Trabajo Práctico 2 - Aprendizaje por Refuerzos

Rodrigo Negri, Sergio Romano

21 de noviembre de 2016

1. Introducción

El objetivo de este trabajo práctico es construir un agente para el juego de Cuatro en Linea (o Conecta Cuatro) utilizando las técnicas de Aprendizaje por Refuerzos vistas en clases y estudiar su comportamiento.

2. Método

2.1. Q-Learning

El agente construido está basado en el algoritmo de Q-Learning [1] visto en clase, un algoritmo de Aprendizaje por Refuerzos para la construcción de un agente *model-free* que utiliza los refuerzos obtenidos para estimar la función de estado-valor $Q(s,a) : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{N}$ que representa la máxima ganancia esperada a partir del estado $s \in \mathbb{S}$ si se ejecuta la acción $a \in \mathbb{A}$, y siguiendo la política óptima en función de la estimación de Q en cada paso. El código del agente se encuentra en `https://gist.github.com/sromano/dc4e0ab78f9834b9b58b06020f9d48b4`

2.2. Evaluación del agente

Para los resultados, compararemos la performance del agente de Q-Learning con otro que llamaremos Random, el cual seleccionará aleatoriamente (con una distribución uniforme) cuál acción llevar a cabo dentro de las posibles para cada jugada de la partida. Cuando no se aclare lo contrario, el algoritmo de Q-Learning utilizará una política ϵ-greedy con $\epsilon = 0{,}2$, una tasa de aprendizaje $\alpha = 0{,}3$, un factor de descuento $\gamma = 0{,}9$, los valores de la función estado-valor $Q(s,a)$ inicializados en 1, y los refuerzos por terminar una partida en 1, 0.5 y -1 si el agente ganó, empató o perdió respectivamente.
En los experimentos, evaluaremos la evolución del aprendizaje entre ambos algoritmos en distintas condiciones, los efectos de contar con entrenamiento previo, de inicializar la función de estado-valor con distintas variantes, de alternar la tasa de aprendizaje (α) y de distintas políticas de explotación-exploración: ϵ-greedy y softmax.

3. Resultados

En la figura 3.1 podemos ver la evolución de un agente de Q-Learning contra el agente Random, quien en menos de 20.000 juegos ya empieza a sacar una ventaja considerable respecto del otro. La figura muestra el promedio de aciertos cada 100 juegos, moviendo la ventana 20 pasos en cada punto de la figura. Podemos notar que no alcanza un 100 % de efectividad debido a la dificultad intrínseca de seleccionar una estrategia siempre ganadora en el juego y a que el algoritmo de Q-Learing utiliza una estrategia ϵ-greedy para continuar explorando. Por tal motivo, en la figura 3.2 podemos ver el porcentaje de aciertos en 1000 partidas después del entrenamiento donde $\epsilon = 0$ para evitar exploraciones aleatorias. También hemos realizado una prueba llevando el tablero al doble de su tamaño (12 filas y 14 columnas), y en tal caso, el algoritmo de Q-Learning no pudo obtener una ventaja significativa del algoritmo Random en menos de 200.000 ciclos de entrenamiento debido al aumento en la dificultad de exploración de los estados del juego.

