realizado para obtener información relativa a los diferentes instantes de tiempo.

5.1. Introducción

Luego de aplicar el análisis multi-neuronal con resultados satisfactorios, se deduce que la información respecto al tono se encuentra guardada en un grupo de neuronas. Por lo tanto los clasificadores mejorarán su performance cuando la entrada se componga de varias neuronas.
Dado que el tono es enseñado al animal a los $4000ms$, se pretende identificar cuanta información acerca del tono se encuentra almacenada a medida que transcurre el tiempo.

5.2. Implementación

Se analizarán pequeños períodos de tiempo con el fin de observar como evoluciona la performance de clasificación a lo largo del tiempo. Se recorrió la matriz de cada neurona de a 10 en 10 ms tomando ventanas de 300 ms y realizando la suma de los spikes en dichas ventanas, con lo cual se pasó de trabajar con una matriz de 8001 columnas llenas de 0 y 1 a trabajar con una matriz de 771 columnas llenas de números enteros.

Se probaron diferentes tamaños de ventana y estos impactaron significativamente en la performance. En capítulos posteriores se analizarán las ventajas y desventajas de los distintos tamaños de ventana. Por el momento, y para continuar con el análisis, se dejará el tamaño de ventana fijo en 300 ms.
Este tamaño permitirá ver como la performance crece en función del tiempo.

NEURONA A

Milisegundos

Trial/Milisegundos	1	2	...	x	x+1	...	...	x+Wsize-1	x+Wsize	...	n-2	n-1	n	Respuesta
Trial 1	1	1	...	1	0	...	...	0	1	...	1	0	0	NOGOc
Trial 2	0	1	...	1	1	...	...	0	1	...	0	1	1	GOc
Trial 3	1	0	...	1	0	...	...	0	1	...	0	1	0	NOGOc
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
Trial m	0	1	...	0	1	...	...	1	0	...	0	0	1	GOc

Wi

Figura 5.1: Division de la neurona en ventanas

En forma análoga a lo explicado en el capítulo anterior, se utilizarán las neuronas como variables y la suma de los spikes en la ventana como valor, con lo cual se analizará individualmente la clasificación en cada ventana.

A este par de X e Y se lo denominará X_w e Y_w ya que en este caso hay un par por cada ventana (window).

NEURONA A

Trial/Ventana	w1	w2	...	wi	...	wn	Respuesta
Trial 1	3	6	...	3	...	5	NOGOc
Trial 2	4	10	...	2	...	6	GOc
Trial 3	15	0	...	10	...	10	NOGOc
...	...	...	...	...	...	...	...
Trial m	8	3	...	11	...	4	GOc

ΣWi

NEURONA B

Trial/Ventana	w1	w2	...	wi	...	wn	Respuesta
Trial 1	7	3	...	4	...	9	GOc
Trial 2	9	16	...	3	...	3	GOc
Trial 3	12	4	...	5	...	17	NOGOc
...	...	...	...	...	...	...	...
Trial m	6	0	...	1	...	5	NOGOc

ΣWi

MATRIZ X (Wi) **VECTOR Y (Wi)**

Trial/Neurona	Neurona A	Neurona B	...	Neurona N	Respuesta
Trial 1	2	4	...	6	GOc
Trial 2	11	3	...	8	GOc
Trial 3	3	5	...	3	NOGOc
...	...	...	...	...	...
Trial m	10	1	...	5	NOGOc

Figura 5.2: Composición de la matriz X_w y el vector Y_w

Nuevamente, se tomará el 70 % de la matriz X para entrenamiento y el 30 % restante para medir la performance de clasificación.

5.3. Resultados

Se corrió entonces el algoritmo LDA para cada una de las ventanas y se graficó la performance en función de la ventana.

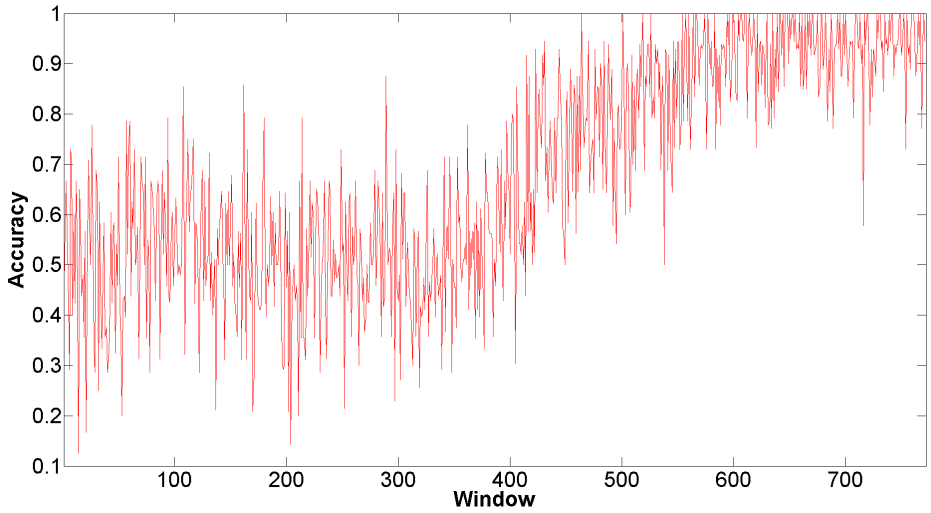


Figura 5.3: LDA - Performance en funcion de la ventana.

Para poder analizar mejor los resultados se graficó nuevamente la performance pero esta vez en función del tiempo (ms), realizando una transformación del espacio de ventanas al espacio de tiempo. Esto es posible debido que hay una relación directa entre las ventanas y el tiempo.
La ecuación utilizada para aproximar a que ventana corresponde la performance es $t = w * 10 + 291$. De esta manera para el milisegundo 8001 corresponderá la ventana 771. $(8001 = 771 * 10 + 291)$.

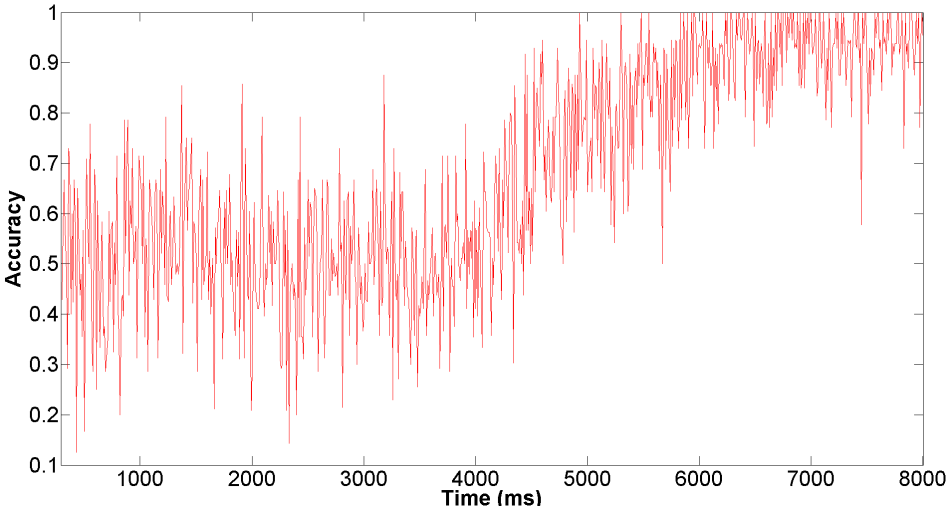


Figura 5.4: LDA - Performance en funcion del tiempo.

En la figura 5.4 se puede apreciar claramente como a partir del milisegundo 4000 la performance comienza a incrementarse en media.

