Informe de Prácticas
Grupo de Neurociencia Computacional
Curso 2022/23

Tarea de Codificación de Dos Estímulos Diferentes

AUTOR:

Javier Luque Serrano

TUTOR PROFESIONAL:

Néstor Parga Carballeda

21 de junio de 2023

Índice

Abstract

Redes artificiales y cerebros reales pueden usar estrategiascomputacionales similares.
En este trabajo se ha entrenado una red neuronal recurrente de neuronas conun modelo de espigas, para lo cual se ha utilizado el algoritmo full-FORCE.
La tarea consiste en la codificación de dos estímulos diferentes(agrupado (A) y extendido (E)), de manera que la red pueda diferenciarlosen una posterior tarea. Para ello, se han utilizado dos targets, unatractor que define la longitud del estímulo y un bump cuya función esclasificar el estímulo A. Ha sido posible entrenar la red con las salidasdeseadas, con una gran precisión.

Palabras clave: Aprendizaje supervisado, codificación, conversión de código, redes neuronales recurrentes.

1. Introducción

1.1. Introducción teórica

1.1.1. Modelos computacionales

El campo de la Neurociencia Computacional surgió alrededor de 1990 y, gracias al trabajo de Hodgkin y Huxley, Lapicque et. al., el campo de estudio evolucionó desde un modelo matemático del potencial de acción, hasta la gran variedad de modelos de neuronas y redes neuronales que tenemos a día de hoy.
Actualmente existen varios modelos de neuronas, entre los que se encuentran el modelo de tasa de disparo (rate), o el modelo goteo-integración y disparo (LIF por sus siglas en inglés), siendo este un modelo biológicamente más realista que describe la evolución del potencial de membrana de una neurona mediante una ecuación diferencial ordinaria.

$$\tau_m \cdot \frac{dV_i(t)}{dt} = V_{rest} - V_i(t) + I_{ext} \tag{1}$$

donde τ_m es la constante de tiempo característico de la membrana e I_{ext} una corriente externa que se le aplica a la neurona.

Por otro lado, es importante hablar de los distintos tipos de redes neuronales, entre los que destacan la red neuronal directa (feedforward) y la recurrente (RNN). Este segundo tipo de red neuronal, de nuevo, se trata de un modelo biológicamente más realista, debido a que las conexiones entre neuronas pueden darse en cualquier dirección, pudiendo así mantener la información en memoria, lo que permite a la red trabajar con dicha información. Por el contrario, el modelo feedforward sólo permite conexiones con la unidad (neurona) postsináptica, de manera que se trata de un modelo más simple; nos centraremos en el modelo RNN.

Este modelo se puede combinar con los distintos tipos de neurona, dando lugar a diferentes formas de tratar una tarea concreta. Hay los distintos tipos de aprendizaje que pueden realizarse en una red neuronal, aprendizaje *supervisado*, *no supervisado* y *con refuerzo*.
De aquí surgen distintos algoritmos, en función de qué tipo de aprendizaje se quiere aplicar para la tarea en cuestión. Entre ellos, destacan algoritmos de aprendizaje supervisado que usan una RNN a la que se le introduce una señal de entrada F^{in}, y mediante unos pesos plásticos, genera una señal de salida $z(t)$ que se introduce de nuevo a la red como "target"para que aprenda, modificando los pesos de salida.
Siguiendo este esquema, una implementación exitosa de este tipo de algoritmos es el Full-FORCE, el cual puede utilizar como red principal una RNN de espigas, o sRNN (compuesta de neuronas LIF conectadas mediante pesos lentos (plásticos) y pesos rápidos) o una de tipo rate (cuyas unidades están todas conectadas mediante pesos plásticos) para ser entrenada mediante una RNN auxiliar con modelos de neuronas rate que genera un target y se implementa como señal de entrada a la red principal.

En todo este proceso se crea un error en la señal de salida que genera la red, que tiene que ser minimizado.

$$error(t) = z(t) - z_{tgt}(t) \tag{2}$$

La manera más adecuada de implementar esta minización en RNNs es Recursive Least Squares"(RLS)

1.1.2. Áreas funcionales de la corteza cerebral relevantes en este estudio

Es importante conocer bien cómo afecta un determinado estímulo a las diferentes áreas del cerebro, y cómo las redes neuronales los convierten en informes perceptivos mediante señales eléctricas.