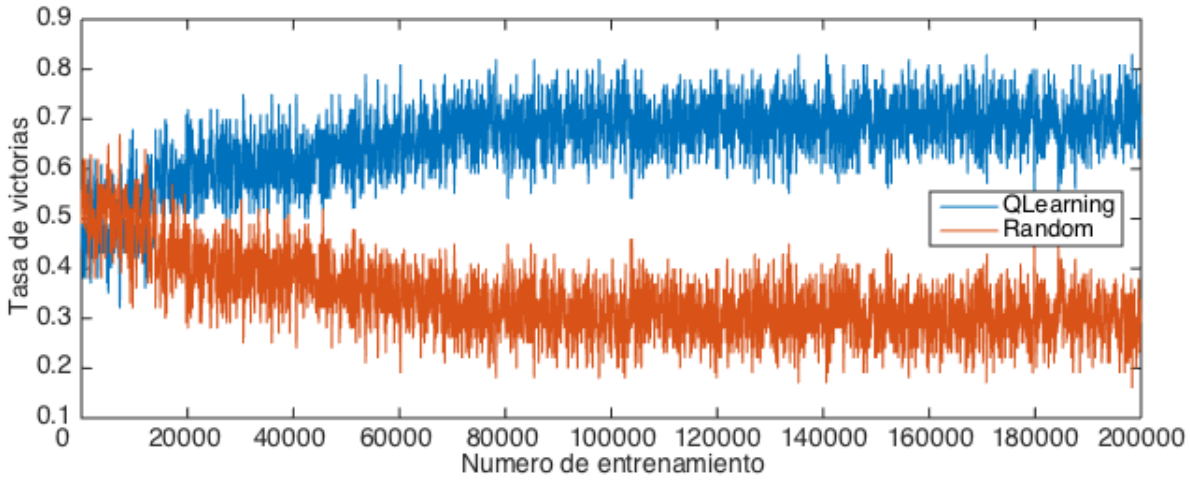

Figura 3.1: Q-Learning vs Random

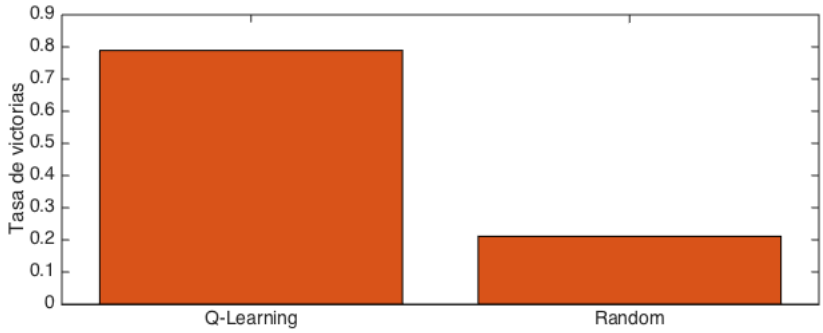

Figura 3.2: Evaluación después de entrenamiento: Q-Learning vs Random

En la figura 3.3 podemos ver la evolución de dos agentes de Q-Learning compitiendo entre sí. Notemos que aquí ninguno puede sacar una ventaja considerable sobre el otro. Aquí también la figura muestra el promedio de aciertos cada 100 juegos, moviendo la ventana 20 pasos en cada punto de la figura. Y volvemos a verificar, como se ve en la figura 3.4, que la diferencia en las últimas 1000 jugados tampoco es significativa cuando no se permite seguir explorando ($\epsilon = 0$) a ninguno de los dos agentes.
Por otro lado, en la figura 3.5 vemos un agente de Q-Learning sin entrenamiento contra otro pre-entrenado en 200.000 partidas (las primeras 100.000 contra un agente Random y las otras 100.000 contra un agente de Q-Learning). Allí podemos notar cómo en las primeras jugadas, la experiencia previa le permite triunfar sobre el otro hasta que a partir de la jugada 20.000 logra llegar al mismo nivel de conocimiento sobre el juego.