Estos resultados se encuentran dentro de lo esperado debido a que el estímulo auditivo le es enseñado a la rata en el milisegundo 4000, por ende, la información del tono debe encontrarse en un momento posterior a ese instante. Una vez tomada la decisión, es consistente que la información persista en el resto de las ventanas.
Se puede observar también que la performance fluctua demasiado con respecto a la ventana tomada. Se debe analizar si es un problema de esa ventana o simplemente es debido a que los trials fueron tomados al azar. En el siguiente capítulo se analizará esta cuestión en detalle.

Capítulo 6

Iteraciones y resultados

Resumen: En vista de los resultados obtenidos, y con el objetivo de validar que el porcentaje hallado anteriormente no haya sido casualidad, se corrieron 500 iteraciones para cada ventana.

6.1. Introducción

Para cada ventana se creará la matriz X y el vector Y y luego se correrá el algoritmo de clasificación 500 veces generando X_{RUN}, Y_{RUN}, X_{TEST} e Y_{TEST} tomando los datos de forma aleatoria, con el objetivo de promediar y calcular el desvío estandar de la performace de clasificación.

6.2. Análisis con LDA

De las 500 iteraciones realizadas, se graficó el promedio y el desvío estándar de la performance de clasificación en función del tiempo.

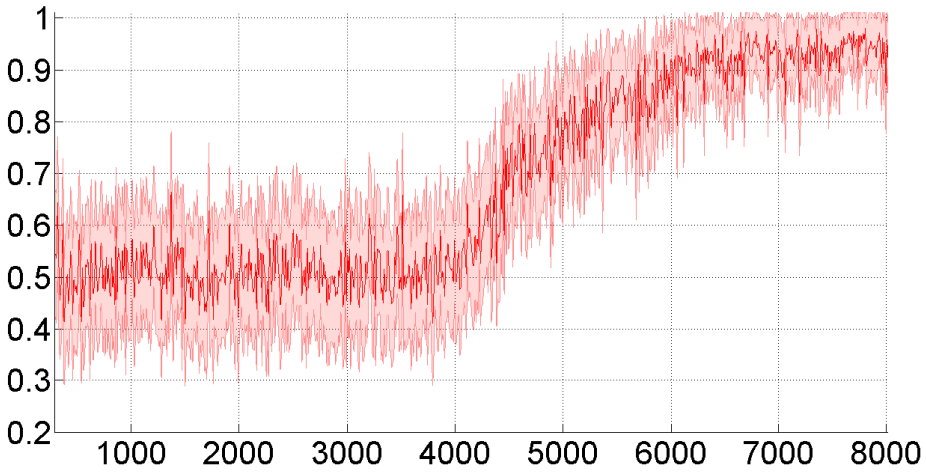


Figura 6.1: LDA - Performance en funcion del tiempo - Promedio de las 500 corridas

Dado que tomamos 500 iteraciones y que la perfomance con su desvío estandar a partir de cierto instante nunca disminuye de un 70 %, podemos afirmar con toda seguridad que la información del tono se encuentra almacenada en las neuronas.
Se observa ademas que, desestimando algunas ventanas donde el desvio estandar cae, a partir del milisegundo 6000 la performance con su desvio, se mantiene por encima del 80 %.

A continuación, se correrán el resto de los algoritmos de clasificación para intentar mejorar la performance obtenida.

6.3. Análisis con Bayes-Naive

Se entrenó un calsificador Bayes-Naive con una distribución de datos multinomial, dado que esta distribución suele tener buenos resultados para características discretas.
La probabilidad a priori de las clases se configuro como "empiricas", es decir que la misma será igual a la frecuencia relativa de distribucion de cada clase.
Al igual que con LDA y los demás algoritmos, se corrieron 500 iteraciones para luego tomar el promedio de la precisión y su desvío estandar. Se muestran a continuación los resultados graficados en función del tiempo.

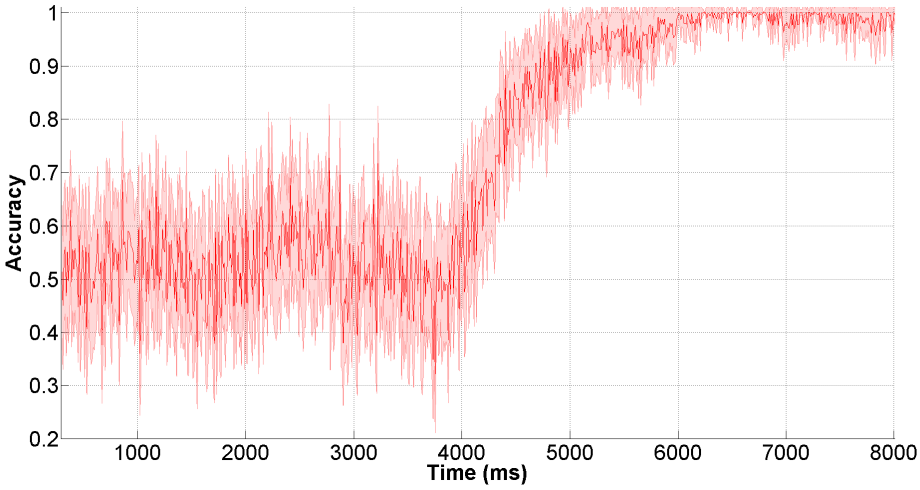


Figura 6.2: Bayes Naive - Performance en funcion del tiempo - Promedio de las 500 corridas

Se puede apreciar como a partir del milisegundo 5000 la performance con su desvio estandar nunca decae del 80 %. A su vez, a partir del milisegundo 6000 se mantiene sobre el 90 %.

6.4. Análisis con SVM

Se entrenó un clasificador SVM con una funcion de kernel lineal.

Se presentan, a continuación, los resultados de la performance de clasificación en función del tiempo.

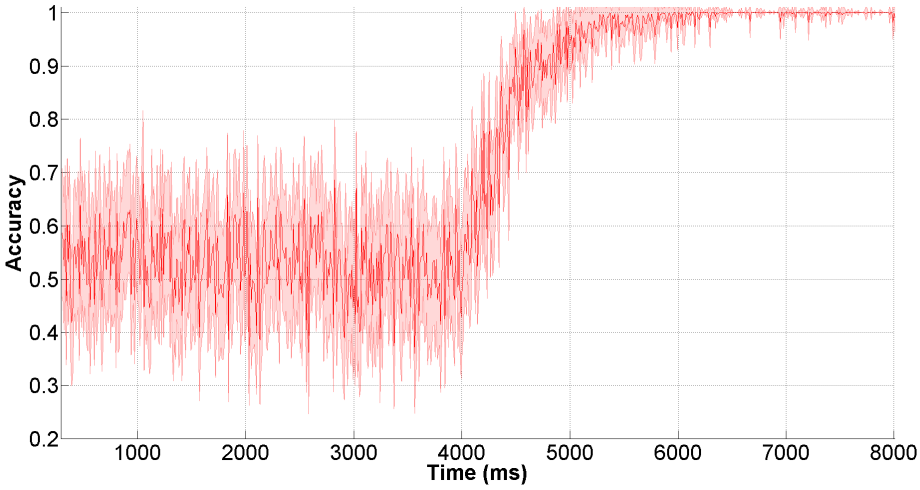


Figura 6.3: SVM - Performance en funcion del tiempo - Promedio de las 500 corridas

En este caso a partir del milisegundo 5500 la performance con su desvio estandar se mantiene sobre el 90 %, y a medida que el tiempo transcurre se obtienen valores casi perfectos.

6.5. Análisis con Random Forest

Se entrenó un clasificador random forest con 300 árboles. El mtry que indica el número máximo de variables en cada arbol se configuró en 12. El mismo se calculó como el piso de la raíz cuadrada de la cantidad de neuronas. Se aconseja calcular el mtry de esta forma por primera vez, y luego si fuera necesario ir variando el valor hasta obtener la mejor performance.
Se muestran, entonces, los resultados obtenidos hallando el promedio y el desvío estandar de la performance en las 500 iteraciones.

