Memoria

Blanca Cano Camarero y Iker Villegas Labairu.

4 de noviembre de 2022

Indice de contenidos

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 llamdas 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.

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 lineas 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 obscilaciones 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

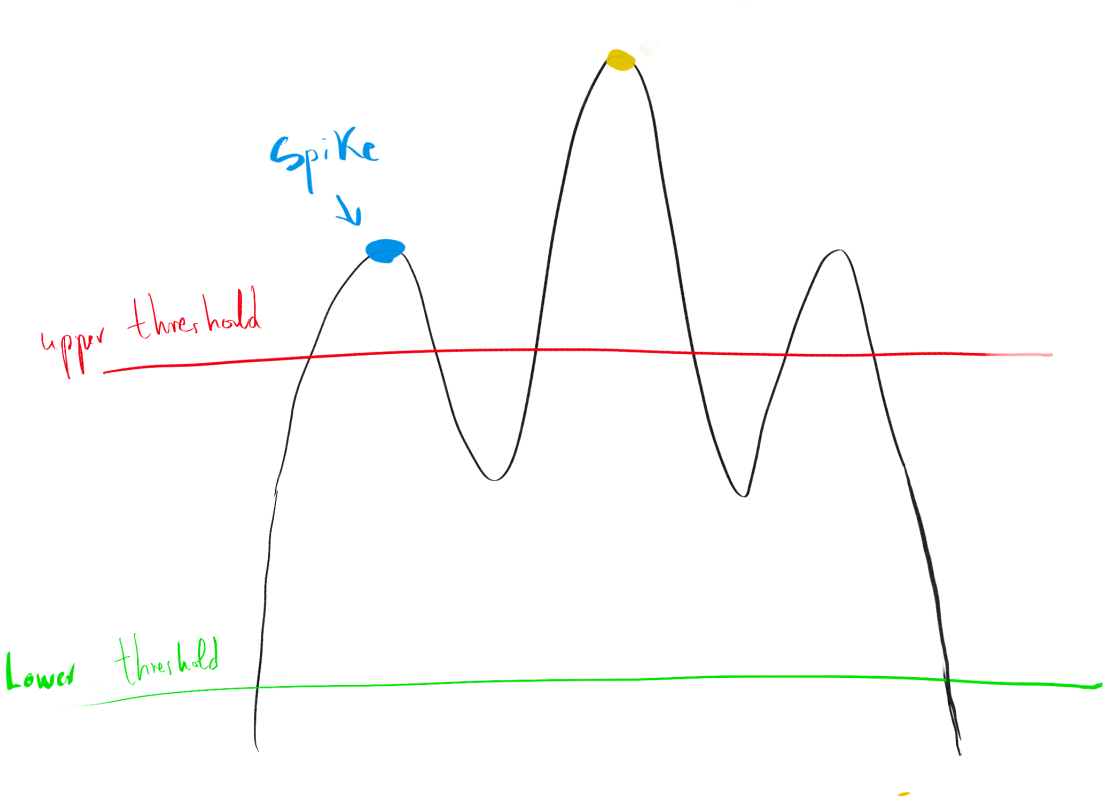

Figura 2.1: Dualidad spikes

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

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 *spikes*
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 inspencción viual para ver qué está acontenciendo 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 7.1 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 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 subieremos 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.