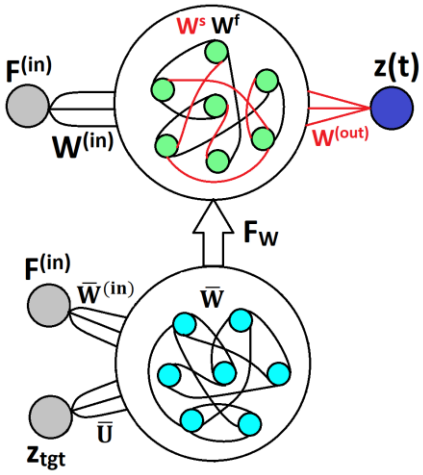


Figura 1: Esquema del algoritmo Full-FORCE, Figure 5 - Parga et. al. (2022) [3]

La corteza cerebral es muy compleja, con áreas muy diferenciadas para cada tarea neuronal. En este caso concreto, es de especial interés el Córtex Somatosensorial, encargado de procesar e informar cualquier estímulo sensorial, tales como el tacto, la temperatura, la propiocepción o la nocicepción.
Las neuronas del *córtex somatosensorial primario* (**S1**) muestran las respuestas sensoriales invariantes, es decir, en este área se representa fiel y homogéneamente la temporalidad de cada estímulo; mientras que en el lóbulo frontal, las neuronas exhiben respuestas complejas y heterogéneas asociadas con la memoria operativa (de trabajo) y los informes de percepción. Las señales que generan las redes neuronales de estas dos áreas corresponden a distintas etapas del proceso cognitivos, de manera que el *córtex somatosensorial secundario* (**S2**) actúa como intermediario y contiene ambos tipos de señales. Además, las señales que ofrece S2 pueden depender fuertemente del contexto de la tarea en cuestión, siendo en esta área donde comienza a verse signos de codificación de la memoria operativa, comparación e informes de decisión asociados a los estímulos.

Por último, dentro del lóbulo frontal se encuentra el *córtex premotor dorsal* (**DPC**), conectado directamente con la espina dorsal. La red neuronal de DPC causa señales nerviosas causan patrones mucho más complejos que los de S1, lo que lo hace un sistema mucho más sofisticado.

1.2. Métodos

Para la realización de esta tarea, se utilizó el software MATLAB para entrenar una sRNN mediante aprendizaje por refuerzo, usando el algoritmo full-FORCE; en la Figura 1 se puede ver un esquema del algoritmo usado. Además, para una mayor eficiencia en la ejecución de los códigos, se usó un cluster del grupo de Neurociencia Computacional (UAM).

¿En qué consiste la tarea sobre la que se ha hecho el estudio?
Esta tarea consiste en la codificación de dos estímulos diferentes, como se ven en la Figura 2 denominados *estímulo agrupado (A)* y *estímulo extendido (E)*. Este objetivo se consiguió entrenando una sRNN mediante la implementación de dos targets que la red tiene que replicar. Estos dos targets se tratan de, en primer lugar, un atractor que se usa para definir la duración de ambos estímulos, y en segundo lugar, un bump o vientre creado mediante una distribución beta que aparece en el centro del estímulo cuando éste se trata del agrupado, es decir, simula la agrupación de los tres pulsos centrales. Mediante estos dos targets y la ausencia del bump en el estímulo extendido, la red es capaz de codificar y diferenciar ambos estímulos.

Esta tarea concreta es innovadora en el ámbito computacional, pues es la primera vez que se desarrolla. Sin embargo, está basada y sus resultados serán comparados con un experimento que el Grupo de Neurociencia Computacional de la UAM hizo en colaboración con un laboratorio del IFC de la UNAM. [5].

Para llegar a los resultados finales, se hicieron varias adaptaciones en el algoritmo de entrenamiento hasta

conseguir resultados exitosos. En este sentido, exitoso significa que la red simula con la mayor exactitud posible los targets que se le imponen como refuerzo. En el Anexo I se exponen todas los resultados previos al definitivo.

1.2.1. Esquema de la tarea

En primer lugar, se mostrará el esquema que se ha tomado para el entrenamiento de la red, obtenido de la ejecución de un script independiente del código principal, llamado Trial_Codificacion., el cual ha servido para la comprobación de las distintas modificaciones antes de su implementación en el algoritmo.

