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

Índice de figuras

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 que permitan 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.

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

Capítulo 2

Set de datos

Resumen: Descripción y Análisis del set de datos

2.1. Introducción

En esté capítulo se pretende explicar el origen del set de datos, así como su composición, estructura y formato.
También 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 (**?**).

Se realizaron 30 sesiones de registro, obteniéndose luego de un proceso de spike sorting mas de 90 neuronas. (**?**)

Luego de someter al animal al estímulo, si el tono se trataba de la frencuencia de 1 kHZ , 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 de 8 kHZ el animal

debía esconder la lengua.

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 -3000 a +2000 ms , coincidiendo el inicio de cada trial con el inicio de la presentación del estímulo.

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 estadisticos 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, descripto por (**?**). 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 sí.
En este caso patrtiular, 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 invertibles. 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 (**?**).

Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest. The generalization error for forests converges a.s. to a limit as the number of trees in the forest becomes large.

(**?**)

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 learning.
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).
Dado que es necesario calcular productos internos entre vectores, es fácil inferir que en espacios de grandes dimensiones los mismos resultan computacionalmente 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 (**?**).

The support-vector network is a new leaming machine for two-group classification problems. The machine conceptually implements the following idea: input vectors are non-linearly mapped to a very high- dimension feature space. In this feature space a linear decision surface is constructed. Special properties of the decision surface ensures high generalization ability of the learning machine. The idea behind the support-vector network was previously implemented for the restricted case where the training data can be separated without errors. We here extend this result to non-separable training data. High generalization ability of support-vector networks utilizing polynomial input transformations is demon- strated.

(?)

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.
A continuación se incluye una cita de (?), en el mismo 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.

The naive Bayesian classifier is fast, incremental, can deal with discrete and continuous attributes, has excellent performance on real life problems and can explain its decisions as the sum of information gains. However, its naivety may result in poor performance in domains with strong dependencies among attributes.

(?)

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 capitulo anterior.
En este capítulo se presentan algunos enfoques tomados y los resultados obtenidos.

4.2. Interpretaciones

La técnica utilizada para entrenar los algorítmos estadísticos utilizados es el aprendizaje supervisado, detallado en (**?**). // Esta técnica consiste en predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto una serie de ejemplos, los datos de entrenamiento.

> *Supervised learning: The learner receives a set of labered examples as training data and makes predictions for all unseen points. This is the most common scenario associated with classification, regression, and ranking problems.*
>
> (**?**)

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-neuronal

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

Figura 4.2: Performance de separacion con LDA de los trials GOc y NOGOc de una sola neurona.

Luego de varias pruebas 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

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 un vector compuesto 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

Observando los gráficos se puede apreciar que la performance de clasificación mejora pero sigue sin ser una performance significativa.

En el capítulo siguiente se plantearán hipótesis que ayudarán a realizar una mejor clasificación.

Capítulo 5

Enfoque por ventanas

Resumen: Se describe en este capítulo el método por el cual se comenzaron a obtener resultados satisfactorios.

5.1. Introducción

Luego de aplicar los métodos descriptos en los capítulos anteriores sin obtener resultados relevantes, se llegó a dos hipótesis sobre las cuales se basa la solución obtenida.

5.2. Hipótesis planteadas

Hipótesis A
Las clases a separar rotan con respecto a las variables de entrada en el tiempo.

Hipótesis B
La información respecto al tono se encuentra guardada en un grupo de neuronas, por lo tanto los clasificadores deberían mejorar su performance cuando la entrada se componga de varias neuronas.

5.3. Implementación

Planteada la hipótesis A, se analizarán pequeños períodos de tiempo con ell fin de ayudar a los algoritmos a clasificar mejor en el caso de rotaciones en los datos. 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.

Cumpliendo entonces con las hipótesis A y B, 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. Entonces, 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 minimamente en todas las neuronas. Se encontraron N trials GOc y M trials NOGOc disponibles como mínimo en todas las neuronas. Se tomó entonces el mínimo valor de los dos (GOc y NOGOc) para conseguir que la matriz tenga la misma cantidad de trials de uno y de otro.

Se tiene la matriz X con j 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 este par de X e Y se lo denominará Xw e Yw ya que hay un par por cada ventana (window). A su vez, se tomará el 70 % de la matriz X para entrenamiento y el 30 % restante para medir la performance de clasificación.

5.4. Resultados

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

Se puede apreciar claramente como a partir del milisegundo x la performance comienza a incrementarse llegando y manteniéndose en un 90 %.

Estos resultados se encuentran dentro de lo esperado debido a que el estímulo auditivo le es enseñado a la rata en el milisegundo z, 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.

Capítulo 6

Iteraciones y validación de 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 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.
Dado que tomamos 500 iteraciones y que la perfomance con su desvío estándar a partir de cierto instante nunca disminuye de un 80 %, podemos afirmar con toda seguridad que la información del tono se encuentra almacenada en las neuronas.