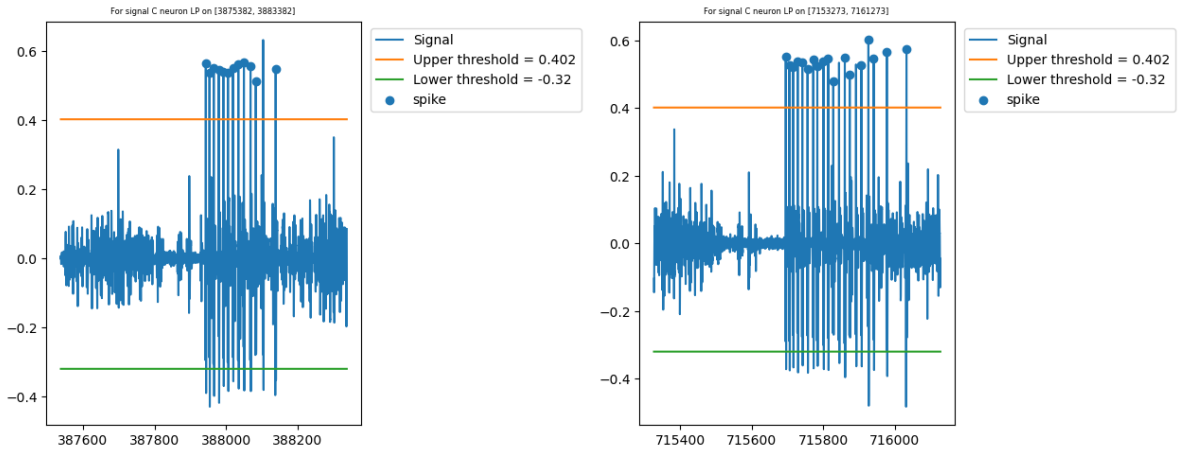

Figura 2.2: Umbral inferior demasiado bajo para señal C, neurona LP

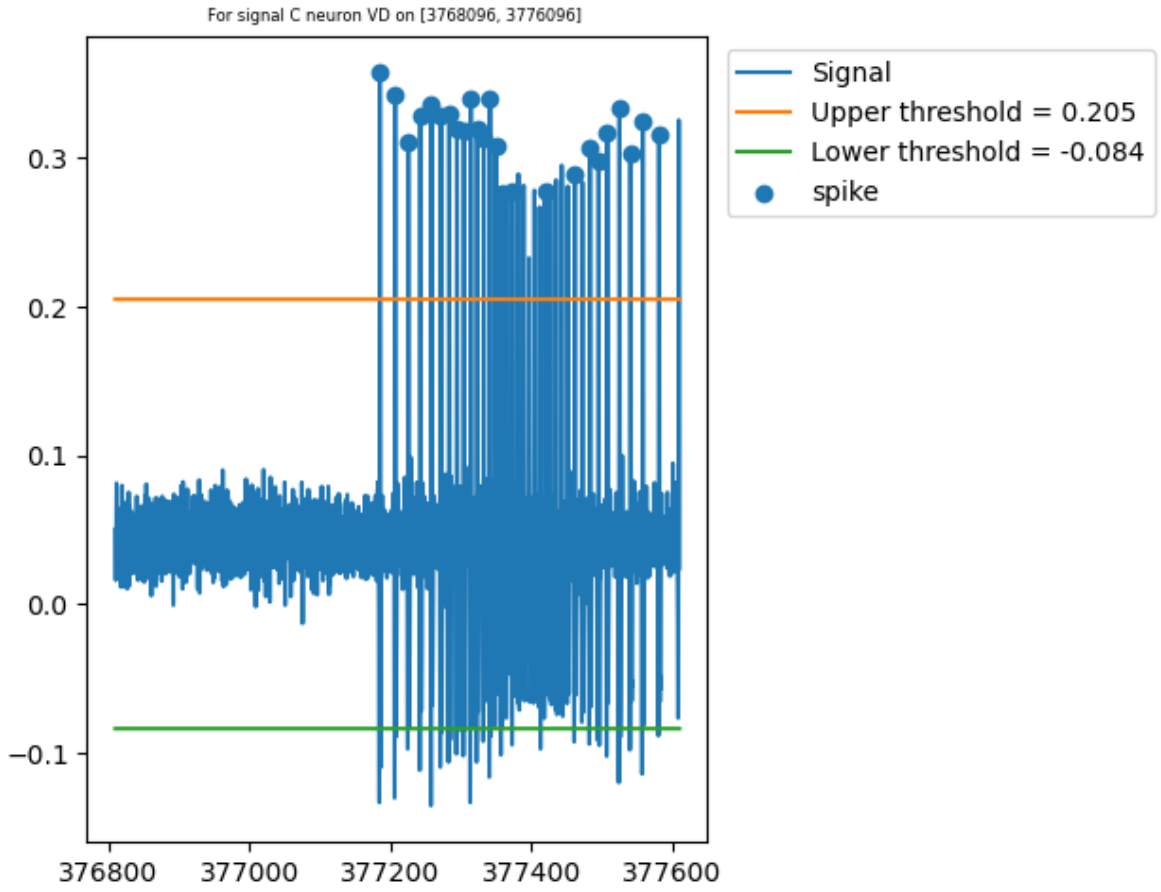

Figura 2.3: Umbral inferior demasiado bajo para señal , neurona VD

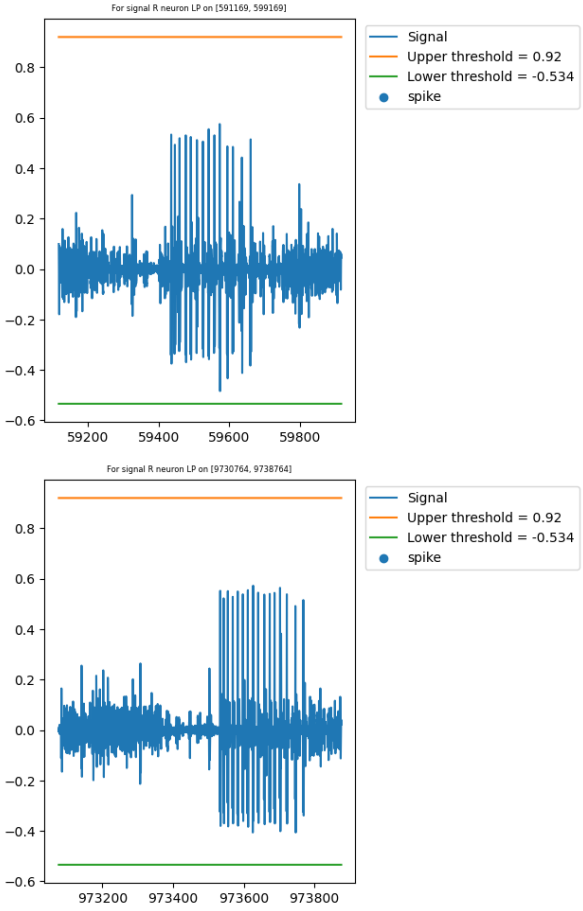

Figura 2.4: Umbrales demasiado alejados de la señal para la señal R neurona LP.