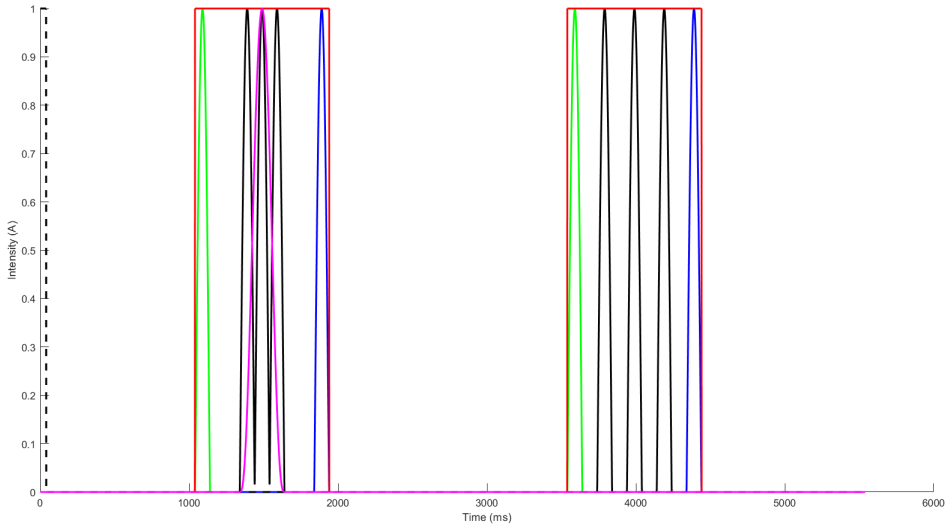


Figura 2: Esquema de un ensayo de clase 3 obtenido mediante el script independiente Trial_Codificacion.m.

Éste consiste en, por un lado, **cuatro señales de entrada o inputs**:

1. El probe down que marca el comienzo del ensayo, *en negro con líneas discontínuas*.
2. El primer pulso *en verde*, que determina el inicio del atractor.
3. La agrupación de los tres pulsos centrales, *en negro*, que instruye a la red para generar el bump, en caso del estímulo A; o no, en el caso del estímulo E.
4. El último o quinto pulso, *en azul*, el cual quita a la red del atractor.

Y por otro lado, **dos señales de salida o outputs**, que serán los dos targets que se han mencionado anteriormente; el atractor, *en rojo*, y el bump, *en magenta*.

Entonces, se implementará esta tarea en el algoritmo full-FORCE, que se desarrolla en tres etapas: una primera etapa llamada *"demean"* en la que se prepara a la red para empezar el entrenamiento, generando una corriente denominada *Imu*, utilizada para poner a la RNN en un régimen de disparo irregular. En una segunda etapa de entrenamiento o *"train"*, donde se usa la regla de aprendizaje RLS, utilizando una red de rates que genera el target y una LIF que realiza la tarea. Por último, la tercera etapa será de comprobación o *"test"*, donde se generan todos los archivos necesarios para el análisis de la actividad neuronal y su comportamiento.
Por último, y entre otros scripts auxiliares, se encuentra el script principal *demo_SN.m*, que será el que llamará a todas las anteriores funciones y le dará valor a todas las variables de entrada necesarias.

Ahora bien, **¿cómo sabe la red si un ensayo ha sido correcto o no?**
Para ello sólo hay que comparar la corriente de salida de la red con el target. Es decir, un ensayo es considerado correcto si el output de la red $\mathbf{z(t)}$ y el target $\mathbf{F^{out}(t)}$ cumplen la siguiente condición durante la presentación del target [Ingrosso et. al., 2019]:

$$\frac{\sum_t z(t)F^{out}(t)}{\sqrt{\sum_t z(t)^2}\sqrt{\sum_t (F^{out}(t))^2}} > 0{,}5 \tag{3}$$

1.3. Objetivos

El objetivo principal de esta tarea es que la red consiga generar de la forma más precisa posible los dos targets que se disponen (atractor y bump). Una vez conseguido esto, se deberá hacer un análisis comparativo con los experimentos [4] y [5] para entender qué similitudes puede tener esta red neuronal computacional con las áreas involucradas en la percepción de los estímulos sensoriales que se le presentan a un animal.
Además, para una correcta completitud de la tarea, la red deberá comparar y diferenciar los estímulos con exactitud.

2. Resultados

Lo principal a la hora de entender la tarea que vamos a tratar es conocer cuántas variantes pueden existir en un mismo entrenamiento, a lo cual llamaremos çlase". Teniendo en cuenta que son dos los tipos de estímulos que se presentan en la tarea, son **cuatro** las permutaciones o **clases** que tenemos. Por orden, éstas clases serán **E-E, E-A, A-E y A-A**, y se usarán todas durante todas las etapas del entrenamiento.

2.1. Desempeño de la red

Partiendo de la ecuación 3, a continuación se mostrará el archivo *Accuracy.txt* que se generó durante el entrenamiento, en el que se expone el número y porcentaje de ensayos correctos por cada clase:

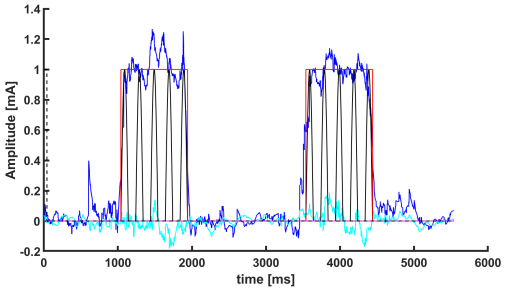
Cuadro 1: Proporción de ensayos correctos por clase.