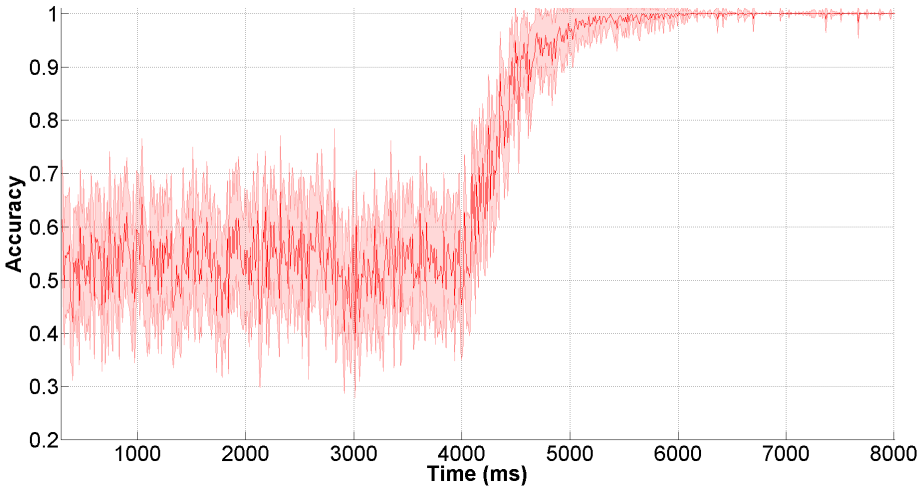


Figura 6.4: Random Forest - Performance en funcion del tiempo - Promedio de las 500 corridas

En la figura 6.4 se observa que este metodo converge rapidamente a 100 % a partir del milisegundo 4000. Ya desde el 5000 la performance se mantiene por encima del 90 % y se incrementa llegando al 100 % y manteniendose estable durante casi todo el resto del tiempo.

A simple vista parece ser Random Forest el metodo que mejor clasifica, seguido por SVM y Bayes Naive en ese order. En el capitulo siguiente se realizarán las comparaciones correspondientes entre los algoritmos, con el objetivo de encontrar el mejor clasificador para *spikes* neuronales.

Capítulo 7

Comparaciones y corridas apareadas

Resumen: En este capítulo se compararán los distintos resultados y se realizarán corridas utilizando exactamente las mismas entradas para los diferentes algoritmos con el objetivo de lograr una comparación estadística válida.

7.1. Introducción

Existen muchas formas de comparar los resultados obtenidos anteriormente, en este capítulo se mostrarán algunas de ellas.

7.2. Comparación entre los métodos

Se incluyen en el siguiente gráfico los cuatro métodos corridos anteriormente en función del tiempo.

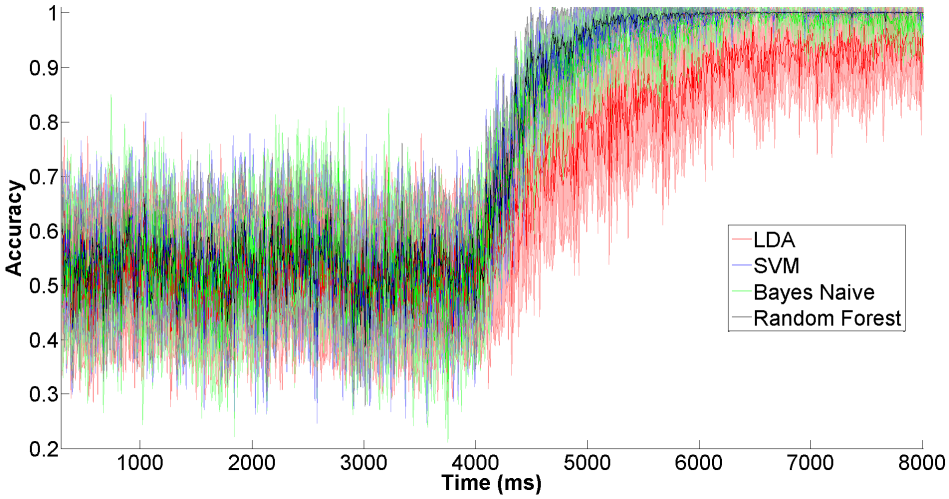


Figura 7.1: Comparación no apareada para todas las neuronas.

Como se había detallado en el capítulo anterior, se observa una gran diferencia entre LDA y cualquiera de los otros tres métodos. Al querer comparar Bayes-Naive, Random Forest y SVM existe la dificutad que cada uno tiene mejor perfomance en diferentes zonas. Dependiendo de que ventana elija, será mas eficiente uno u otro método.

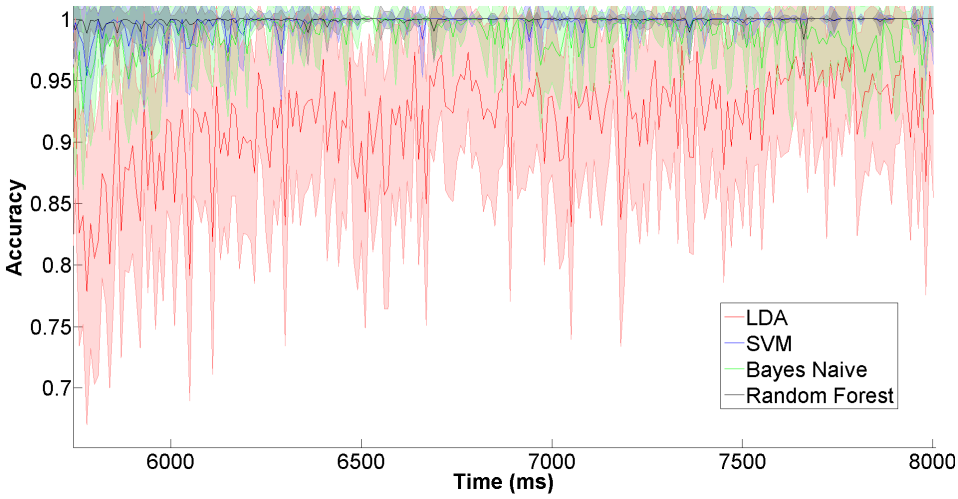


Figura 7.2: Comparación no apareada para todas las neuronas con zoom.

La figura 7.3 permite observar lo que ocurre en el último período de tiempo. La performance de Bayes Naive parece ligeramente inferior a la de Random Forest y SVM.

Debido a que el tono es mostrado a partir del milisegundo 4000, no tiene sentido comparar los métodos antes de este instante, por ende, para poder apreciarlos mejor se calculó la media del promedio de la precisión para todas las ventanas posteriores al milisegundo 5000 (cuando los metodos empiezan a establizarse). Se presenta el correspondiente gráfico debajo, junto a su desvío estandar.

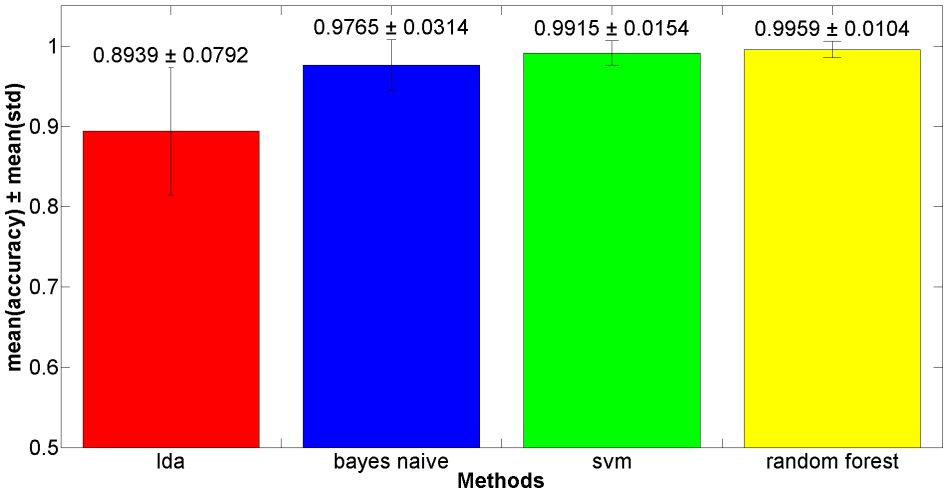


Figura 7.3: Comparación no apareada para todas las neuronas en barras a partir del milisegundo 5000.