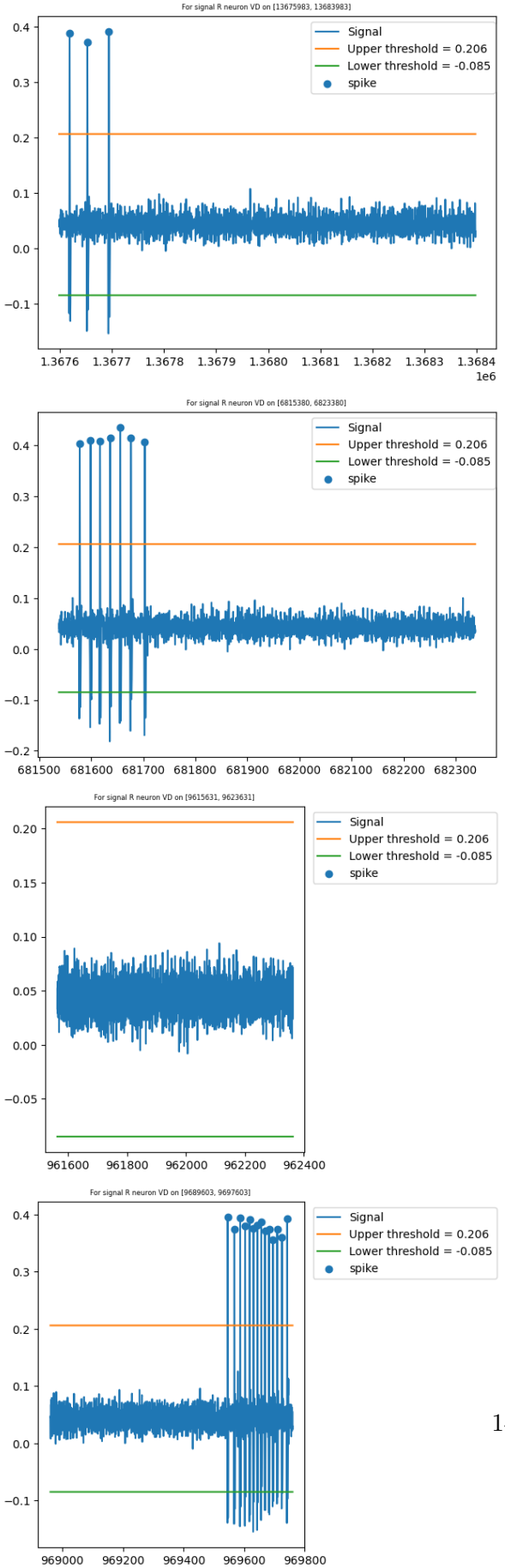

Figura 2.5: Umbrales correctos para R neurona VD.