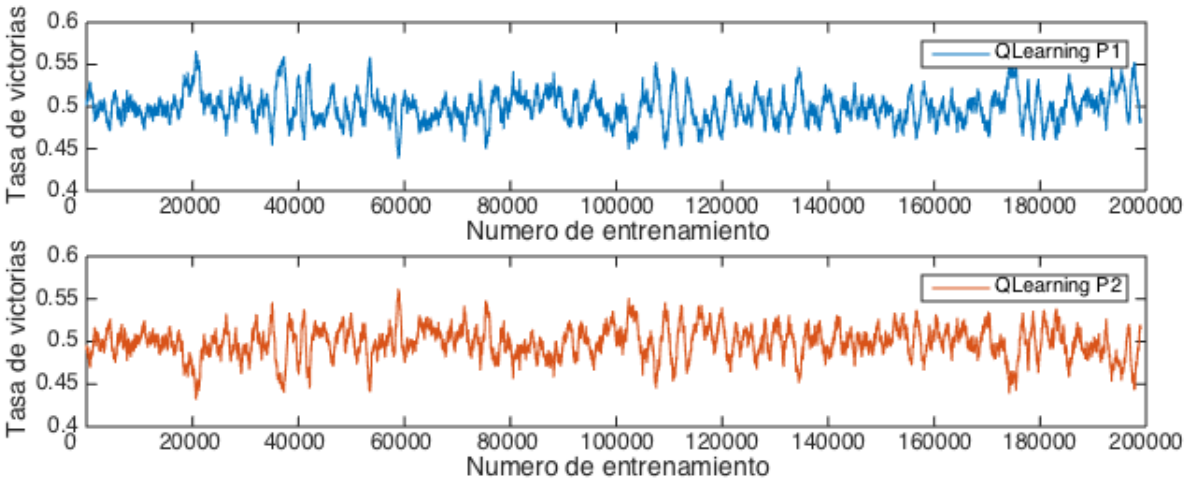

Figura 3.3: Q-Learning vs Q-Learning

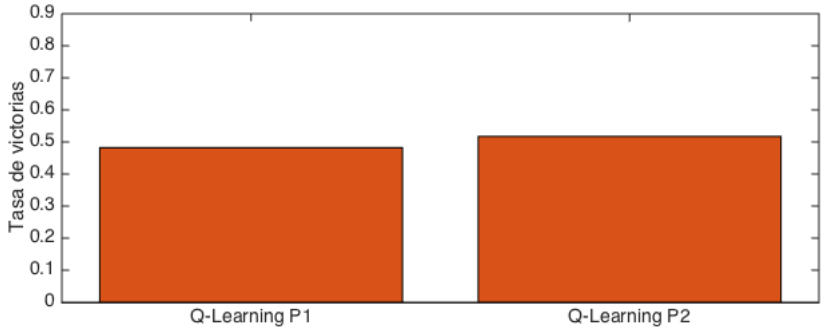

Figura 3.4: Evaluación después de entrenamiento: Q-Learning vs Q-Learnings

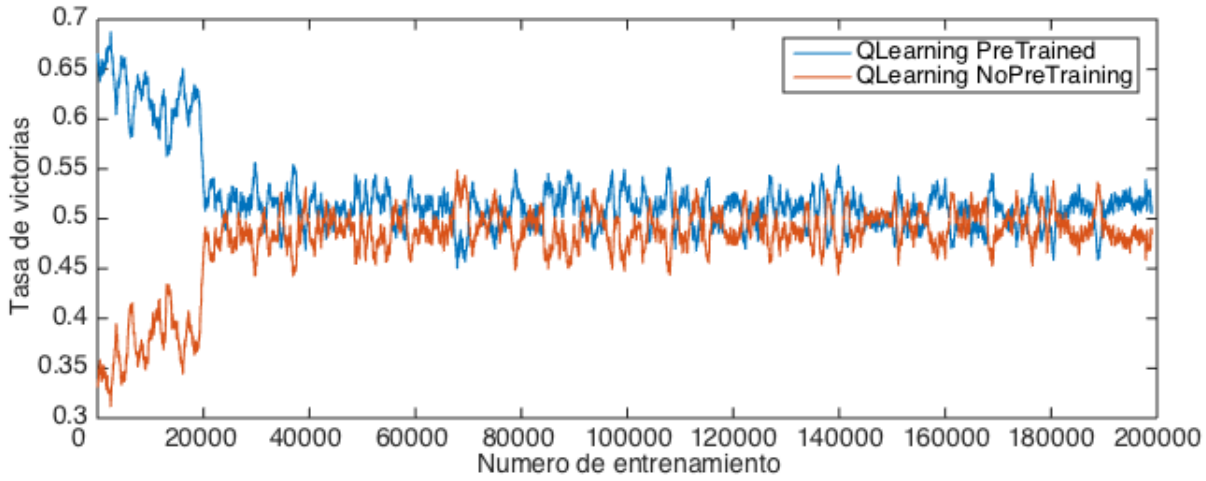

Figura 3.5: Efecto de pre-entrenamiento en Q-Learning

La figura 3.6 muestra los efectos de inicializar la función de estado-valor $Q(s, a)$ con distintos valores. En todos los casos, la tasa de victorias está calculada respecto de un agente Random y promediando las últimas 500 partidas (con una ventana de 500 partidas entre promedios). Los dos mejores casos son cuando empieza inicializada en 0 o en 1, y el peor cuando empieza en 10. Esto se debe a que el refuerzo de una victoria lo hemos definido como 1. Por lo tanto, si cada estado por defecto tiene un refuerzo de 10 (mucho mayor al de la victoria de una partida) el agente tardará mucho tiempo en converger a los valores adecuados para identificar qué estados lo llevan a una victoria debido a este desbalance. Por otra parte, la evolución de un agente cuyos estados están inicializado al azar (con una uniforme entre 0 y 1), no hace fácil la convergencia a una buena solución. Esto también lo podemos verificar en la etapa de evaluación (figura 3.7) como así también en la figura 3.8 donde compiten dos agentes de Q-Learning, uno con sus estados inicializado en 1 y el otro al azar.

