

Memoria

Blanca Cano Camarero y Iker Villegas Labairu.

23 de octubre de 2022

Índice de contenidos

Prefacio	3
1 Introducción	4
2 Cálculo de <i>spikes</i>	5
2.1 Lectura de los datos	5
2.1.1 Descripción	5
2.1.2 Requisitos	5
2.2 Lectura de los datos	6
2.3 Diseño del algoritmo de cálculo de <i>spikes</i>	6
2.3.1 Motivación del algoritmo	7
2.4 Determinación de los umbrales	7
3 Cálculo de la información mutua	17
3.1 Abstracción del problema	17
3.2 Cálculo de la información mutua	17
3.3 Formulación del experimento	18
4 Summary	20
References	21
5 Apéndice	22
5.1 Resultado experimentos para la determinación de los umbrales	22

Prefacio

Práctica de la asignatura de Teoría de la Información del máster de Ciencia de Datos de la UAM del curso 2022-2023.

Esta memoria ha sido generada con [Quarto](#) y el lenguaje utilizado ha sido [Python](#).

Las funciones declaradas se encuentra en el directorio **src** y contiene las siguientes:

- `read_data.py`: Lee los datos en formato `csv` y devuelve un `dataframe` con ellos.

los notebooks en formato `.qmd` es la memoria ejecutable y que hace llamadas a tales funciones.

Para poder ejecutar la memoria entera dispone de un **Makefile** cuyas funciones básicas son:

- `make` o `make render` Para renderizar un pdf.
- Visualización de la memoria en html `make preview`.

Para ejecutar alguna casilla concreta puede abrir.

1 Introducción

Añadir descripción de la práctica.

See Knuth (1984) for additional discussion of literate programming.

2 Cálculo de *spikes*

2.1 Lectura de los datos

2.1.1 Descripción

Para gestión de la información se utilizará la biblioteca de pandas, no es necesario gestionar la memoria porque las arquitecturas de nuestros ordenadores la manejan sin problemas.

La estructura de los ficheros viene dada en la información de los datos, en el fichero `InformacionFicheros.txt` y en las tres primeras líneas de los mismos (las cuales deberán de ser obviadas para la lectura del fichero).

2.1.2 Requisitos

- Tener las respectivas biblioteca instaladas (pandas, matplotlib y numpy).
- Los datos deben encontrarse en el path indicado en la variable `data_path`.

En el siguiente fragmento de código puede observar la cabecera de los datos:

- El intervalo de muestreo es de `0.1ms`.
- Hay dos canales, una por cada neurona.
- Y en total se han tomado 19847700 muestreos.

```
print('Datos fichero trozo C')
print(23*'-')
!head -n 14 ./DatosSinapsisArtificial/InformacionFicheros.txt
```

```
Datos fichero trozo C
-----
```

```
La estructura de los ficheros es:
-----
```

TrozoC.txt -> Control
Las tres primeras líneas del fichero son:
Sample interval = 0,100000 ms
Number of channels = 2
Number of samples per channel = 19847700

y a continuación las columnas:
Columna 1 -> LP
Columna 2 -> VD

```
## Data information
sample_interval = 0.1
samples_per_channel_trozoC = 19847700
```

2.2 Lectura de los datos

Para la lectura de los datos se va a utilizar la biblioteca *Pandas* y la función `read_csv` puede encontrar la implementación de la misma en el directorio `src/read_data.py`.

```
from src.read_data import read_data, signal

signal['C'].head(4)
```

	LP	VD
0	0.004883	0.015259
1	0.001526	0.024109
2	-0.010681	0.031128
3	-0.022278	0.041809

Figura 1: Primeras 4 filas de la señal leída.

2.3 Diseño del algoritmo de cálculo de *spikes*

Para calcular los *spikes* se ha optado por utilizar un doble umbral la descripción del algoritmo es la siguiente y la puede encontrar en el fichero `src/signal_to_binary.py`:

Dada una señal `signal` que es una lista unidimensional de la señal. Para que cuento como señal debe de superar el umbral superior `upper_threshold` y ser la primera vez o que ya se haya alcanzado un valor inferior a `lower_threshold`.