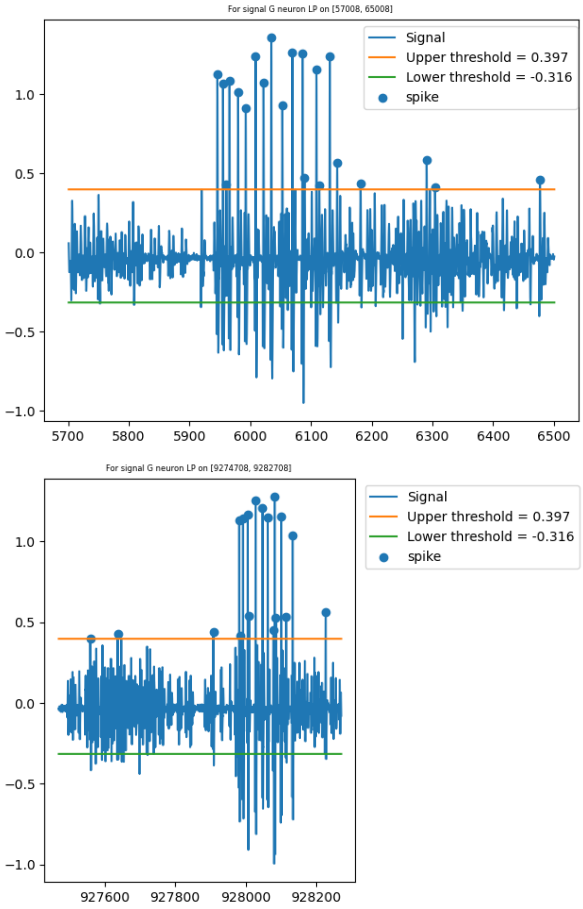

Figura 2.6: El umbral superior está demasiado cerca

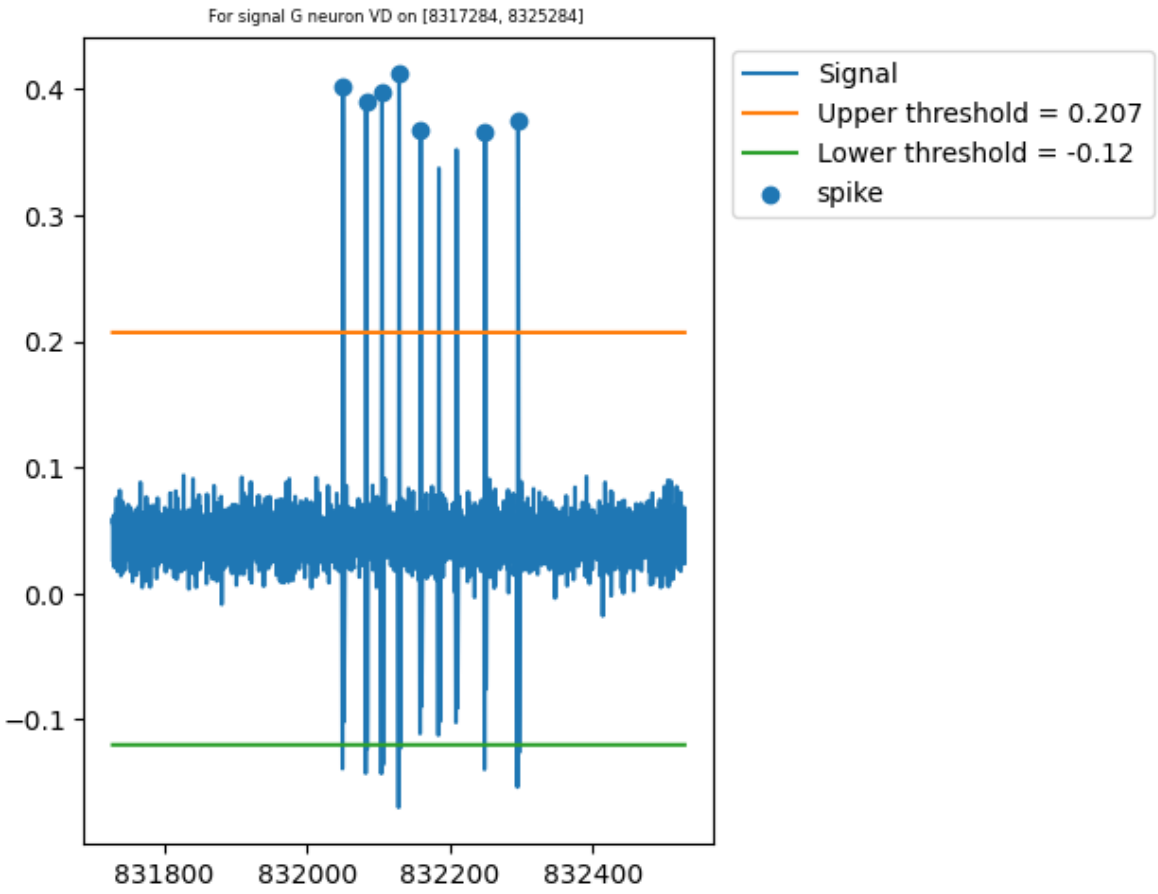

Figura 2.7: Para la señal G neurona VD el umbral inferior es demasiado bajo.

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 un 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 definit como

$$MI(X,Y) = S(X) + S(Y) + S(X,Y), \tag{3.1}$$

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

$$S(X) = -\sum_i p(x_i)\log_2(p(x_i)) \tag{3.2}$$

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 un una binaria (ver algoritmo `src/signal_to_binary.py`).
2. Se fija un tamaño de ventana y *stride*.
3. Para la venta 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 tamaño de ventana.

Es por ello que vamos a fijar de ahora en adelante $stride = bits$ y vamos a aumentar el ancho de ventana, tomaremos las siguientes casuísticas y las compararemos a posteriori:

La condición de tamaño de ventana: - No genera ninguna ventana falsa. - Cuantil 0.05 de las distancias entre *spikes*. - Aceptaremos un 5% de ventanas falsas.