	Nº de ensayos correctos	Nº total de ensayos	% de ensayos correctos
Clase 1	1063	1063	100
Clase 2	996	996	100
Clase 3	972	972	100
Clase 4	969	969	100

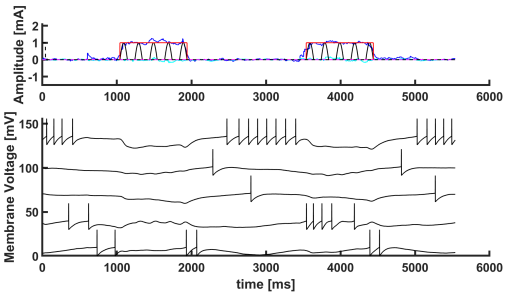
2.2. Propiedades de la red entrenada

Es importante destacar los resultados que se obtuvieron con el entrenamiento, los cuales expondremos y analizaremos a lo largo de este apartado. Tanto para los ensayos de entrenamiento como para los de test, se obtienen 3 figuras:
Una primera que muestra un esquema con todo lo incluido en la Figura 2, a lo que se le añade la respuesta de la red a los dos targets, $z_s(1) \equiv$ atractor y $z_s(2) \equiv$ bump (Figuras 3a, 4a, 5a y 6a).
Una segunda figura con dos paneles: en el primero se muestra de nuevo el esquema para hacer una comparación de los tiempos, y debajo de éste, el potencial de membrana $V_i(t)$ (Figuras 3b, 4b, 5b y 6b).
Por último, la última figura muestra de nuevo dos paneles: en el superior se exhibe el esquema de la tarea y la tasas de disparo media $\bar{r}(t)$, mientras que en el inferior se muestra un ráster de la red durante todo el ensayo (Figuras 3c, 4c, 5c y 6c).

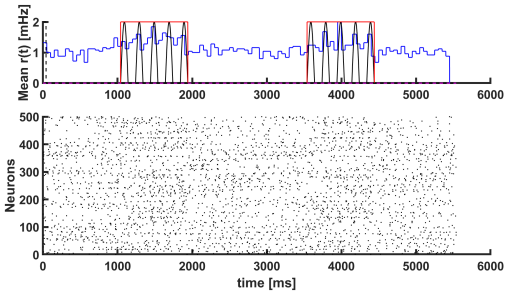
Por tanto, con el objetivo de exponer el aprendizaje de la red, se mostrarán las tres figuras obtenidas tras la ejecución del código, divididas por clase, siempre de la etapa test, ya que son resultados más fieles al entrenamiento:



(a) Esquema de un ensayo de clase 1. En azul se dispone la respuesta de la red al atractor y en cian la respuesta al bump.

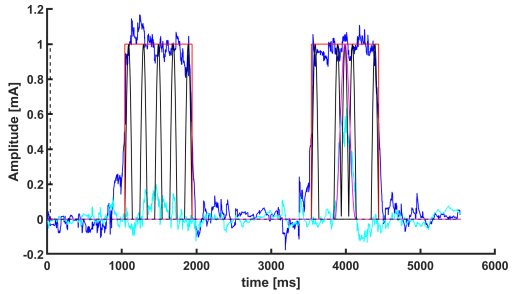


(b) Potencial de membrana de cinco neuronas arbitrarias con el esquema del ensayo de clase 1 para su comparación temporal.

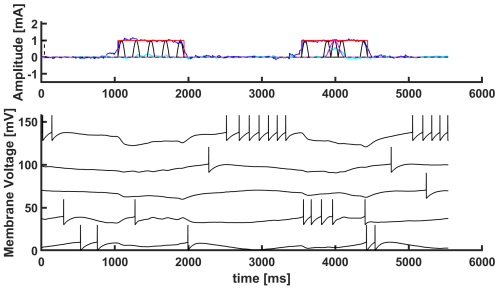


(c) Ráster de toda la población de neuronas junto a la tasa de disparo media a lo largo del ensayo de clase 1.

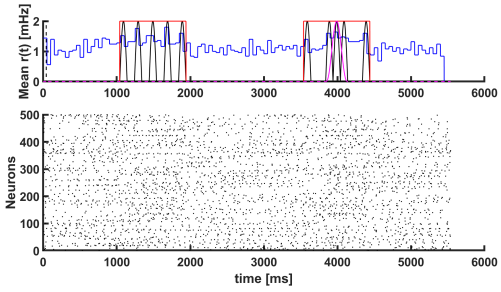
Figura 3: Clase 1



(a) Esquema de un ensayo de clase 2. En azul se dispone la respuesta de la red al atractor y en cian la respuesta al bump.