Además una vez que se supera el umbral se colocará cuando la tendencia vaya a bajar. Esto queda reflejado con los siguientes estados:

- **Estado 1:** Si `s > upper_threshold` entonces :

- i) `last = s`
- ii) pasar a estado 2.

- **Estado 2:** Si `s < last` entonces:

- i) poner un spike en señal anterior
- ii) `last = -inf`
- iii) pasar a estado 3 Si no entonces:
- iv) `last = s`

- **Estado 3:** Si `s < lower_threshold` entonces:

- i) Cambiar a estado 1

2.3.1 Motivación del algoritmo

Notemos que este algoritmo detecta el *spike* como el primer instante antes de que la señal empiece a decaer (punto azul) y en situaciones donde tras una caída no lo suficientemente baja y una subida aunque sea superior (punto amarillo) se tomaría al primero como punto de *spike*.

Esta decisión se ha tomado ya que filosóficamente se podría entender *spike* como el instante en el que toma un *valor grande* y que el resto son oscilaciones del pico. En caso de que se desee tener el valor amarillo bastaría con subir el umbral superior.

2.4 Determinación de los umbrales

Para determinar los umbral vamos a suponer que la señal sigue una distribución normal, ya que este tipo de distribución modela fenómenos con mecanismos complejos y desconocidos.

La distribución normal posee la propiedad de que

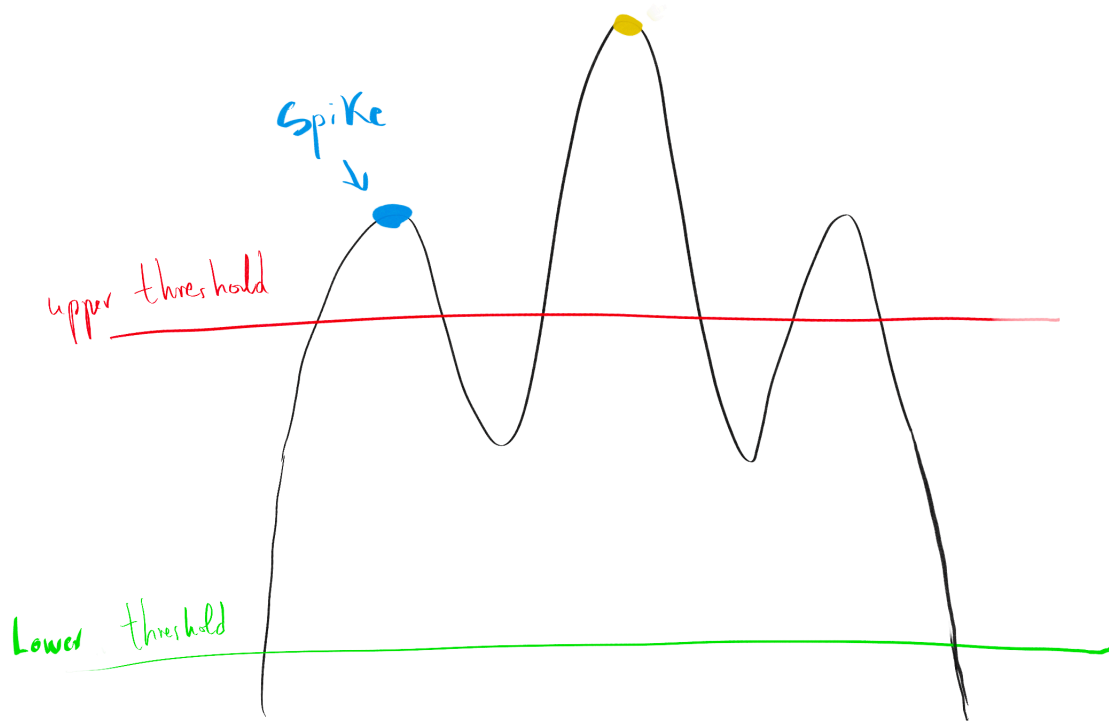


Figura 2.1: Dualidad spikes

Porcentaje de la población dentro de la normal	Distancia a la media
80%	1.281 σ
90%	1.645 σ
95%	1.956 σ
99%	2.576 σ
99.9%	3.291 σ
99.99%	3.891 σ
99.999%	4.892 σ
99.9999%	5.326 σ
99.99999%	6.109 σ

Los *impulsos* son eventos *raros* y por tanto serán aquellos que se encuentren más alejados de la media más

Hemos realizado por tanto un experimento para ver la dependencia entre el umbral seleccionado y el número de *spikes* detectados.