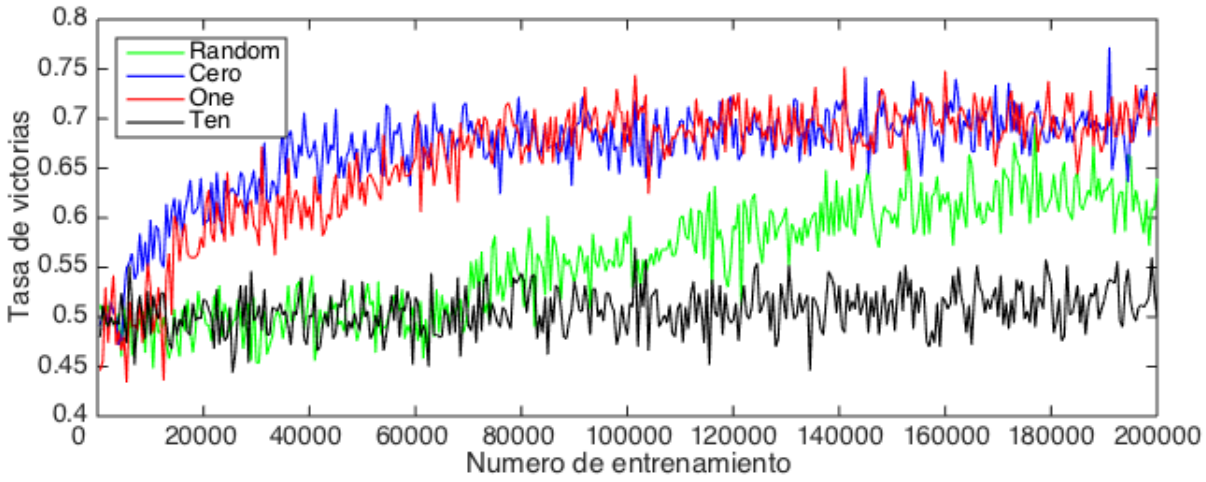

Figura 3.6: Inicialización de $Q(s, a)$

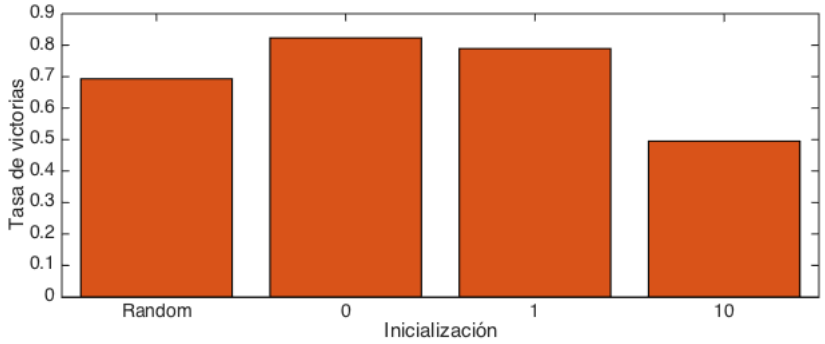

Figura 3.7: Evaluación después de entrenamiento: Inicialización de $Q(s, a)$

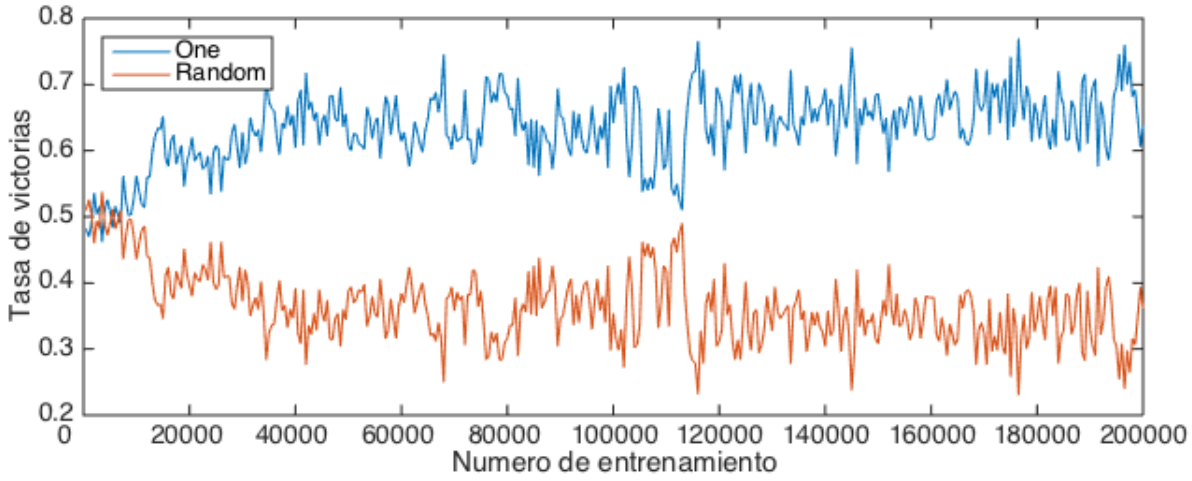

Figura 3.8: Q-Learning (Inicializado en uno) vs Q-Learning (Inicialización Random)

La figura 3.9 muestra distintos corridas de un agente de Q-Learning contra un agente Random promediando los aciertos cada 500 jugadas y variando la tasa de aprendizaje α que regula cómo se actualiza la experiencia recibida en la función Q en cada nuevo estado. Podemos notar que los valores en los límites del dominio, cercano a cero e igual uno, tienen una mala performance ya que en la primera casi no se actualiza con la experiencia y en la segunda el valor recibido pesa demasiado respecto de la experiencia previa. Para el caso de valores más altos (0.3, 0.6 y 0.9) el aprendizaje parece más rápido porque rápidamente la nueva experiencia de una victoria hace pesar esa estrategia. En cambio, cuando $\alpha = 0{,}1$ el aprendizaje para obtener esa tasa de victorias requiere de más jugadas. Sin embargo, una vez que todos los algoritmos terminaron el entrenamiento de 200.000 jugadas, en la etapa de evaluación, las cuatro variantes que no están en los bordes del dominio obtienen un resultado similar como puede verse en la figura 3.10.