Demostrada en esta instancia que las hipótesis parecen ser ciertas, o al menos dan excelentes resultados, 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
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.

Se puede apreciar que la performance sube a un 99 % donde se estabiliza, su desvío estandar es menor a N a partir del milisegundo M.

6.4. Análisis con Random Forest

Se entrenó un clasificador random forest con 300 árboles.
Se muestran, entonces, los resultados obtenidos hallando el promedio y el desvío estandar de la performance en las 500 iteraciones.

Al igual que con Bayes-Naive se observa una separación de las clases casi perfecta a partir del milisegundo N.

6.5. Análisis con SVM

Se entrenó un clasificador SVM con 30.000 iteraciones.
Se presentan, a continuación, los resultados de la performance de la clasificación en función del tiempo.
Se observa nuevamente que a partir del milisegundo m la información para separar las dos clases (GOc y NOOGOc) se encuentra guardada en las neuronas.

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.

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.

Debido a que el tono es mostrado a partir del milisegundo m, no tiene sentido comparar los métodos antes de este instante, por ende, para poder apreciarlos mejor se promedió el promedio de la precisión para todas las ventanas posteriores al milisegundo 5000. Se presenta el correspondiente gráfico debajo, junto a su desvío estandar.
A pesar que SVM 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 a la hora de clasificar información de spikes neuronales. Se procedió, entonces, a realizar corridas de los métodos con exactamente 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. Comparación de resultados apareados

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.

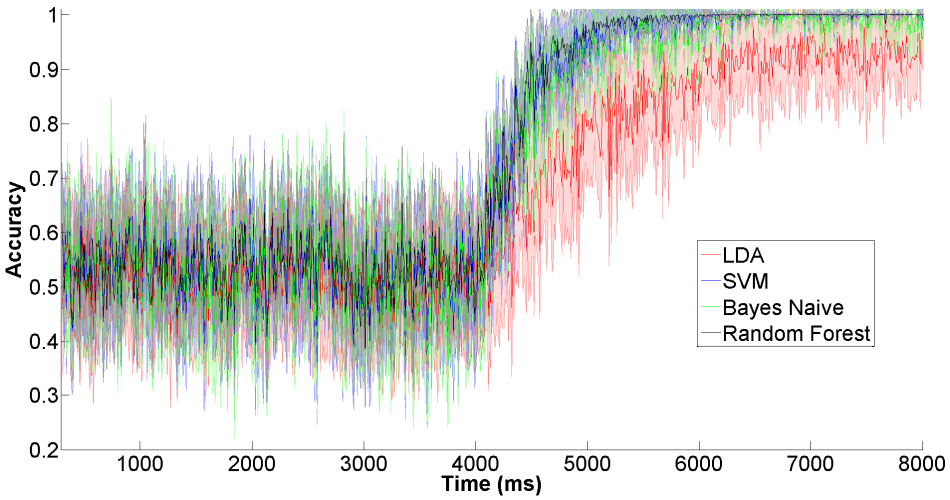

Figura 7.1: Comparacion apareada para todas las neuronas.

La figura **??** permite observar los metodos a lo largo de los 8001 ms.

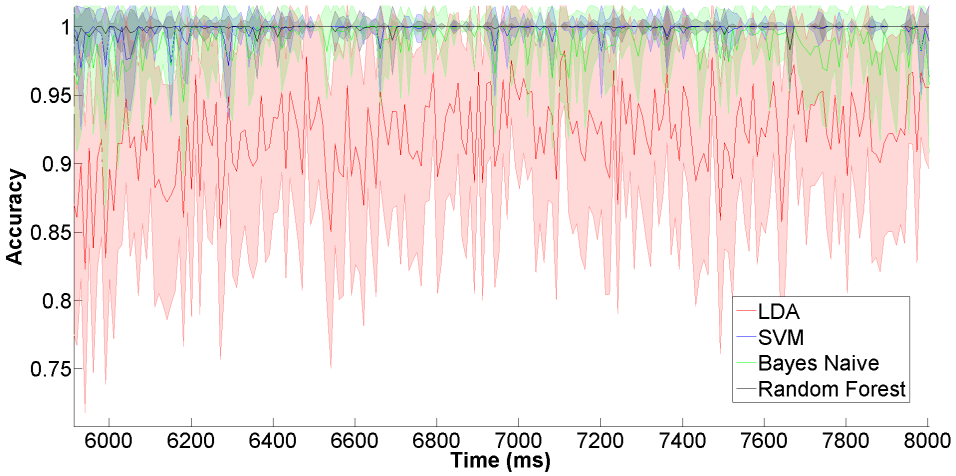

Figura 7.2: Comparacion apareada para todas las neuronas con zoom.

Para poder observar cuales metodos poseen mejor performance se muestra la figura **??** donde se puede apreciar como los metodos de Random Forest y SVM estan por encima de Bayes Naive y LDA durante la mayoría de las ventanas. En el grafico **??** se aprecia lo expresado anteriormente.