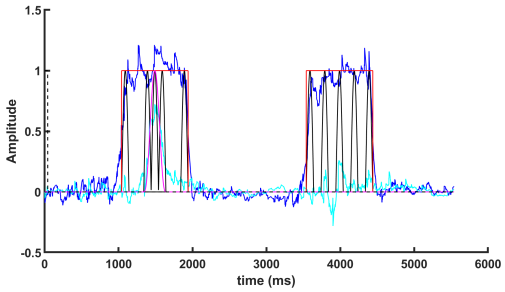


(b) Potencial de membrana de cinco neuronas arbitrarias con el esquema del ensayo de clase 2 para su comparación temporal.

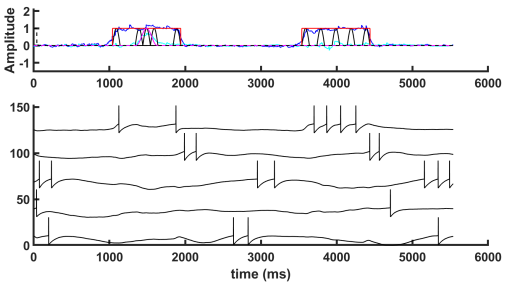


(c) Ráster de toda la población de neuronas junto a la tasa de disparo media a lo largo del ensayo de clase 2.

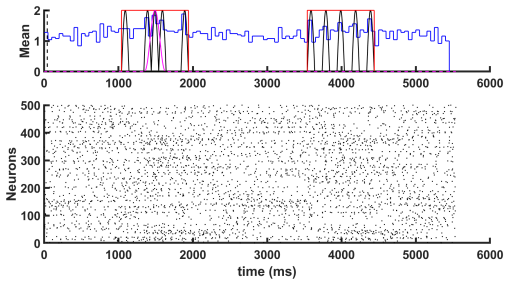
Figura 4: Clase 2



(a) Esquema de un ensayo de clase 3. En azul se dispone la respuesta de la red al atractor y en cian la respuesta al bump.

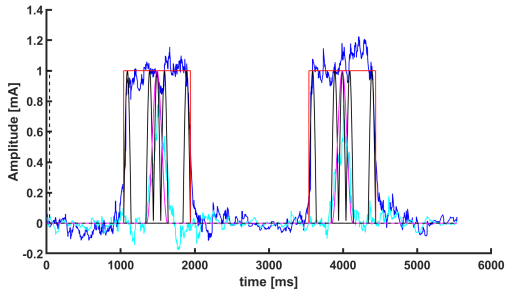


(b) Potencial de membrana de cinco neuronas arbitrarias con el esquema del ensayo de clase 3 para su comparación temporal.

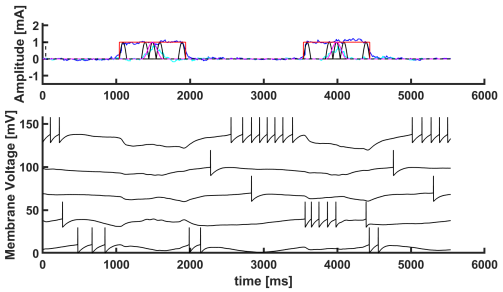


(c) Ráster de toda la población de neuronas junto a la tasa de disparo media a lo largo del ensayo de clase 3.

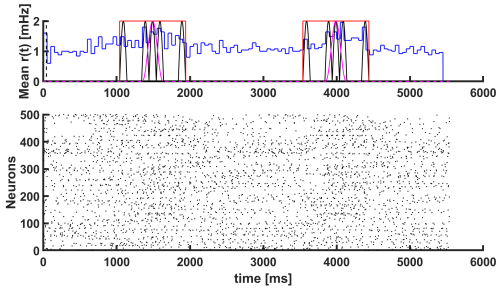
Figura 5: Clase 3



(a) Esquema de un ensayo de clase 4. En azul se dispone la respuesta de la red al atractor y en cian la respuesta al bump.



(b) Potencial de membrana de cinco neuronas arbitrarias con el esquema del ensayo de clase 4 para su comparación temporal.



(c) Ráster de toda la población de neuronas junto a la tasa de disparo media a lo largo del ensayo de clase 4.