Éste consiste en variar los umbrales conforme a la distancia a media y ver el número el número de spikes detectados (puede consultar la implementación en `src/get_thresholds.py`)

El experimento puede ser ejecutado con `make experimento_umbrales` y se encuentra implementado en el fichero `src/experiment_gets_threshold.py`.

Los resultados han sido los siguientes:

Distancia del umbral bajo	Distancia alta	Umbral bajo	Umbral alto	Número de <i>spikes</i>
1.956	1.956	-0.161	0.161	54464
1.956	2.57	-0.161	0.211	41905
1.956	4.892	-0.161	0.402	31065
2.57	1.956	-0.211	0.161	45446
2.57	2.57	-0.211	0.211	38986
2.57	4.892	-0.211	0.402	31064
4.892	1.956	-0.402	0.161	10408
4.892	2.57	-0.402	0.211	10343
4.892	4.892	-0.402	0.402	10241

Vemos que el umbral bajo determina crucialmente el número de *spikes* realizaremos una inspección visual para ver qué está aconteciendo en varias secciones aleatorias de la muestra (si se encuentra en un entorno de ejecución podría modificar los valores de `higher_thresholds` y de `lower_threshold`).

Hemos repetido el experimento para cada uno de los trozos y cada neurona. Puede consultar los resultados en el apéndice 5.1 o bien ejecutarlos por si mismo `make experimento_umbrales`; ese comando mostrará los resultados en la terminal y además los almacenará en el directorio `experiment_results/get_threshold.txt`.

A la vista de los resultados de los primeros umbrales en un primer estadio hemos tomado como criterio tener los umbrales lo más *grandes* posibles siempre y cuando el número de spikes no decaiga dramáticamente. La selección primera ha resultado:

Tabla 2.3: Selección primera de umbrales

Trozo	Neurona	Umbral inferior	Umbral superior	Número de spikes
C	LP	-0.320	0.402	30308
C	VD	-0.084	0.205	21246
R	LP	-0.534	0.920	24076
R	VD	-0.085	0.206	17618
G	LP	-0.316	0.397	25889
G	VD	-0.120	0.207	13127

Como método de validación de estos umbrales hemos formulado el siguiente experimento:

Para cada trozo y neurona se realizará una inspección visual de cinco rango aleatorios de valores, si para estos se escapan *impulsos* que visualmente se consideran válidos se retocará el umbral.

Puede ejecutar el experimento con: `make plot_experimental_thresholds` que no solo mostrará en pantalla las gráficas si no que las almacena en la carpeta `img/04_calculo_spikes`.

Vamos a proceder a mostrar algunos de los ejemplos representativos:

Para la señal C neurona LP va a ser necesario subir un poco el umbral inferior, ya que en dos casos spoikes de rangos aleatorios se ha escapado un estímulo por el rango inferior.

Es por ello que vamos a considerar el nuevo umbral bajo como `-0.211`.

Para la misma señal la neurona VD ocurre un efecto parecido que procederemos a paliar con aumentando el umbral.

Es por ello que lo subiremos a `-0.041`.

Para la señal R de la neurona LP los *spikes* se separan demasiado de los umbrales.

Vamos a ajustar los umbrales a

Para el caso de la misma señal neurona VD, podemos observar que es el umbral es correcto.

Para G LP los umbrales están demasiado cerca de la media.