En la figura 7.4 se aprecia una comparación ventana a ventana de la media de cada método.

Random Forest mayor que LDA	100,00%
Random Forest mayor que Bayes Naive	87,70%
Random Forest mayor que SVM	54,15%
SVM mayor que LDA	100,00%
SVM mayor que Bayes Naive	80,40%
Bayes Naive mayor que LDA	99,67%

Figura 7.4: Porcentaje en el que la media de un método es mejor que la de otro a partir del milisegundo 5000.

Queda entonces demostrado que LDA es inferior a los otros tres metodos cuando se trata de clasificar *spikes* neuronales. En la seccion 7.3.2 se profundizará sobre estos resultados.

A pesar que Random Forest pareciera tener mayor precisión, los desvíos estandar de Random Forest, Bayes-Naive y SVM se solapan. Dada dicha situación, no es posible inferir cual de estos tres métodos es preferible. Se procede, en la siguiente sección, a realizar corridas de los métodos con exac-

tamente las mismas entradas, dado que en ciertas ventanas la performance de los métodos podría verse afectada por las neuronas y los trials elegidos aleatoriamente.

7.3. Resultados apareados

7.3.1. Comparación

Se entrenaron, entonces, los cuatro algoritmos estadísticos con las mismas entradas realizando 500 iteraciones para obtener el promedio y el desvío estandar de la precisión en los cuatro métodos. De esta manera se podrá apreciar si existe un clasificador que funcione mejor en ciertas ventanas o instantes de tiempo.

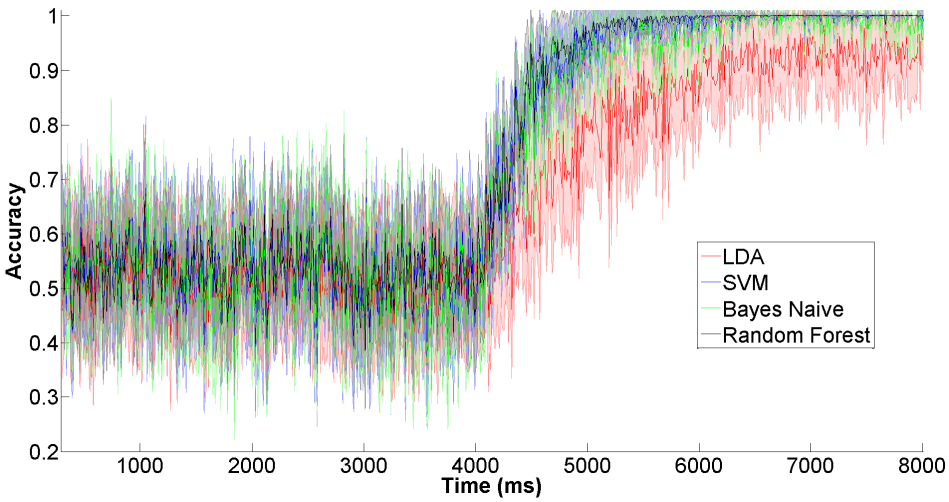


Figura 7.5: Comparación apareada para todas las neuronas.

La figura 7.5 permite observar los métodos a lo largo de los 8001 ms.

Para poder observar cuales métodos poseen mejor performance se muestra la figura 7.6 donde se puede apreciar como Random Forest y SVM estan por encima de Bayes Naive y LDA durante la mayoría de las ventanas.

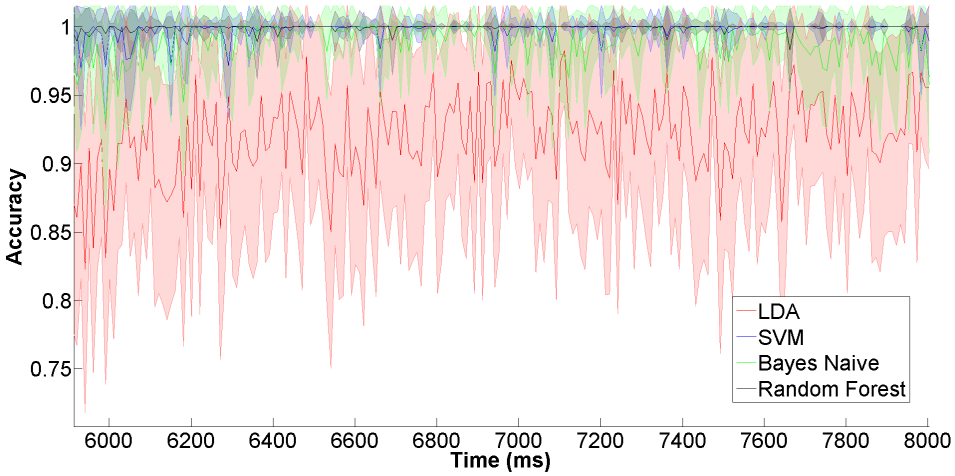


Figura 7.6: Comparación apareada para todas las neuronas con zoom.

En el grafico 7.7 se aprecia lo expresado anteriormente. Se puede ver mas claramente la diferencia entre los métodos a partir del milisegundo 5000

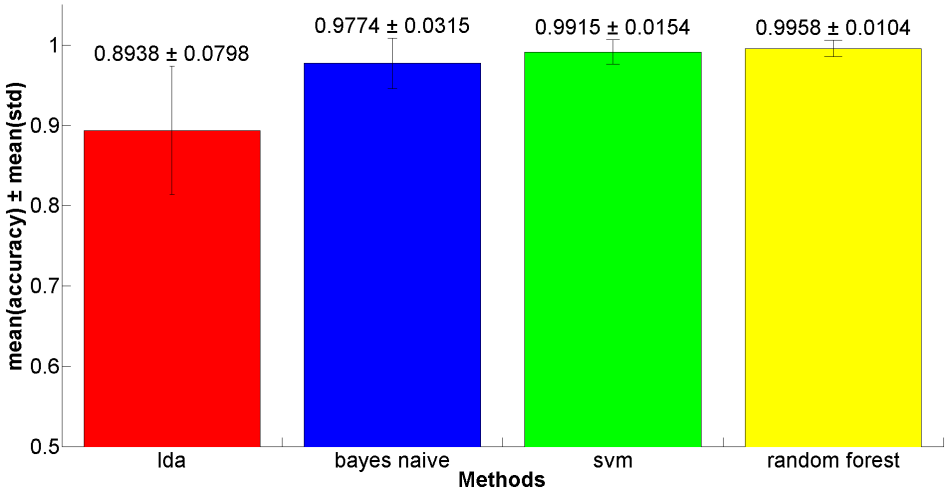


Figura 7.7: Comparación apareada para todas las neuronas en barras a partir del milisegundo 5000.

Dado que se quiere poder conocer cual método es que mejor performance tiene, se compararon todos los métodos entre si, ventana a ventana.

Random Forest mayor que LDA	100,00%
Random Forest mayor que Bayes Naive	87,71%
Random Forest mayor que SVM	53,16%
SVM mayor que LDA	100,00%
SVM mayor que Bayes Naive	85,05%
Bayes Naive mayor que LDA	99,34%

Figura 7.8: Porcentaje en el que la media de un método es mejor que la de otro a partir del milisegundo 5000.

