16.82-Neurociencia Computacional
1C 2024

Trabajo Práctico 2:

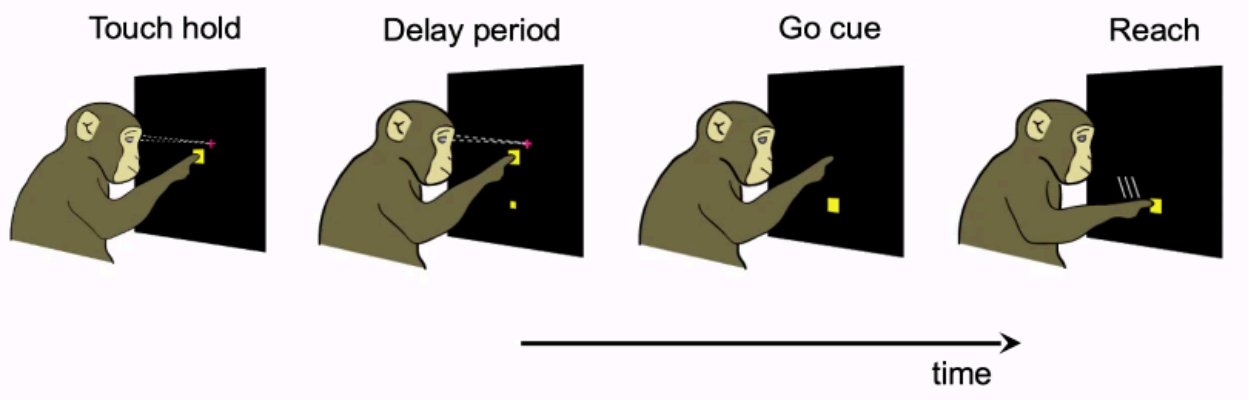
Adaptado de ***Intro to Neuroengineering*** del Dr. Jacob Robinson y Dr. Caleb Kemere.

Otros trabajos utilizando estos datos (como inspiración para proyecto final):

- Model-Based Neural Decoding of Reaching Movements: A Maximum Likelihood Approach
- Model-Based Decoding of Reaching Movements for Prosthetic Systems

Tenemos datos de actividad neuronal medida con Neuropixels de un mono realizando la tarea motriz que vimos en la clase de Reducción de Dimensionalidad y Redes Neuronales y resumida acá. La tarea que realiza el mono es una "tarea de alcanze retrasado" (delayed reach task), donde cada episodio consiste de las siguientes cuatro fases:

1. Mantener el dedo sostenido en el centro y los ojos fijos en una cruz
2. Aparece una marquita (target) en la pantalla indicandole al mono donde deberá mover el dedo. El mono debe quedarse mirando el punto fijo y no debe mover el dedo de lugar (esta es la parte de "retraso")
3. Desaparece el cuadrado y la cruz de la pantalla y se agranda la marca a donde debe mover el dedo. Eso le indica al mono que ya puede realizar el movimiento. El tiempo de espera entre que aparece la marca y se le da la señal está aleatorizado, tal que el mono no pueda adelantarse.
4. Cuando el mono procesa que se le dió la señal (que tendrá cierto tiempo de reacción), el mono finalmente realiza la acción motriz.



Los targets aparecen en una de 8 posiciones. Su tarea será, utilizando estos datos, poder predecir en qué dirección el mono movió el brazo a partir de la actividad neuronal, analizando también qué partes de la actividad son útiles para la decodificación. Utilizará el ***periodo de planificación*** del mono, comprendido entre el 2 y el 3 que se mencionó arriba (es decir, desde que se le indica al mono que tiene que mover su brazo en cierta dirección, pero antes de que comience el movimiento).

Ustedes contaran con un código esqueleto (**GuiaTP2.ipynb**), los datos de los disparos (**datos_disparos_mono_tp2.npz**) y los metadatos de la actividad (tiempos y targets de cada episodio) (**metadata_mono_tp2.npy**)

16.82-Neurociencia Computacional
1C 2024

Ejercicios:

1. *Precisión en la decodificación y la ventana de planificación.*
   El primer set de análisis que tienen que realizar es cuantificar el efecto de la duración y posición de la ventana de planificación en la precisión del clasificador. El código de ayuda calcula la precisión para los episodios de ventana corta (755ms). “Porcentaje correcto”. Si corre este código muchas veces verá que el resultado cambia ligeramente porque estamos eligiendo episodios para entrenar al azar.
   a. **(15%)** Modificar el código para usar los primeros 750 ms de la ventana de planificación tanto para los episodios largos y cortos, cambiando el renglón ***plan_spikes = extract_plan_spikes()*** por ***plan_spikes = extract_plan_spikes(window_length=750)*** y ***target_trials = np.argwhere(short_trials & (trial_reach_target==c)).squeeze()*** a ***np.argwhere((trial_reach_target==c)).squeeze()*** (sin el “*short_trials*”). Ahora tenemos el mismo número de episodios de entrenamiento pero muchos más de testeo. Cómo cambió la precisión?
   b. **(15%)** Debajo de la celda de “Porcentaje correcto” hay esqueleto de código para evaluar el rendimiento como una función de la longitud de ventana de planificación. Complete este código (pueden reciclar mucho de lo de arriba), corra la figura e interprétela. ¿Qué duración de ventana de planificación es necesaria para un rendimiento razonable en la decodificación? (Debería darles una figura parecida a la guardada en el notebook).
   c. **(10%)** Ahora ajuste la duración de la ventana de decodificación a 250ms y trate de empezar la ventana a distintos tiempos pasando el argumento de ***start_offset=offset*** a la función ***extract_plan_spikes(),*** donde esa variable *offset* varía de 0 a 500 dentro de un bucle (500 es lo máximo que entra para ventanas de planificación de 755ms!). Interprete los resultados. ¿En qué momento aproximadamente aparece la actividad neuronal más útil para decodificar?
2. *¿Cómo afecta el rendimiento la cantidad de neuronas que puedo medir?*
   a. **(20%)** Modifica el código del Problema 1 para elegir un subconjunto de neuronas aleatoriamente para utilizar para decodificar. Usando una ventana de planificación de 250ms con 100ms de desfasaje, evalúa la precisión promedio del decodificador en función del número de neuronas utilizado. Toma grupos de 30, 60, 90, 120 y 150 neuronas, muestreando por lo menos 25 veces por tamaño de grupo. Aproximadamente cuantas neuronas necesitan para buen rendimiento?**Ayudita:** Pueden usar *np.random.choice()* para conseguir índices de neuronas al azar. Recuerden poner *replace=False* para que no repita neuronas.
   b. **(20%)** Cómo varía el rendimiento si toman ***i)*** las 30 neuronas con ***mayor*** tasa de disparo promedio ***ii)*** las 30 neuronas con ***menor*** tasa de disparo promedio ***iii)*** 30 neuronas al azar (muestreen muchas veces).
   c. **(10%) Puntos extra:** ¿Cuál es el mínimo número de neuronas que dan buen rendimiento? ¿Cómo las puede obtener?
3. *¿Cómo afecta el rendimiento la cantidad de datos de entrenamiento?*

a. **(20%)** Modifique el código del problema 1 para testear el efecto de cambiar el tamaño del set de entrenamiento, usando una ventana de 250ms con 100ms de desfasaje. ¿Cómo cambia la precisión del decodificador con 5, 10, 15, 20, 25, 30, 35 y 40 muestras por target?