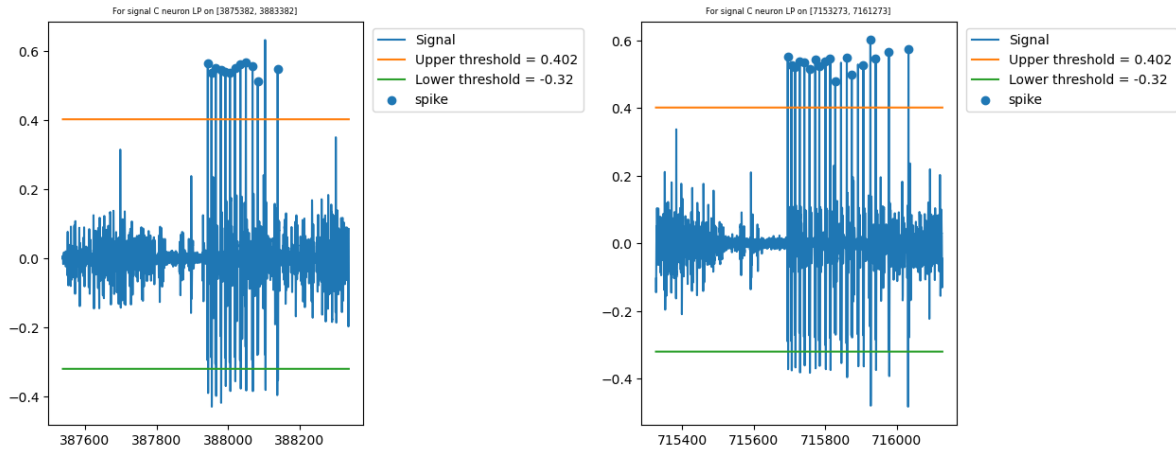


Figura 2.2: Umbral inferior demasiado bajo para señal C, neurona LP

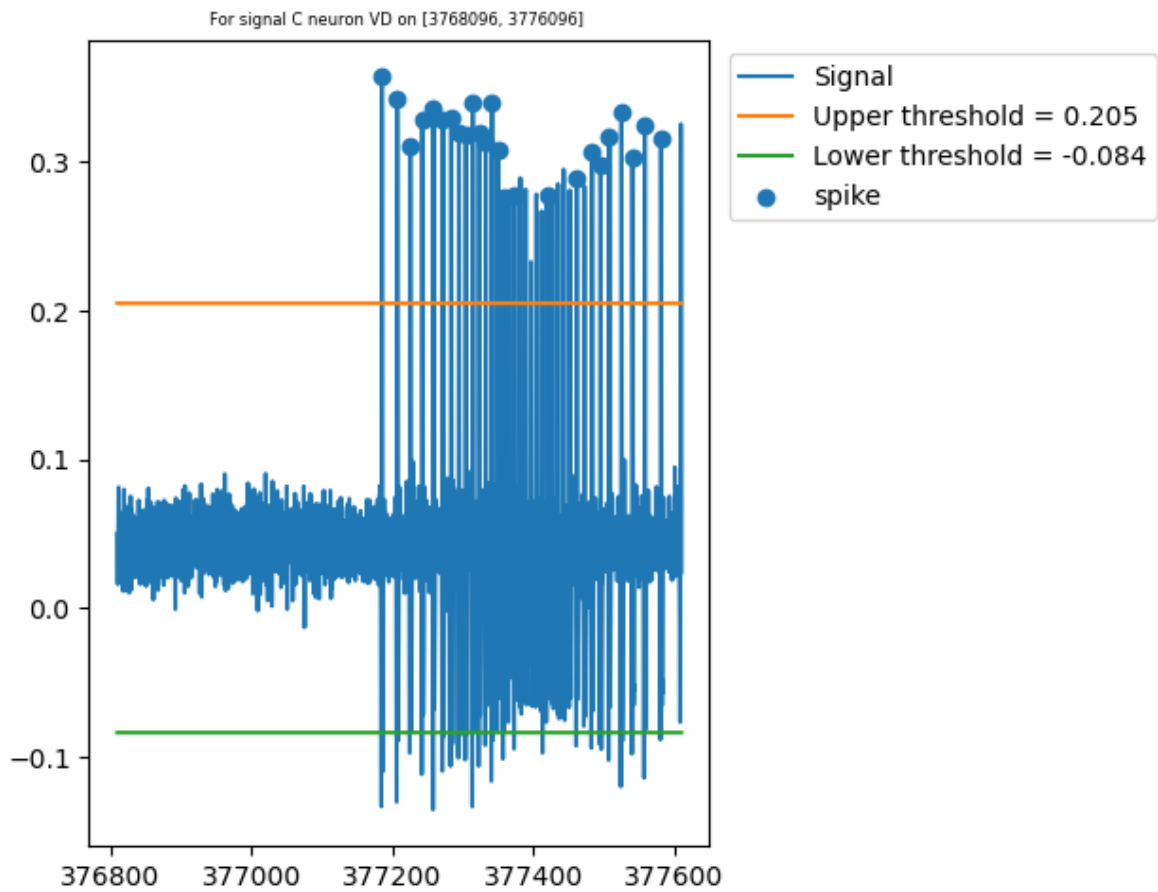