La implementación del cálculo de ventana se realiza en `/src/spike_distance.py` puede ejecutar tal fichero o hacer `make calcular_distancia_spike` el código se segmenta en: - Calcular las distancias entre spikes consecutivos. - Calcular la distancia mínima. - Calcular el cuantil 0.05 de las distancias calculadas. - Realizar una búsqueda binaria del tamaño de ventana que hace que al rededor del 5% sean ventanas falsas.

Los resultados han sido los siguientes:

Tabla 3.1: Mínima distancia entre los *spikes*

Trozo	Neurona	Intervalo mínimo entre spikes	Media distancias	std
C	LP	24	9994824.93	6056023.94
C	VD	10	10229353.45	5779379.91
R	LP	16	8356519.06	4859624.69
R	VD	16	8422905.58	4842002.99
G	LP	17086	9173668.30	4109764.72
G	VD	65	8413845.72	4894766.83

Tabla 3.2: Distancia de *spike* para cuantil de distancia de 0.05

Trozo	Neurona	Distancia cuantil 0.05
C	LP	108
C	VD	86
R	LP	41
R	VD	96
G	LP	26336
G	VD	92

Trozo | Neurona | Porcentaje ventanas falsas | Distanca admitir ventanas falsas

Tabla 3.3: Tamaño de ventana para cuantil de distancia de 0.05

Trozo	Neurona	Tamaño cuantil 0.05	Porcentaje ventanas falsas	Tamaño admitir ventanas falsas
C	LP	108	4.91%	186
C	VD	86	4.94%	207
R	LP	41	5.01%	124
R	VD	96	4.92%	213
G	LP	26336	4.72%	153362
G	VD	92	5.02%	247