7.3.2. Conclusión

En las figuras 7.4 y 7.8 se muestra el porcentaje de ventanas en las que un método es superior a otro.
Se observa que tanto Random Forest como SVM son superiores a LDA en media siempre. A su vez, Bayes Naive es superior a LDA en media con un nivel de significación de $0,01$.
Se concluye entonces que LDA es inferior a los tres métodos anteriores para el análisis de *spikes* neuronales en roedores.

En contraste Random Forest y SVM son superiores en media a Bayes Naive con un nivel de significación de $0,13$ y $0,20$ respectivamente para corridas no apareadas y de $0,13$ y $0,15$ respectivamente para corridas apareadas.
Si bien estos niveles de significación no son concluyentes para determinar la superioridad de un método sobre otro, son valores interesantes que pueden ayudar a decidir que clasificador utilizar a la hora de clasificar *spikes*.

En conclusión se recomienda utilizar Random Forest o SVM a la hora de analizar *spikes* neuronales registrados en el área tegmental ventral del cerebro de una rata.

Capítulo 8

Conjunto reducido

Resumen: En este capítulo se reducirá la cantidad de neuronas con las que se entrenan los métodos para poder evaluar cuantas neuronas se necesitan como mínimo para clasificar los trials.

8.1. Introducción

Debido a la eficacia de los métodos estudiados en los capítulos anteriores se hará foco en este capítulo en la clasificación de los trials reduciendo gradualmente el número de neuronas utilizadas para extraer la información.

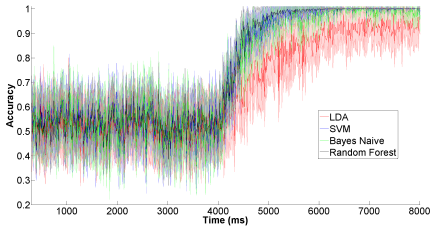
8.2. Análisis comparativo

Partiendo de la matriz X y el vector Y, generados con anterioridad para cada ventana, se tomó un porcentaje de neuronas para correr nuevamente todos los algoritmos. Los porcentajes elegidos fueron 100 %, 75 %, 50 %, 25 %, 10 % y 5 % del total de neuronas.

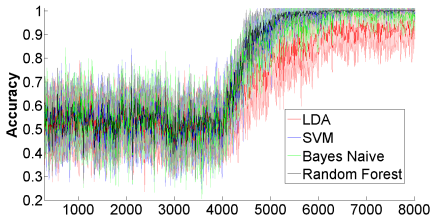
Porcentaje	Neuronas
100%	153
75%	115
50%	77
25%	39
10%	16
5%	8

Figura 8.1: Relación Porcentaje - Cantidad de neuronas.

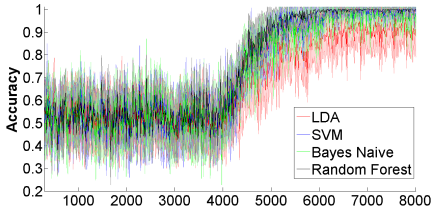
Una vez fijado dicho porcentaje, la selección de las neuronas fue realizada en forma aleatoria.
Se muestran a continuación los resultados obtenidos.



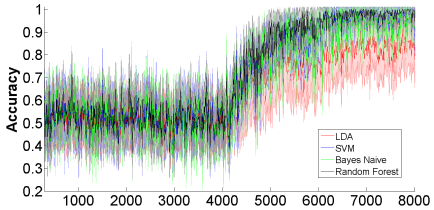
(a) Comparación apareada para 153 neuronas.



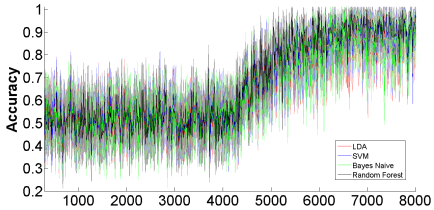
(b) Comparación apareada para 115 neuronas.



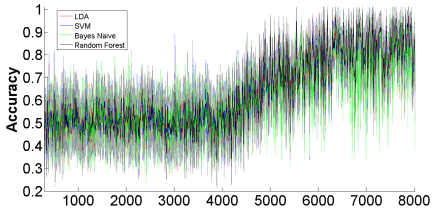
(c) Comparación apareada para 77 neuronas.



(d) Comparación apareada para 39 neuronas.



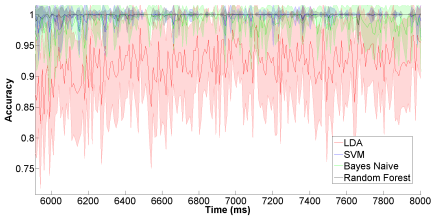
(e) Comparación apareada para 16 neuronas.



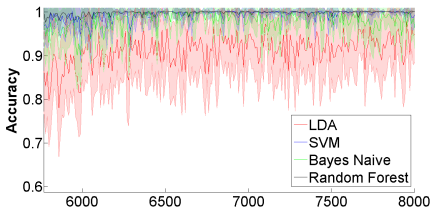
(f) Comparación apareada para 8 neuronas.

Figura 8.2: Gráficos performance/tiempo.

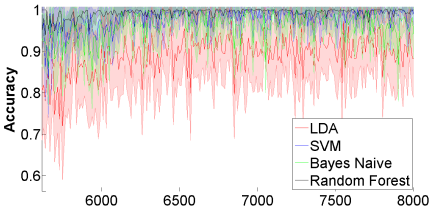
En la figura 8.2 se ve como cuando las neuronas disminuyen, disminuye la performance de los metodos.



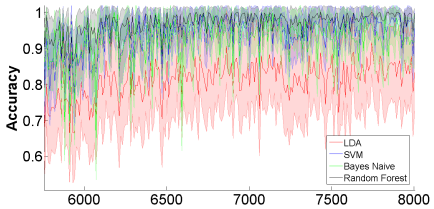
(a) Comparación apareada para 153 neuronas.



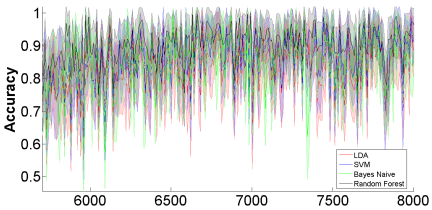
(b) Comparación apareada para 115 neuronas.



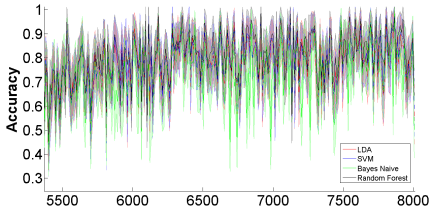
(c) Comparación apareada para 77 neuronas.



(d) Comparación apareada para 39 neuronas.



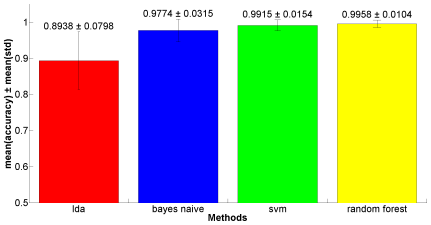
(e) Comparación apareada para 16 neuronas.



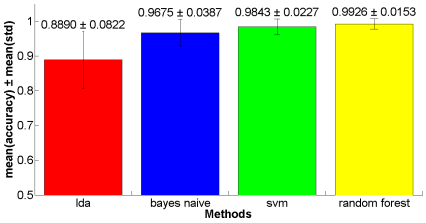
(f) Comparación apareada para 8 neuronas.

Figura 8.3: Gráficos performance/tiempo con zoom.

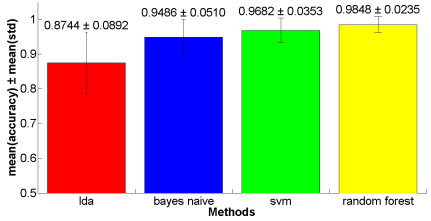
Al realizar zoom en los últimos $2500ms$ observamos como para las cuatro primeras figuras LDA tiene una performance diferenciable de los otros metodos, en cambio para las últimas dos, todos los metodos se superponen bajando a una performance entre 0,6 y 0,9.