Figura 2.3: Umbral inferior demasiado bajo para señal , neurona VD

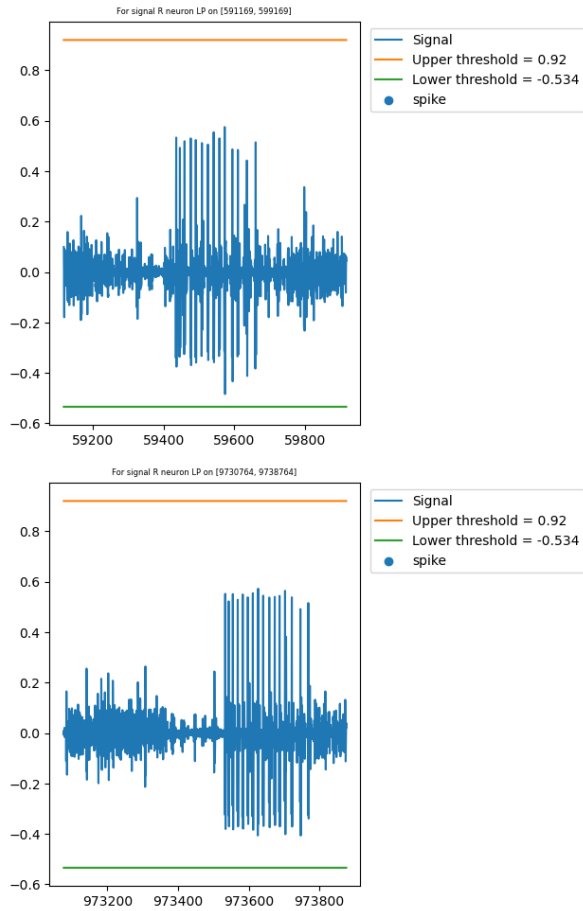


Figura 2.4: Umbral demasiado alejado de la señal para la señal R neurona LP.