Tabla 3.4: Tamaño de ventana en función de un porcentaje de ventanas falsas

Trozo	Neurona	Porcentaje ventanas falsas	Distanca admitir ventanas falsas
C	LP	4.91%	186
C	VD	4.94%	207
R	LP	5.01%	124
R	VD	4.92%	213
G	LP	4.72%	153362
G	VD	5.02%	247

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

Para ello el nuevo acción entiende que solo habrá un 5 de ventanas falsas

Donde el tamaño de ventana nuevo, para cada trozo vendrá dado como:

$$\text{tamaño ventana} = \min(distLP, distVD) + 1$$

Tabla 3.5: Ventanas máximas calculadas

Trozo	Ventana máxima distancia	Ventana máxima cuantil 0.5	Ventana máxima falsas
C	11	87	187
R	17	42	125
G	66	93	248

Las nuevas señales binarias vendrán dadas por:

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

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

4 Cálculo de entropía normaliza o información mutua normalizada

Esta medida se usa para medir la transferencia de información del estímulo S a la neurona respuesta R.

Viene dada por la siguiente expresión

$$E_{RS} = \frac{MI_{RS}}{H(S)}$$

Donde H ya hemos comentado que es la entropía de S, es decir la máxima cantidad de información que se puede transmitir del estímulo a la neurona respuesta.

Está acotado entre

$$0 \leq E_{RS} \leq 1$$

Si $E_{RS} = 0$ significa que toda la información es perdida, es decir respuesta y estímulo son completamente independientes. $E_{RS} = 1$ sería la sincronización completa.

5 Suposición de otro tipo de codificación de eventos

Se pretende en este apartado utilizar otro tipo de codificación de eventos para el cálculo de las probabilidades y la información mutua.

En nuestro caso vamos a proponer el sistema de codificación SAX (ver artículo Lin et al. (2003)).

5.1 Descripción del sistema de codificación SAX

SAX permite reducir una serie temporal de tamaño n a otra de tamaño w usualmente $w << n$.

El tamaño del alfabeto,a, será con la restricción de que $a > 2$.

Los pasos a seguir son:

1. Transformación de los datos en **PAA** *Piecewise Aggregate Approximation*

De la siguiente manera: Una serie temporal C de longitud n puede ser represetada en un vector $\bar{C}$ de dimensión w.

El elemento i-ésimo de $\bar{C}$ viene dado por

$$\bar{c}_i = \frac{w}{n} \sum_{j=\frac{n}{w}(i-1)+1}^{\frac{n}{w}i} c_j.$$

Notemos que esto no es más que una media de los elementos contiguos.

2. **Discretización**

Cada componente de la nueva señal transformada será mapeado por un símbolo en función del rango de valor en que se encuentre.

Para ello se definen los *breakpoints*, queno son más que una lista de números $B = b_1, ..., b_{a-1}$ de tal forma que su area bajo un normal $\mathcal{N}(0,1)$ sea para cada uno de ellos $\frac{1}{a}$. Además se tiene que $b_0 = -\infty$ y $b_a = +\infty$.

De esta manera formaremos las palabras $\hat{C}$ que vendrá dada como

$$\hat{c}_i = \alpha_j, \text{ sii } \beta_{j-1} \leq \beta_j$$

y con esto se obtendría la nueva señal.

5.2 Observaciones

Notemos que se tienen dos variables libres en este sistema:

- w el tamaño de PAA, que de manera implícita debe de ser un divisor de n el tamaño original para mayor comodidad.
- a el tamaño del alfabeto.

5.3 Sobre nuestra implementación

Puede encontrar la implementación de los algoritmos descritos en `src/SAX.py`, en ellos se encuentran fielmente escritos los pasos mencionados.

Cabe destacar que se han tomado las siguientes decisiones en diseño:

- Por eficiencia y poder reutilizar el código de información mutua se ha tomado por alfabeto a un subconjunto de tamaño a de los números reales.