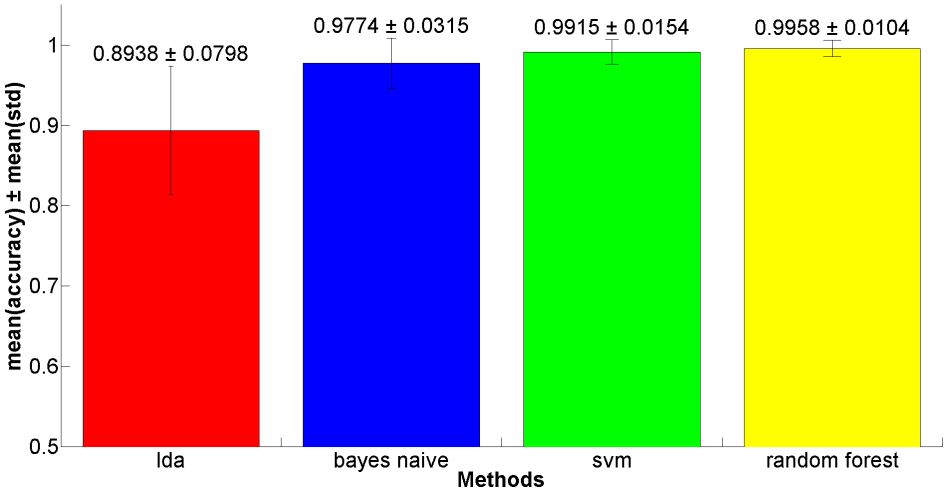

Figura 7.3: Comparacion apareada para todas las neuronas en barras a partir del milisegundo 5000.

Random Forest mayor que LDA	100%
Random Forest mayor que Bayes Naive	88%
Random Forest mayor que SVM	53%
svm mayor que LDA	100%
svm mayor que Bayes Naive	85%
Bayes Naive mayor que LDA	99%

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

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 75 %, 50 %, 25 %, 10 %, 5 % y 3,33 % del total de neuronas. Una vez fijado dicho porcentaje, la selección de las neuronas fue realizada en forma aleatoria. Es necesario tener en cuenta que el total de las neuronas para cada sesión es $N \pm n$, por lo tanto la cantidad total de neuronas para cada porcentaje es:

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

Dados los resultados exitosos plasmados en los capítulos anteriores, se intentará definir cual es el mejor tamaño de ventana dependiendo del problema que se quiera resolver.

9.2. Impacto de la variación del tamaño de ventana

Dada la hipótesis A, variar el tamaño de ventana podría mejorar los resultados obtenidos. A su vez, que el tamaño de ventana 300 ms haya funcionado no termina de comprobar la hipótesis A, puesto que la baja performance de los métodos utilizando los 8001 ms podria deberse al ruido introducido previo al instante en que se le enseña el tono a la rata, ya que a partir de ese momento es cuando se encuentra la información almacenada.
Si este fuera el caso, tomando una ventana de 8001 - t (instante tono), se debería encontrar la mayor performance para el método LDA ya que este separa linealmente y es especialmente sensible a rotaciones.

Adicionalmente, mas allá de la comprobación de esta hipótesis, 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.

La dificultad de comparar la perfomance de diferentes tamaños de ventana recide en que la transformación del espacio de ventanas al espacio del tiempo es distinta dependiendo del tamaño de la ventana.

Si no hubiera rotación en los datos la ventana 8001 - tn debería ser la que muestre una mayor performance midiendo solamente desde tn en adelante. Como se puede apreciar en el gráfico anterior, esto no es lo que sucede, siendo la ventana con tamaño m la que mayor performance presenta.
Por lo tanto, se deberá tomar una ventana de mayor o menor tamaño dependiendo del problema que se quiera resolver.

Capítulo 10

Aplicaciones Prácticas

Resumen: En este capítulo se mostrarán algunas aplicaciones prácticas para los clasificadores estadísticos obtenidos en capítulos anteriores.

10.1. Introducción

En el capítulo anterior se habló del "problema a resolver". En este capítulo se desarrollarán algunos posibles problemas que se pueden resolver utilizando los resultados obtenidos a lo largo de este trabajo.

10.2. Predicción del comportamiento

Supongamos por un momento que volvemos al instante en el que se estaban registrando los spikes correspondientes a las ratas. En vista de los resultados obtenidos anteriormente podemos inferir que con la entrada de n neuronas podemos predecir la respuesta del animal con una seguridad del m % a x ms de haber escuchado el tono. A su vez, variando el tamaño de ventana podríamos obtener la respuesta antes o después con menor o mayor confianza. Adicionalmente, se podria saber qué tono fue enseñado a la rata solamente viendo los spikes de respuesta de la misma. Como se puede apreciar en la figura de abajo, la performance de diferenciación entre tonos es de j %.

10.3.

Capítulo 11

Conclusiones

Resumen:

11.1. Introducción

11.2. Demostración de información auditiva

Apéndice A

Apéndice 1

Resumen:

A.1. Introducción

A.2. 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.3. 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.