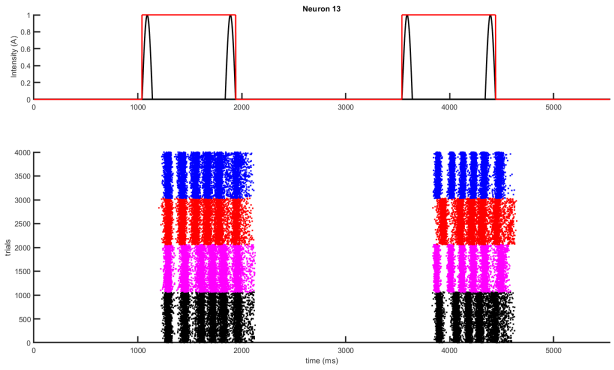
Figura 6: Clase 4

Por otro lado, es también muy importante analizar cómo se comporta cada neurona de forma individual para dar como resultado los vistos arriba. Esto lo haremos mediante el cálculo y representación de la tasa de disparo media

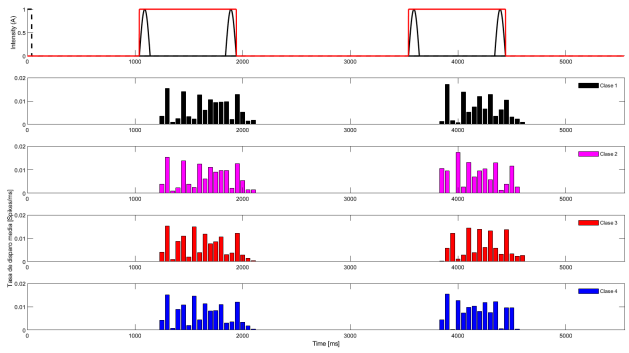
en forma de histograma y un ráster, ambos para cada clase. Estas figuras se representarán debajo del esquema de la tarea para entender, en función del tiempo y de los estímulos, cómo actúa la neurona.
Cada unidad se comporta de manera muy diferente ante una clase concreta, de manera que cada una de ellas codifica la información de forma totalmente diferente, pero la comunicación entre ellas da, finalmente, un resultado que, como podemos ver, se acerca mucho al objetivo inicial.

La población de neuronas con la que la red ha sido entrenada es de $N = 500$, sin embargo, no es posible analizar todas y cada una de ellas, por lo que analizaremos más en profundidad dos de ellas, que parecen tener un mayor interés (neuronas 13 y 19).
Aclarar dos cosas: en ambas figuras, los colores correspondientes a cada clase son los mismos, estos se disponen en la leyenda del histograma; y en cuanto a los ejes, las Figuras 7a y 8a son Ïntensidad [mA]"para el panel de arriba, y "trials"(ensayos) para el panel de abajo en la ordenada, "time [ms]"para la abcisa. En las Figuras 7b y 8b, el eje y corresponde con "Spikes/ms"(acciones de potencial/ms).

Se recuerda que las clases, en orden ascendentes, son E-E, E-A, A-E y A-A.

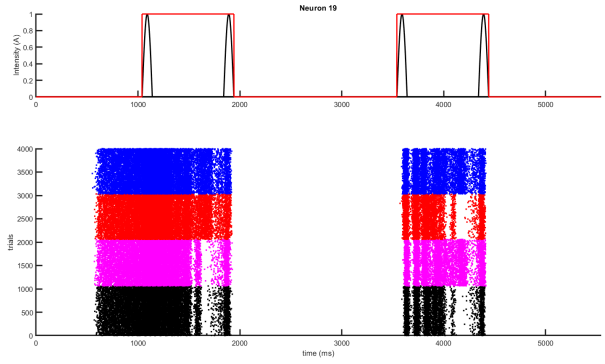


(a) Ráster

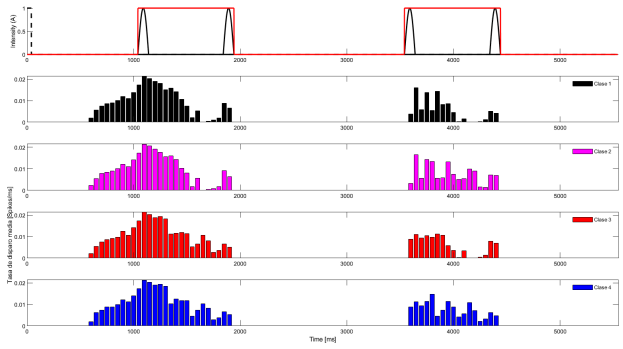


(b) Histograma: tasa de disparo

Figura 7: Neurona 13



(a) Ráster



(b) Histograma: tasa de disparo

Figura 8: Neurona 19

3. Discusión

3.1. Precisión del entrenamiento

En primer lugar, como podemos ver en el fichero Accuracy.txt de la Tabla 1, el porcentaje de ensayos correctos para las cuatro clases es del 100 %. Esto es algo que, a priori, nos dice que la tarea ha sido bien entrenada, lo que no implica necesariamente que la respuesta de la red haya sido acertada.
También hemos conseguido que todos los ensayos sean correctos debido a que la tarea es sencilla, con un bajo número de clases, lo que da lugar a un bajo rango de error.