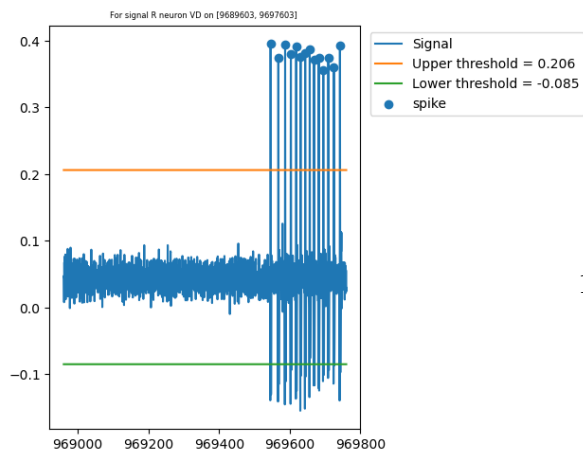
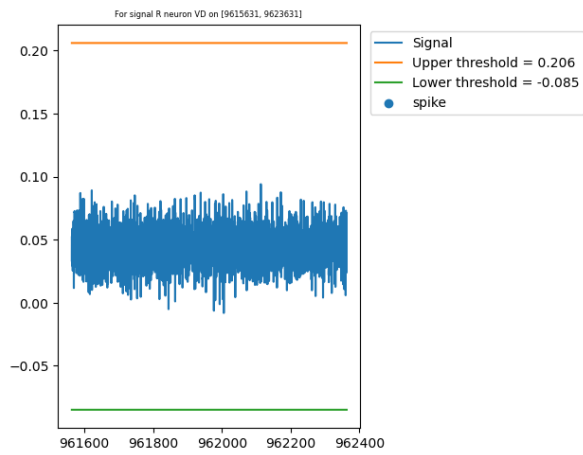
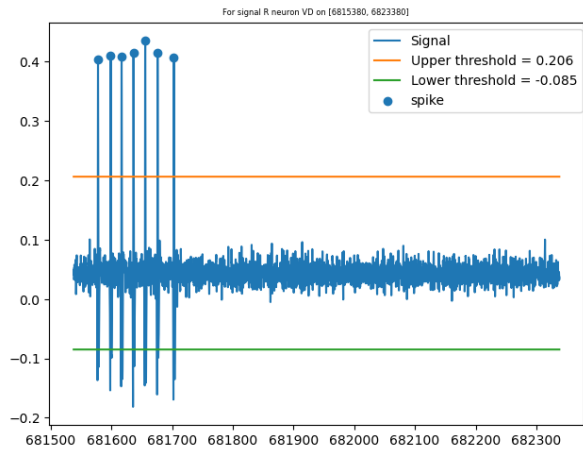
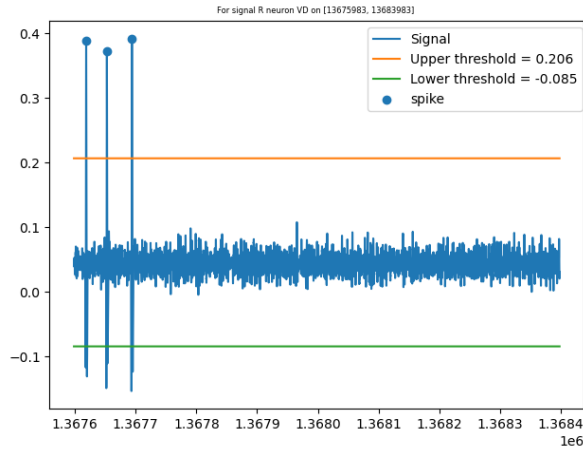


Figura 2.5: Umbrales correctos para R neurona VD.