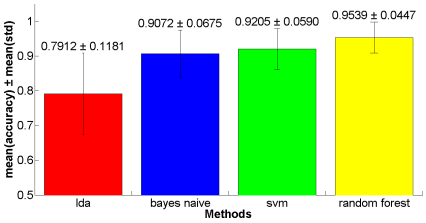
(a) Comparación apareada para 153 neuronas.



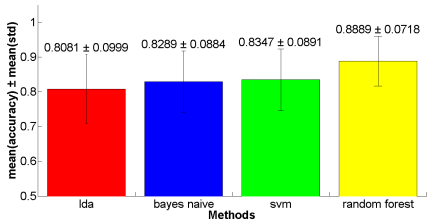
(b) Comparación apareada para 115 neuronas.



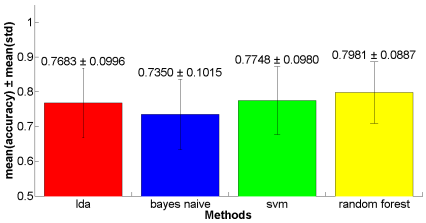
(c) Comparación apareada para 77 neuronas.



(d) Comparación apareada para 39 neuronas.



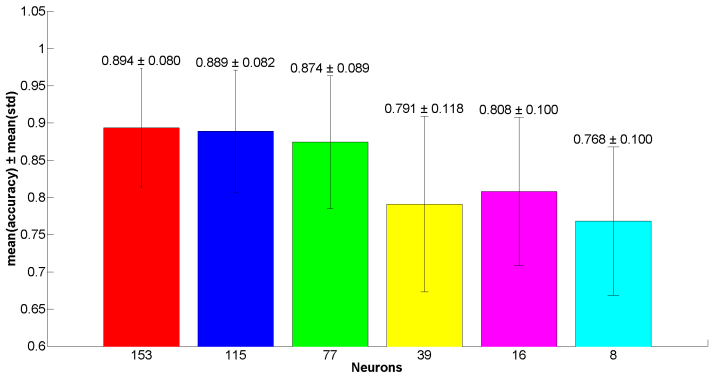
(e) Comparación apareada para 16 neuronas.



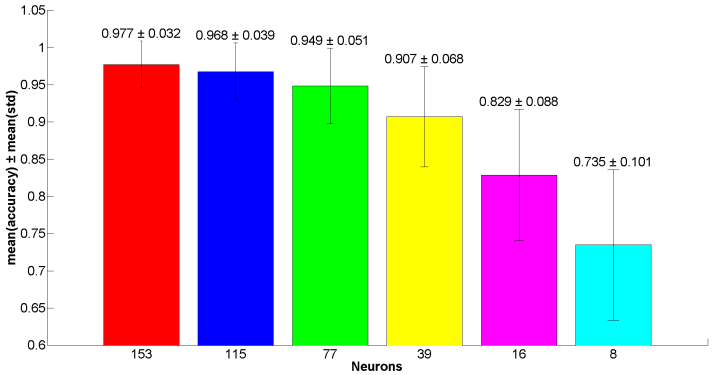
(f) Comparación apareada para 8 neuronas.

Figura 8.4: Gráficos de barras a partir del milisegundo 5000.

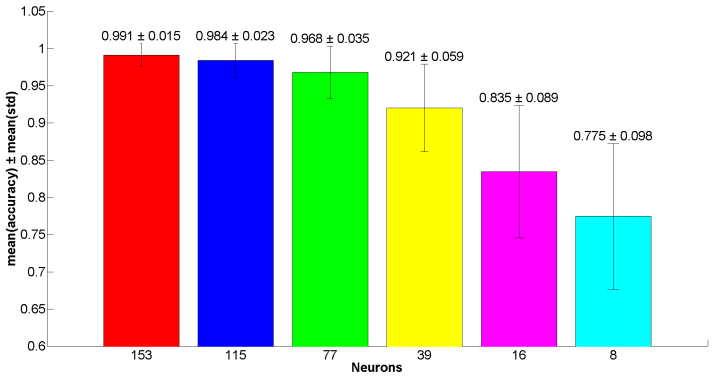
Como se mencionó arriba para la figura 8.4f se aprecia como los metodos se emparejan y tienen *performance* parecida.



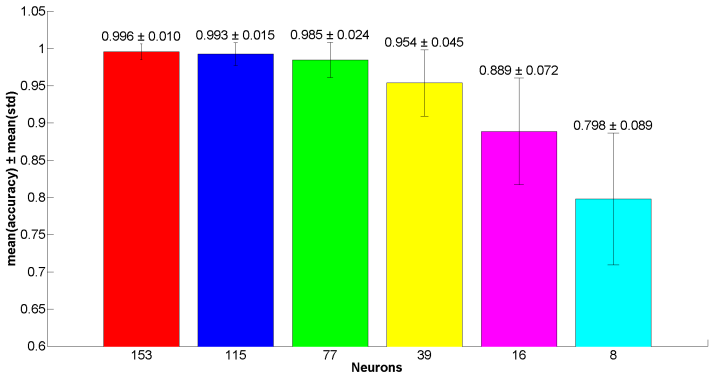
(a) Performance LDA.



(b) Performance Bayes Naive.



(c) Performance SVM.



(d) Performance Random Forest.

Figura 8.5: Gráficos de barras a partir del milisegundo 5000.

Por último en la figura 8.5 se observa que independientemente del método, cuando la cantidad de neuronas es mayor o igual a 77 la performance se mantiene casi constante independientemente de cuantas neuronas mas agreguemos. Por el contrario, parece empezar a decrecer linealmente hacia abajo a medida que se disminuye el número de neuronas.

8.3. Conclusión

Como conclusión, se percibe que los métodos requieren una cantidad significativa de neuronas para clasificar con una buena performance.
Igualmente, se ve que teniendo solamente la información de 8 neuronas, podemos clasificar los trials de GOc vs NOGOc usando Random Forest con una performance de $0{,}798 \pm 0{,}089$ Lo cual puede ser aceptable dependiendo del problema q se quiera resolver.

Capítulo 9

Tamaño de ventana

Resumen: En los capítulos anteriores se fijó la variable tamaño de ventana en 300 ms. En este capítulo se estudiará el impacto de variar dicho tamaño.

9.1. Introducción

Se desea estudiar que impacto tiene variar el tamaño de ventana y como afecta en la clasificación.
A su vez, se quiere encontrar con qué velocidad de respuesta se podría diferenciar un tono GOc de uno NOGOc.
En este capítulo se observará la variación de la performance en función al tamaño y como esto es útil para indicar cuantos milisegundos se deben esperar para poder obtener una clasificación.

9.2. Exploración de tamaños

Dado que el tono ocurre en el milisegundo 4000, se ampliará paulatinamente el tamaño de ventana desde ese instante y se buscará cuál es el mínimo tamaño que resulta en una performance aceptable.

Se entrenó a *Random Forest*, por ser el método que mejores resultados tuvo y porque además es un método robusto al ruido (Breiman, 2001), lo que permitirá que los primeros milisegundos, no afecten demasiado la clasificación de las ventanas a medida que se amplian en el tiempo.