5.4 Experimentación con SAX

5.4.1 Descripción del experimento

Pretendemos calcular la información mutua en distintas cirscunstancias y compararlas con los datos anteriores.

Con este fin w el tamaño de la nueva señal de un trozo T vendrá dado por

$$w = \frac{\text{Tamaño de la señal del trozo } T}{w'}$$

$ donde w' es el tamaño de palabra que se usó en el trozo T de los apartados anteriores.

Por limitaciones temporales y tiempo de costo se han seleccionado las siguientes combinaciones de prueba:

Tabla 5.1: Combinaciones de tamaño de palabra w' para SAX que se van a realizar{#tbl-sax-palabra}

Trozo	Tamaños w'
C	11,87,187
R	17,42,125
G	66, 93,248

Tamaños de alfabeto: $1, 2, 5, 10, 20$ y 50 independientemente del trozo que sea.

Tamaños de bits con el que calcular la información mutua: $1, 2, 3, 4, 5, 6, 7$ y 8 independientemente del trozo que sea.

Para el experimento se han realizado todas las combinaciones posibles de estos parámetros y se ha calculado la información mutua pertinente.

Puede ejecutar el experimento realizando `make calcular_sax` o directamente escribiendo `python src/experiment_sax_mi.py`.

5.4.2 Resultados obtenidos

5.4.3 Trozo C

Puede encontrar la tabla completa con los resultados en el apéndice 7.2

5.5 Notas

- Es muy similar a lo que nosotros hemos hecho, solo que este no discretiza en 0 y 1 solo (a no ser que se indique el alfabeto así) y se usa un percentil homogéneo -> es mejor lo que hemos hecho nosotros para detectar los outliers.

6 Summary

In summary, this book has no content whatsoever.

References

Lin, Jessica, Eamonn Keogh, Stefano Lonardi, y Bill Chiu. 2003. «A Symbolic Representation of Time Series, with Implications for Streaming Algorithms». En *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2-11. DMKD '03. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/882082.882086.

7 Apéndice

7.1 Resultado experimentos para la determinación de los umbrales

7.2 Información mutua obtenida para la codificación SAX

7.2.1 Trozo A

trozo	bits	words	alphabet size	IM(LP,VD)
C	1	11	1	0.00024815504515718345
C	2	11	1	0.001028528475531476
C	3	11	1	0.0021732004578957076
C	4	11	1	0.003579663704288194
C	5	11	1	0.005386666234739712
C	6	11	1	0.007695825616436114
C	7	11	1	0.010386906300554832
C	8	11	1	0.013729273621681859
C	1	87	1	0.0002849437766737073
C	2	87	1	0.0010789203951092619
C	3	87	1	0.0024516116162923707
C	4	87	1	0.004059343543587723
C	5	87	1	0.006087685587868652
C	6	87	1	0.00940811921254836
C	7	87	1	0.013262453561221355
C	8	87	1	0.01610577817892711
C	1	187	1	0.00010869489163545243
C	2	187	1	0.0007406252047386097
C	3	187	1	0.001830320692957077
C	4	187	1	0.0036377228714716825
C	5	187	1	0.006689120569749907
C	6	187	1	0.010156060947295709
C	7	187	1	0.01563190742403764
C	8	187	1	0.020783911704893132
C	1	11	2	0.0005182298593626733
C	2	11	2	0.0024795852806711594

trozo	bits	words	alphabet size	IM(LP,VD)
C	3	11	2	0.0067956803236759455
C	4	11	2	0.013927773912149277
C	5	11	2	0.03151171221116478
C	6	11	2	0.08216009739659569
C	7	11	2	0.22058277844140228
C	8	11	2	0.5569383137386641
C	1	87	2	0.00035897039574805945
C	2	87	2	0.001361959194343676
C	3	87	2	0.006346304795460611
C	4	87	2	0.02673803655749296
C	5	87	2	0.1228606234356242
C	6	87	2	0.41818007907327726
C	7	87	2	1.078962133677889
C	8	87	2	2.2358763601826492
C	1	187	2	0.00021414267703434575
C	2	187	2	0.0018672074628094393
C	3	187	2	0.011107682310548483
C	4	187	2	0.05337350049381939
C	5	187	2	0.21877495621872
C	6	187	2	0.6321478810593355
C	7	187	2	1.455086240946569
C	8	187	2	2.755969934814372
C	1	11	5	0.002644232630698795
C	2	11	5	0.009773159338747561
C	3	11	5	0.044856554473932775
C	4	11	5	0.2897626915538929
C	5	11	5	1.1166215209535455
C	6	11	5	2.737896224607141
C	7	11	5	4.885394705950354
C	8	11	5	7.1549160545849375
C	1	87	5	0.002675269056581442
C	2	87	5	0.016125860153871407
C	3	87	5	0.19732480171007438
C	4	87	5	1.163783987589479
C	5	87	5	3.2532443838401033
C	6	87	5	5.901220205850953
C	7	87	5	8.290037828698518
C	8	87	5	10.059693429503145
C	1	187	5	0.002117483486257399
C	2	187	5	0.021992328126568594
C	3	187	5	0.4190740942927267