3.2. Salida de la red

Observando las Figuras 3a, 4a, 5a y 6a, es posible observar que la respuesta de la red a los targets es muy precisa y los replica con gran exactitud. Aún así, sigue habiendo algo de ruido, nada que no se pueda arreglar.

En las Figuras 3b, 4b, 5b y 6b vemos como ninguna de las 5 unidades se comportan de igual manera para una misma clase, algo que se debe al comportamiento estocástico de las neuronas, de manera que cada una codifica los estímulos de una manera diferente. Sin embargo, debido al sistema complejo que forman el conjunto de todas ellas, puede verse en el panel superior que da una respuesta precisa.

Por último, las Figuras 3c, 4c, 5c y 6c muestran, en el panel superior, una tasa de disparo media que tiene mucho sentido, ya que oscilan entorno a su valor medio, y aumenta ligeramente cuando aparece un target, lo cual se puede ver reflejado en los rásters del panel inferior. (Para que sea más visible, es recomendable alejarse, ya que la lejanía de la imagen crea un efecto visual que oscurece los instantes en los que las neuronas disparaban más acciones de potencia.)
Esta tendencia de las neuronas a disparar más espigas durante los estímulos, se hace más presente si éstos son iguales, lo cual tiene sentido ya que los codificarán siguiendo un mismo patrón.

3.3. Esquemas de una única neurona

En este apartado se compararán los resultados con los de Rossi-Pool et. al., (2021)[4] para comprobar si el comportamiento de las neuronas que componen esta red corresponden con la área S2. Esta conclusión se desmiente inmediatamente, ya que los rásters no se parecen en nada a los de [4], por lo que no podemos asegurar a qué área del córtex pertenecen las neuronas que componen esta red.
Por otro lado, es posible notar que en casi todas las neuronas aparecen seis pulsos en lugar de los cinco que tiene el estímulo. Esto se puede deber a que cinco son para identificar la cadena de pulsos y el sexto para codificarlo.

3.4. Posibles mejoras

A pesar del éxito del entrenamiento de esta red, algo que terminaría de concluir el trabajo y lo mejoraría enormemente sería la implementación de un XOR temporal que se encargase de diferenciar si los dos estímulos dispuestos en un ensayo concreto son iguales o diferentes. Esto se podría hacer fácilmente gracias a que este código permite su implementación. Un posible esquema sería, por ejemplo, disponer después de un probe up, un cajón con signo positivo si las dos cadenas de pulsos son iguales, y por el contrario, un cajón con signo negativo si son diferentes.
La ejecución de este XOR temporal se tiene que corresponder con los resultados obtenidos en el experimento de [5], puesto que se trata de la visión empírica de esta tarea.

Otra posible manera de mejorar la tarea sería mejorar la precisión con la que la red replica los targets que se le disponen. Esto quizás podría hacerse si en lugar de un escalón, como atractor se le define una función sigmoidal que sube y baja al notar el primer y último pulso. De esta forma el target subiría de forma más suave y la red podría identificarlo con más exactitud. Además, si se aumentan el número de ensayos de entrenamiento, muy posiblemente también mejoraría la respuesta de la red.

4. Conclusiones

En general, los resultados obtenidos han sido muy positivos, dando lugar a una tarea de codificación de dos estímulos diferentes muy exitosa.
No se puede saber qué área del córtex sería la encargada de codificar los estímulos, no siendo éste el objetivo principal de la tarea, por lo que no supone una conclusión negativa.

El objetivo principal de este estudio está completo: la codificación de los targets, que produciendo bump (A) o no (E), sugiere una implementación directa de un XOR en una segunda red.

5. Bibliografía

1. Dayan and Abbott. (2001) Computational and Mathematical Modeling of Neural Systems. Capítulos 1 y 7, Sección 5.4, Apéndice 5.11.

2. DePascuale, Churchill and Abbott. (2016) Using Firing-Rate Dynamics to Train Recurrent Networks of Spiking Model Neurons, arXiv:1601.07620 [q-bio.NC].

3. Parga, Serrano and Falcó. (2022) Emergent Computations in Trained Artificial Neural Networks and Real Brains, arXiv:2212.04938v2 [q-bio.NC].

4. Rossi-Pool et.al. (2021) A continuum of invariant sensory and behavioralcontext perceptual coding in secondary somatosensory cortex. *Nature Communications*, 12:2000.

5. Rossi-Pool et.al. (2016) Emergence of an abstract categorical code enabling the discrimination of temporally structured tactile stimuli. PNAS E7966–E7975.