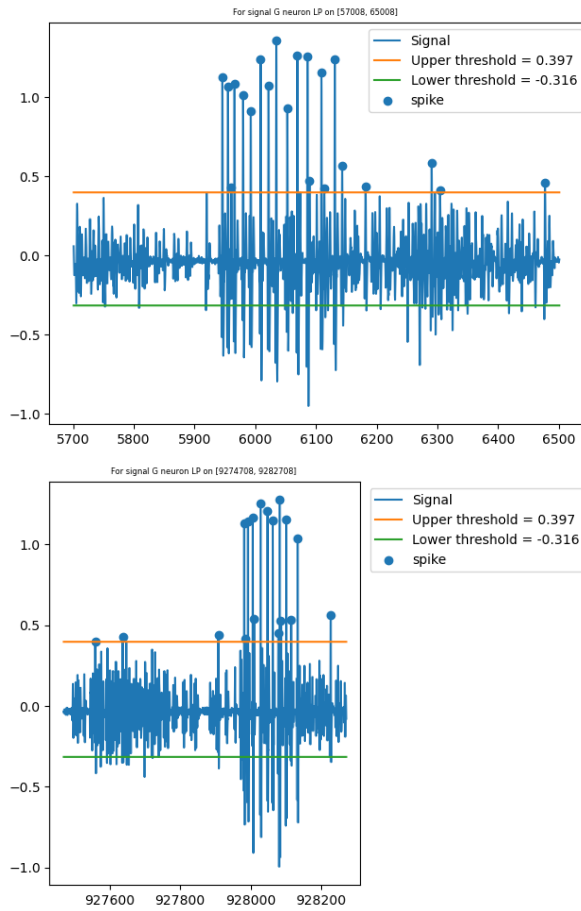


Figura 2.6: El umbral superior está demasiado cerca

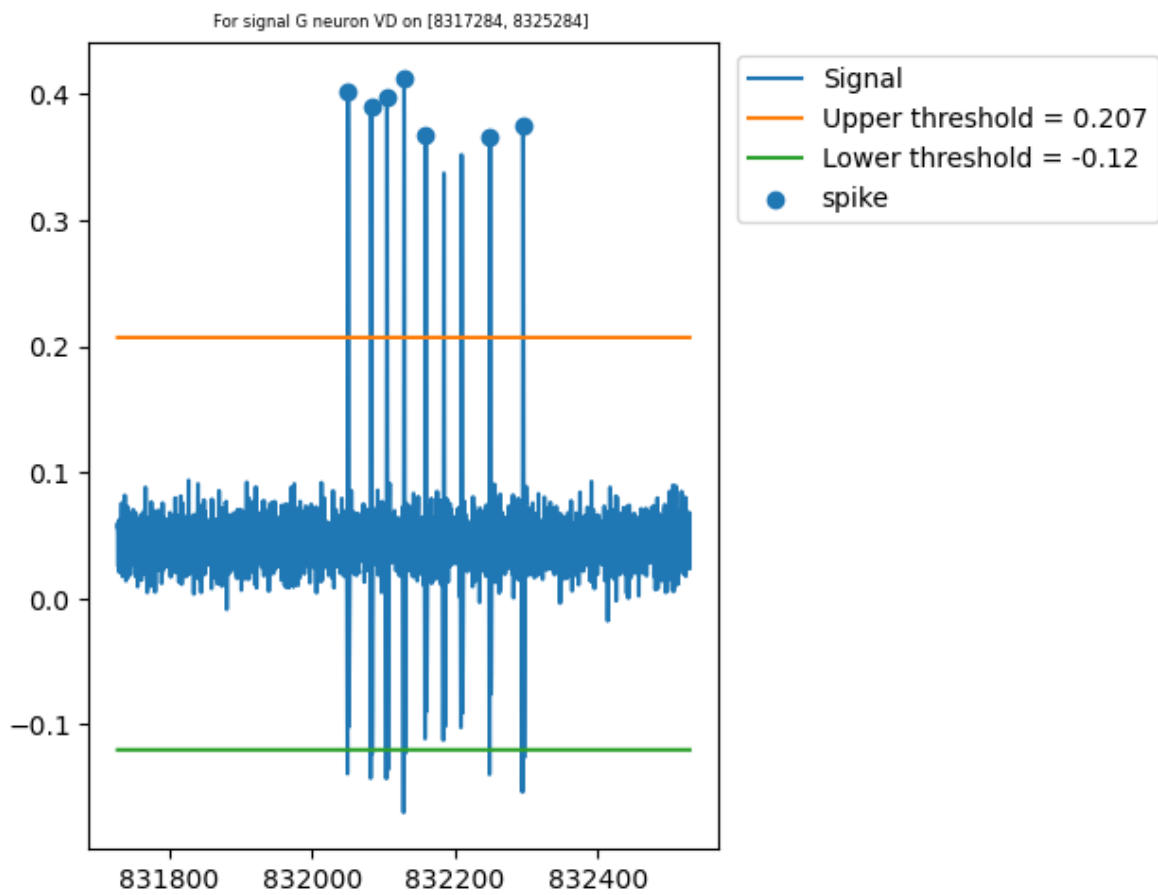


Figura 2.7: Para la señal G neurona VD el umbral inferior es demasiado bajo.