trozo	bits	words	alphabet size	IM(LP,VD)
C	4	187	5	2.3199764359542314
C	5	187	5	5.473442410107175
C	6	187	5	8.48113421257955
C	7	187	5	10.660943808648625
C	8	187	5	12.05381465773515
C	1	11	10	0.004024574980462603
C	2	11	10	0.023398301841709213
C	3	11	10	0.4801280544736759
C	4	11	10	2.745372664491679
C	5	11	10	6.376897634191838
C	6	11	10	9.953007527809302
C	7	11	10	12.711268637131639
C	8	11	10	14.541391428139981
C	1	87	10	0.003951176148957458
C	2	87	10	0.09534519380188833
C	3	87	10	2.0148750120601076
C	4	87	10	6.894191250706406
C	5	87	10	11.278625052808277
C	6	87	10	13.65511399742252
C	7	87	10	14.490412059901923
C	8	87	10	14.628726620737421
C	1	187	10	0.0035871669646834192
C	2	187	10	0.17058251997290164
C	3	187	10	2.70463717000497
C	4	187	10	7.562490477796784
C	5	187	10	11.296209954929086
C	6	187	10	13.083683889229825
C	7	187	10	13.578574811874777
C	8	187	10	13.613191381304558
C	1	11	20	0.005257504383962441
C	2	11	20	0.1328026682875656
C	3	11	20	2.645469769828882
C	4	11	20	7.637184867247232
C	5	11	20	11.9503436289455
C	6	11	20	14.566524678120338
C	7	11	20	15.786440743543693
C	8	11	20	16.268007904068543
C	1	87	20	0.006435332990818665
C	2	87	20	0.6959842625990547
C	3	87	20	6.3716750385889185
C	4	87	20	12.237750569583373

trozo	bits	words	alphabet size	IM(LP,VD)
C	5	87	20	14.673394682137074
C	6	87	20	15.05844560118894
C	7	87	20	14.961597675367718
C	8	87	20	14.793223333549019
C	1	187	20	0.006898247478708264
C	2	187	20	1.0842701127646777
C	3	187	20	7.215229873872396
C	4	187	20	12.255891045618043
C	5	187	20	13.877530616433193
C	6	187	20	14.034483955093329
C	7	187	20	13.875688330654611
C	8	187	20	13.693293308041325
C	1	11	50	0.006208055597015871
C	2	11	50	1.2384320571645908
C	3	11	50	8.463614924696365
C	4	11	50	14.628385841768335
C	5	11	50	16.86687773936527
C	6	11	50	17.34927554153607
C	7	11	50	17.464702200138227
C	8	11	50	17.477040225545803
C	1	87	50	0.014035859612665291
C	2	87	50	3.3208460073086385
C	3	87	50	11.68470901812871
C	4	87	50	15.093791178005088
C	5	87	50	15.349300274054544
C	6	87	50	15.183218447931885
C	7	87	50	14.98489727147203
C	8	87	50	14.79800186709187
C	1	187	50	0.022431582741583966
C	2	187	50	4.63454577701857
C	3	187	50	12.63017991862702
C	4	187	50	14.500063268424503
C	5	187	50	14.360809627001219
C	6	187	50	14.109886483299473
C	7	187	50	13.88817244967031
C	8	187	50	13.695554557758664