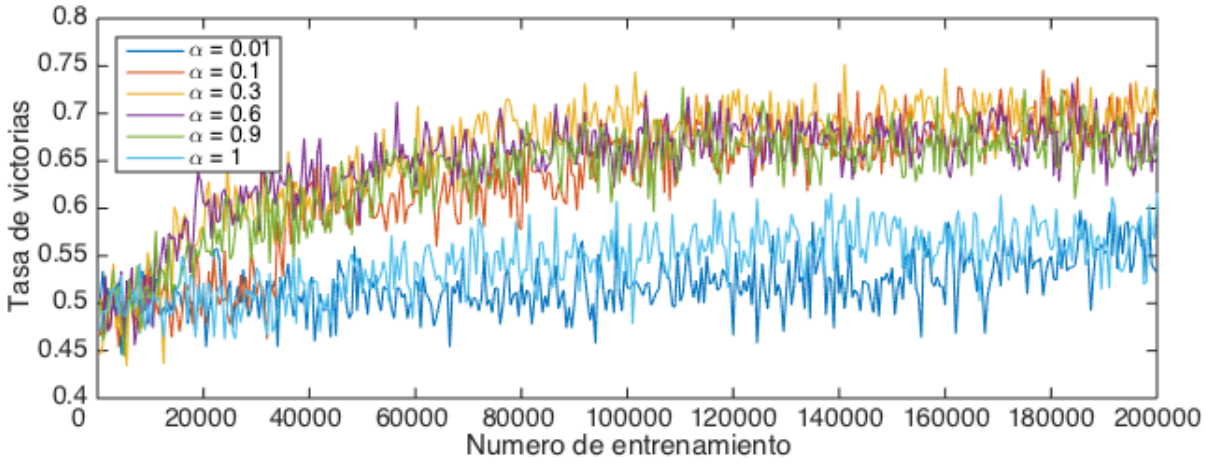

Figura 3.9: Tasa de Aprendizaje (α)

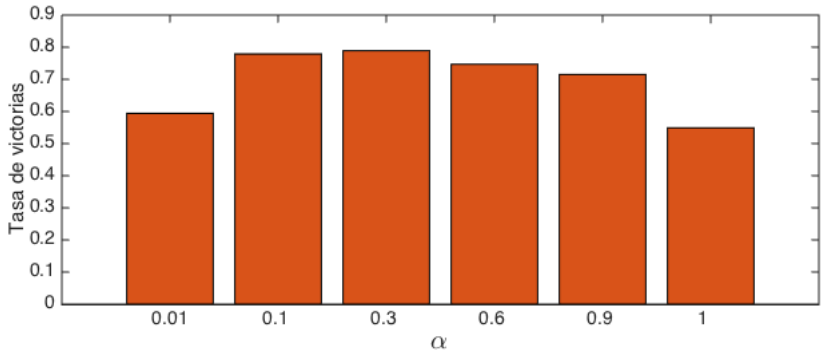

Figura 3.10: Evaluación después de entrenamiento: Tasa de Aprendizaje (α)

En la figura 3.11 puede observarse los efectos de variar la tasa de exploración ϵ de la política ϵ-greedy. Lo primero que podemos destacar es que cuando el valor de ϵ es 1, el agente elige siempre su acción al azar, no pudiendo destacarse sobre el agente Random contra el que compite. Algo muy similar ocurre con los valores muy cercanos a 1. A medida que se aleja de 1 y se acerca a 0, el agente obtiene más victorias rápidamente porque trata de mantenerse aferrado a una política ganadora a costa de explorar menos. Sin embargo, esto tiene una desventaja que se manifiesta en la etapa de evaluación de la figura 3.12. En el caso de $\epsilon = 0$ la performance es peor que la del resto ya que no exploró lo suficiente para aprender. Vale mencionar que la diferencia es clara, pero que la performance de los algoritmos es buena en general por la dinámica del entrenamiento. Ya que en la evaluación todos los algoritmos tienen $\epsilon = 0$ y tuvieron un aprendizaje -en parte similar- porque el agente con el que compiten se comporta de manera aleatoria lo cual en la dinámica del juego (más allá de la política de exploración propia), los lleva a actualizar su función Q en varios de los estados posibles.