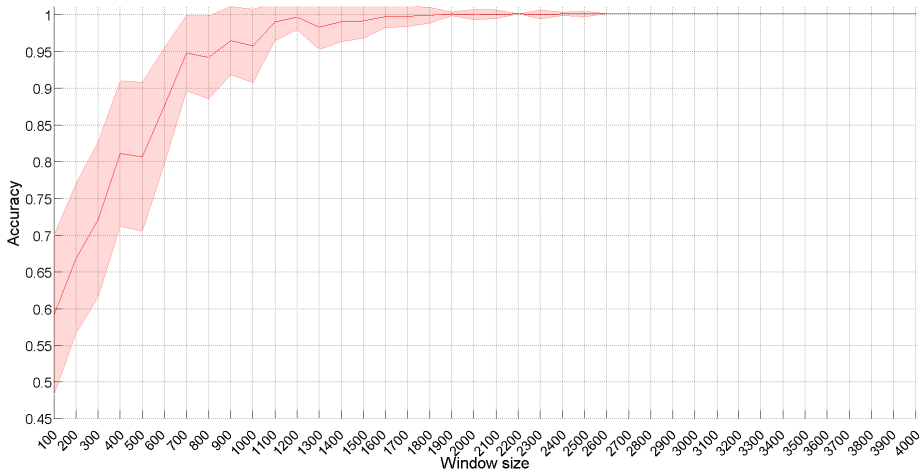


Figura 9.1: Random Forest - Gráfico de la performance en función al tamaño de ventana, desde el milisegundo 4000 - 500 corridas

La figura 9.1 permite observar el crecimiento a medida que se aumenta el tamaño de ventana a lo largo del tiempo, del milisegundo 4000 en adelante. Para apreciar mejor los resultados se realizará un grafico de barras mostrando el rendimiento en cada ventana, aumentando de a $100ms$ la amplitud de las mismas.

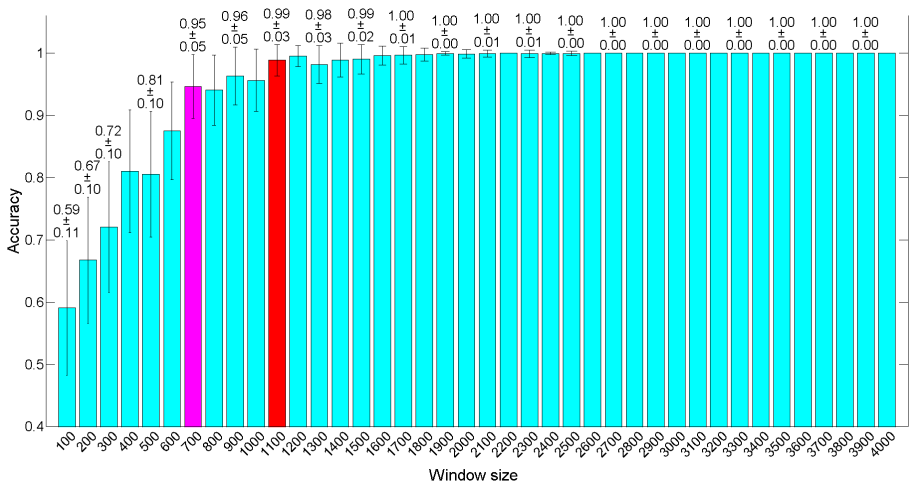


Figura 9.2: Random Forest - Gráfico en barras de la performance en función al tamaño de ventana, desde el milisegundo 4000 - 500 corridas

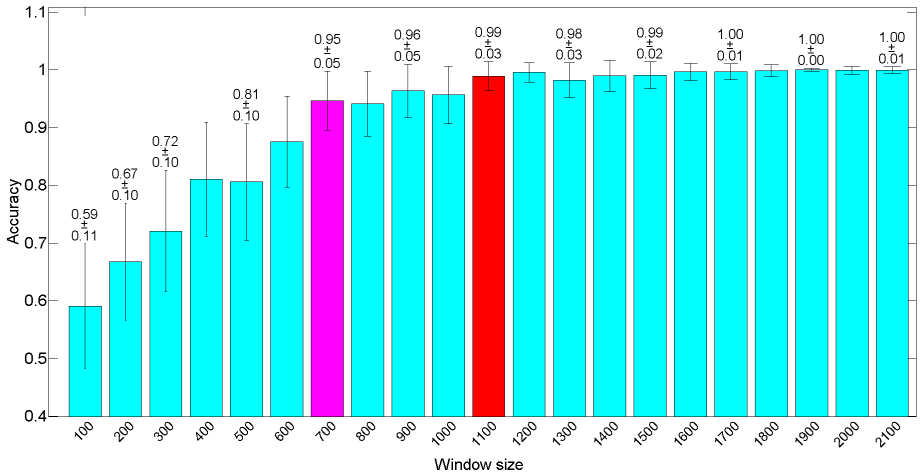


Figura 9.3: Random Forest - Gráfico en barras de la performance en función al tamaño de ventana, desde el milisegundo 4000 - 500 corridas - primeras 20 ventanas

En la figura 9.2 se aprecia como a partir de la amplitud $1800ms$ la clasificación es perfecta. La figura 9.3 muestra una ventana pintada de magenta, con $700ms$ de amplitud. A partir de esta ventana la performance se mantiene por encima de $0{,}94 \pm 0{,}05$.
La ventana pintada de rojo ($1100ms$ de amplitud) indica que la performance se mantendrá por encima de $0{,}98 \pm 0{,}03$ para todo el resto de las amplitudes.

Conocer el menor tamaño de ventana que se necesita para obtener cierta performance es importante debido a que un tamaño de ventana mas chico implica una respuesta más rápida ante el problema de decidir posteriormente a haberle mostrado el tono a la rata si la misma sacará o no la lengua.

Se pretende entonces saber si la performance obtenida es debido a utilizar un tamaño de ventana determinado o si simplemente se debe al avance en el tiempo de las mismas. En otras palabras, se desea averiguar si se pueden obtener resultados similares con un tamaño de ventana mas pequeño que posea los últimos milisegundos de una ventana mas grande.
Se graficó con dicho proposito la performance de las ventanas obtenidas con *Random Forest* para la ventana de tamaño $300ms$, haciendo coincidir el final de las ventanas del gráfico 9.2 con el final de las ventanas de tamaño $300ms$.

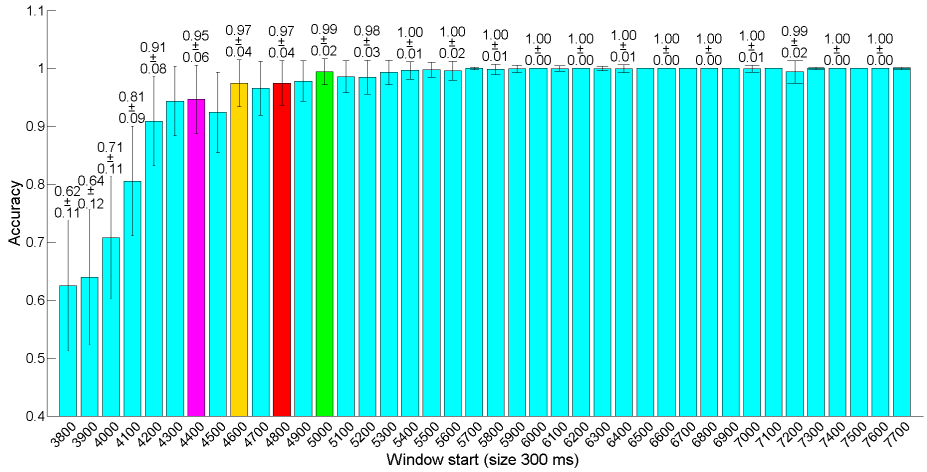


Figura 9.4: Random Forest - Gráfico en barras de la performance en función a la ventana, cada ventana posee $300ms$ - 500 corridas