Tras estas observaciones puede se ha concluido que

Trozo	Neurona	Distancia	σ inferior	σ superior	Umbral inferior	Umbral superior
:-:	:-:	:-:	:-:	:-:	:-:	:-:
C	LP	2.57	4.892	-0.211	0.402	
C	VD	2.57	3.891	-0.041	0.172	
R	LP	1.4	1.956	-0.306	0.348	
R	VD	3.891	4.892	-0.085	0.206	
G	LP	5.326	> 15	-0.432	0.75	
G	VD	3.891	4.892	-0.087	0.207	

: Selección final de umbrales {#tbl-threshold2}

Por la determinación de estos umbrales puede observarse cierta asimetría de la señal: *sube más de lo que baja*, también es llamativo que para una misma neurona cada umbral varíe, esto puede deberse o a la naturaleza propia de la señal o que el aparato de medida o condiciones del experimento sean distintas.

Puede observar las gráficas finales en la carpeta `img/04_calculo_spikes` las que comiencen con 2 o bien ejecutando usted mismo `make plot_experimental_thresholds` (o directamente el programa `python src/experiment_view_threshold.py`).

3 Cálculo de la información mutua

Vamos a proceder con el cálculo de la información mutua entre las dos señales.

3.1 Abstracción del problema

Una vez preprocesada la señal a una secuencia binaria, donde uno significa hay estímulo; la probabilidad de que aparezca una determinada palabra (cadena de n -bits consecutivos) se trata de un proceso estocástico de Poisson.

Es decir, la probabilidad de que pueda aparecer cierta cadena.

El estimador máximo verosímil de una distribución de Poisson es la media, luego fijado un tamaño de palabra y un *stride* calcularemos la frecuencia de cada casuística.

3.2 Cálculo de la información mutua

La información mutua para dos variables X, Y aleatorias se puede definir como

$$MI(X, Y) = S(X) + S(Y) - S(X, Y), \quad (3.1)$$

donde S denota a la entropía y responde a las siguientes fórmulas:

$$S(X) = - \sum_i p(x_i) \log_2(p(x_i)) \quad (3.2)$$

Las implementaciones de estas funciones se encuentran en el ejecutable `src/formulas.py`.

3.3 Formulación del experimento

Para el cálculo de la información mutua se va a realizar el siguiente proceso:

1. Se transforma la señal analógica en una binaria (ver algoritmo `src/signal_to_binary.py`).
2. Se fija un tamaño de ventana y *stride*.
3. Para la ventana y *stride* de las señales *X* e *Y* se obtienen un array de sus palabras.
4. Se calcula la probabilidad de tales

A la vista de los resultados de cada trozo podemos afirmar que el *stride* no juega un papel fundamental pero que sí lo hace el *stride*.

Es por ello que vamos a fijar de ahora en adelante *stride* = *bits* y vamos a aumentar el ancho de ventana, esto es:

Calcularemos la distancia mínima entre spikes para dos señales con el código `/src/spike_distance.py` cuyo resultado es:

Tabla 3.1: Mínima distancia entre los *spikes*

Trozo	Neurona	Intervalo mínimo entre <i>spikes</i>	Media	std
C	LP	24	9994824.932	6056023.942
C	VD	10	10229353.488	5779379.905
R	LP	16	8356519.059	4859624.688
R	VD	16	8422905.584	4842002.988
G	LP	17086	9173668.302	4109764.716
G	VD	65	8413845.722	4894766.835

Es por ello que ampliaremos el tamaño de ventana es decir transformaremos la señal binaria de cada trozo.

Donde el radio de acción es, para cada trozo:

$$\text{radio acción} = \min(\text{distLP}, \text{distVD})$$

Tabla 3.2: Radio de acción

Trozo	Radio de acción
C	10
R	16
G	65

Las nuevas señales binarias vendrán dadas por:

$$\text{señal binaria nueva}[i] = \max \text{señal antigua}[i * \text{radio acción} : (i+1) * \text{radio acción}]$$

De esta manera la señal tendrá un uno si ya lo había o un dos si no lo había.

4 Summary

In summary, this book has no content whatsoever.

References

Knuth, Donald E. 1984. «Literate Programming». *Comput. J.* 27 (2): 97-111. <https://doi.org/10.1093/comjnl/27.2.97>.

5 Apéndice

5.1 Resultado experimentos para la determinación de los umbrales