6. https://en.wikipedia.org/wiki/Computational_neuroscience

Agradecimientos

Debo agradecer enormemente a Luis Serrano Fernandez, quien ha actuado como mi mentor, por su completa disposición y la gran ayuda prestada a lo largo de todo el trabajo de prácticas.
Asimismo, he de agredecer también a Néstor Parga Carballeda por su interés al ofrecerme esta oportunidad de prácticas en el Grupo de Neurociencia Computacional; así como a Miguel Ángel Sánchez-Conde por prestarme toda la información necesaria para completar con éxito este periodo.

6. Anexos

Anexo I: Evolución de la tarea

En cada modificación que se hizo al código, con intención de obtener los resultados más exitosos posible, se hicieron ejecuciones modificando distintos parámetros para comprobar su relevancia, bien como el número de ensayos de entrenamiento ($TRLS$) o de test ($Ttest$), algunos tiempos como el delay entre estímulos ($inter\_stim$) o el tiempo para terminar el ensayo después del segundo estímulo ($delay$); así como la semilla que le da identidad a la red ($randseed$).

Por orden se expondrán otras posibles conversiones en código de los estímulos. Es importante aclarar que las figuras que se exponen corresponden a los ensayos con un resultado más optimista, además de que se han elegido aquellos ensayos asociados a las clases 2 y 3, para que se vea cómo actúa la red ante los dos tipos de estímulos.

1. Primero se trató de mantener los dos estímulos en memoria y, después de un probe up (final del ensayo), hacer una conversión del estímulo agrupado y extendido a un pulso largo y corto, respectivamente, usando una distribución beta. Los resultados, como se ve en la Figura 9, no fueron muy exitosos.

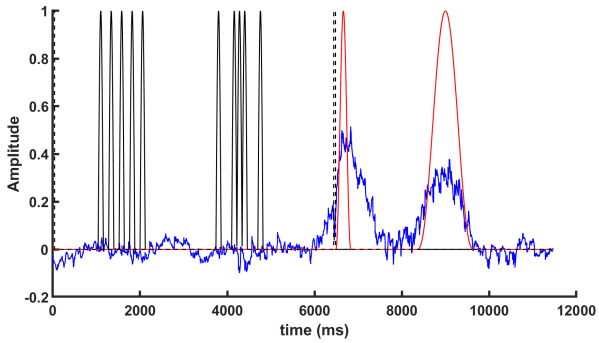


Figura 9: Distribución beta después del probe up.

2. A continuación, dejando los tiempos del esquema intactos, se cambió el target por un escalón, para comprobar si el error estaba en la distribución beta. En la Figura 10 se puede comprobar que el resultado es ligeramente más aproximado, pero sigue sin ser exitoso.

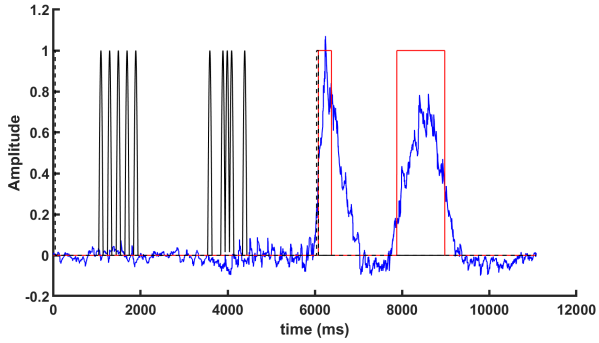


Figura 10: Función escalón después del probe up.

3. Finalmente, para una tercera modificación del código, se eliminó el probe down, disminuyendo la duración del ensayo, y se introdujeron los targets dentro de los estímulos. En este caso, los targets, en forma de escalón, trataban de diferenciar ambos estímulos codificando la duración entre el primer y segundo pulso de cada uno. En la Figura 11 se ve que los resultados han mejorado bastante, muy seguramente debido a que la red ya no tiene que mantener en memoria los dos estímulos para luego hacer la conversión del target y ser entrenada. Aún así, sigue habiendo mucho ruido.

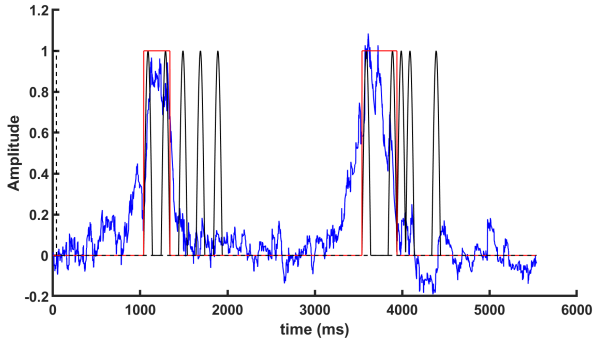


Figura 11: Función escalón entre pulsos 1 y 2.