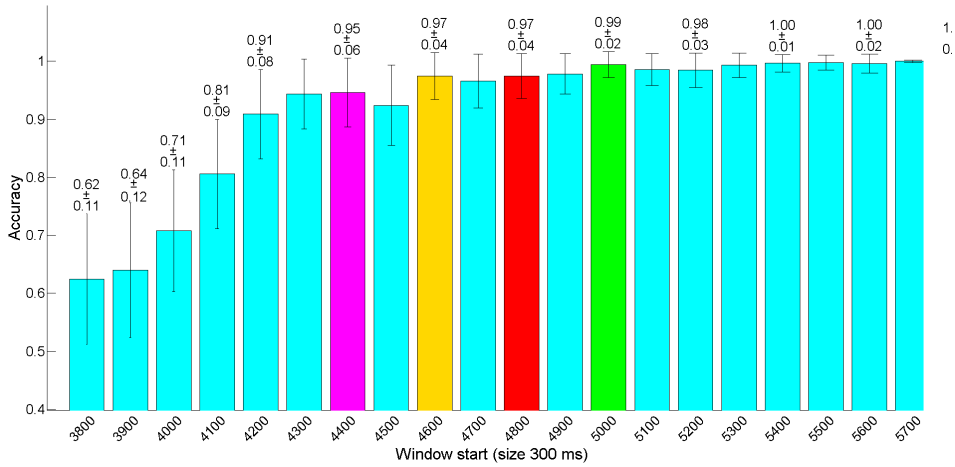


Figura 9.5: Random Forest - Gráfico en barras de la performance en función a la ventana, cada ventana posee $300ms$ - 500 corridas - primeras 20 ventanas

En la figura 9.5 se observan cuatro ventanas pintadas. Las coloreadas con magenta y rojo representan las ventanas de $700ms$ y $1100ms$ de la figura 9.3. La ventana pintada de amarillo indica el punto donde la performance se mantiene por encima de $0{,}94 \pm 0{,}05$, asi como la ventana pintada de verde indica que la performance se mantendrá por encima de $0{,}98 \pm 0{,}03$ para todo el resto de las amplitudes. Tomando de a $300ms$, se necesitan $200ms$ mas para poder garantizar las condiciones planteadas anteriormente. La figura 9.4 es

similar a la figura 9.2, a pesar que se aprecian algunas pequeñas fluctuaciones en los últimos milisegundos de la figura 9.4 que no se perciben en la otra. Se puede observar mejor en la figura 9.6.

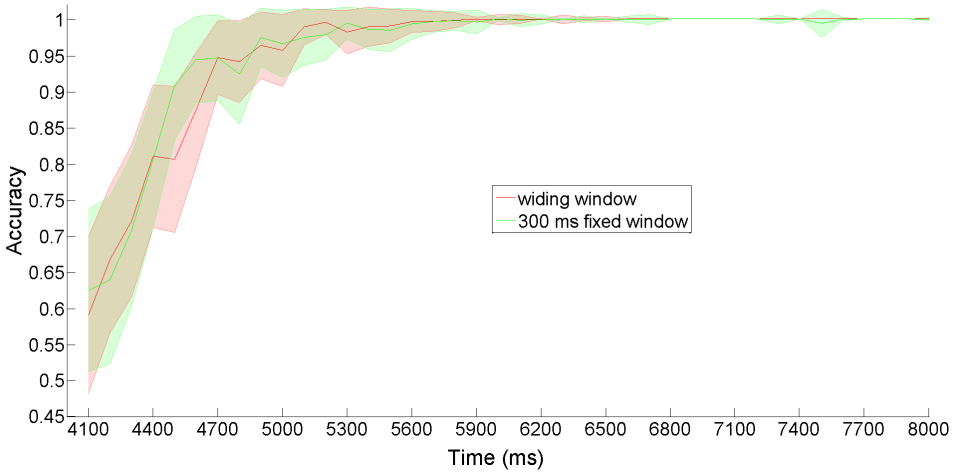


Figura 9.6: Random Forest - Ventana de tamaño fijo en $300ms$ (Verde) vs Ventana ampliándose a medida que avanza el tiempo (Rojo).

Los desvíos de ambas figuras se solapan durante casi todo el tiempo, con lo cual se puede inferir que siempre que el tamaño de ventana sea mayor a $300ms$, y usando *Random Forest* como método de clasificación, la performance no variará significativamente con el tamaño de ventana.

Queda por estudiar entonces cuanto se puede achicar el tamaño de ventana sin perder performance de clasificación.
No se explayará este tema ya que carece de relevancia, debido a que $300ms$ no es una gran amplitud de ventana y a su vez no aporta información respecto a la velocidad con la cual se pueden separar los ensayos.

9.3. Conclusión

A partir que el tono es enseñado a la rata, se deben esperar 700 ms antes de poder clasificar con una performance de $0{,}94 \pm 0{,}05$ y $1100ms$ para garantizar una performance mayor a $0{,}98 \pm 0{,}03$, siempre clasificando con *Random Forest* entrenado con los parametros especificados en el Capítulo 6.5. Se aconseja entrenar el clasificador con todos los datos disponibles a partir del tono, para evitar fluctuaciones, como observamos en el gráfico 9.6.

Apéndice A

Set de datos

A.1. Formato y distribución de los datos

El set de datos fue adquirido en un archivo de matlab dividido en 2 estructuras, las cuales a su vez están subdivididas en 30 sesiones. Cada sesión se subdivide en trials.

Las dos estructuras son:

dataBEH: Contiene los resultados, las respuestas de las ratas a los estimulos auditivos. Se trata de 4 vectores para cada sesión, cada uno asociado a un resultado (GOc, GOi, NOGOc, NOGOi). Cada vector contiene los numeros de trials para esa sesion que se corresponden con la respuesta que el vector representa.

dataVTA: Contiene los spikes neuronales registrados. Posee neuronas (representadas en matrices) para cada sesion. Estas matrices tienen como filas los trials (representado por sus números consecutivos) y como columnas el tiempo, representado en milisegundos. En su interior hay un 0 o un 1 indicando si en ese par (trial, milisegundo) hubo o no un spike.

A.2. Observaciónes y análisis de los datos

Se observa que el número de neuronas es variable por cada sesión, variando entre 1 y 13, con un promedio de 5 neuronas por sesión.

El numero de trials tambien es variable entre 64 y 725 dependiendo de la sesión.

Las neuronas de una misma sesión tienen el mismo número de trials.

Todas las neuronas tienen 8001 ms de datos.

No se observan datos faltantes en el set de datos.

El hecho de que los trials y las neuronas sean variables en cada sesión va a ser una cuestion a considerar a la hora de decidir la forma de aplicar un algoritmo clasificador.

Se puede observar que la distribución de los resultados es bastante despareja:
GOc: 3169
GOi: 367
NOGOc: 2701
NOGOi: 686

Esto se debe a que las ratas fueron previamente entrenadas hasta alcanzar un 80 % de efectividad antes de empezar a registrar las mediciones.

Una distribución tan despareja puede generar dificultades para los clasificadores.

Bibliografía

BREIMAN, L. Random forests. *Machine learning*, vol. 45(1), páginas 5–32, 2001.

CORTES, C. y VAPNIK, V. Support-vector networks. *Machine learning*, vol. 20(3), páginas 273–297, 1995.

FISHER, R. A. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, vol. 7(2), páginas 179–188, 1936.

FUNAHASHI, S., CHAFEE, M. y GOLDMAN-RAKIC, P. Prefrontal neuronal activity in rhesus monkeys performing a delayed anti-saccade task. *Nature*, vol. 365(6448), páginas 753–756, October 1993.

KONONENKO, I. Successive naive bayesian classifier. *Informatica*, vol. 17, páginas 167–74, 1993.

MOHRI, M., ROSTAMIZADEH, A. y TALWALKAR, A. *Foundations of machine learning*. MIT press, 2012.

QUIROGA, R. Q., NADASDY, Z. y BEN-SHAUL, Y. Unsupervised spike sorting with wavelets and superparamagnetic clustering. *Neural Computation*, vol. 16, páginas 1661–1687, 2004.

RAO, C. R. The utilization of multiple measurements in problems of biological classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 10(2), páginas 159–203, 1948.