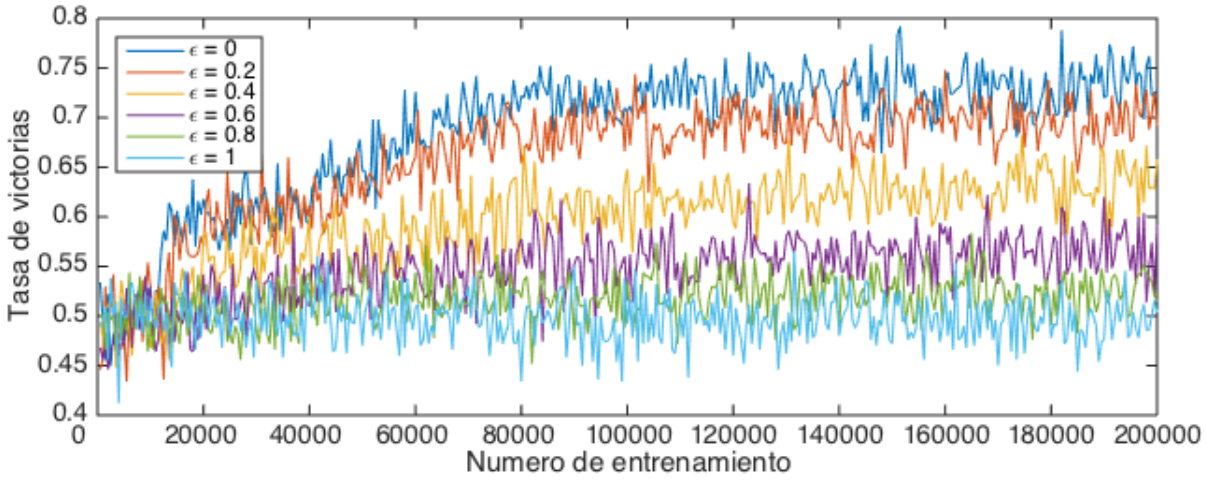

Figura 3.11: Parámetro ϵ en ϵ-greedy

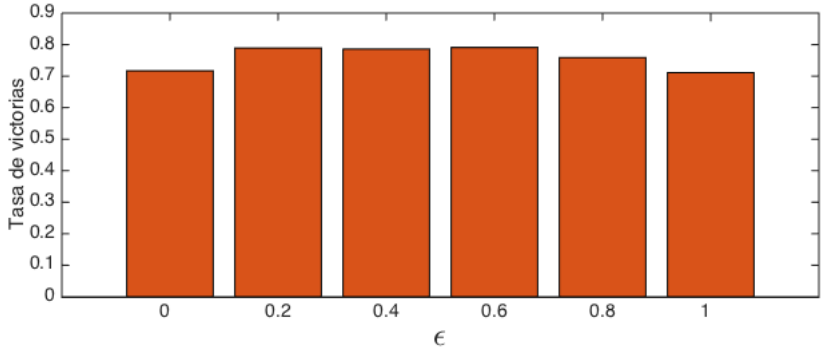

Figura 3.12: Evaluación después de entrenamiento: ϵ-greedy

Por último, la figura 3.13 muestra una política de explotación-exploración alternativa a ϵ-greedy conocida como softmax. En la figura puede verse el promedio de victorias cada 500 jugadas para tres variantes del algoritmo: con

baja temperatura ($\tau = 0{,}1$), con alta temperatura ($\tau = 1000$) y con temperatura decreciente (empieza con $\tau = 1000$ y disminuye a la décima parte cada 10000 intentos). Podemos ver que cuando la temperatura es excesivamente alta, el comportamiento es igual al del agente aleatorio (eligiendo al azar de una distribución uniforme sobre los posibles estados). Por otra parte, si bien la temperatura baja y la política decreciente alcanzan un mismo nivel de aciertos al final es importante destacar que la política decreciente llega a ese nivel mucho más rápido ya que se permite una política de exploración más agresiva al principio que le permite conocer más estados para luego ir optando cada vez más por mantenerse dentro de las políticas ganadoras.

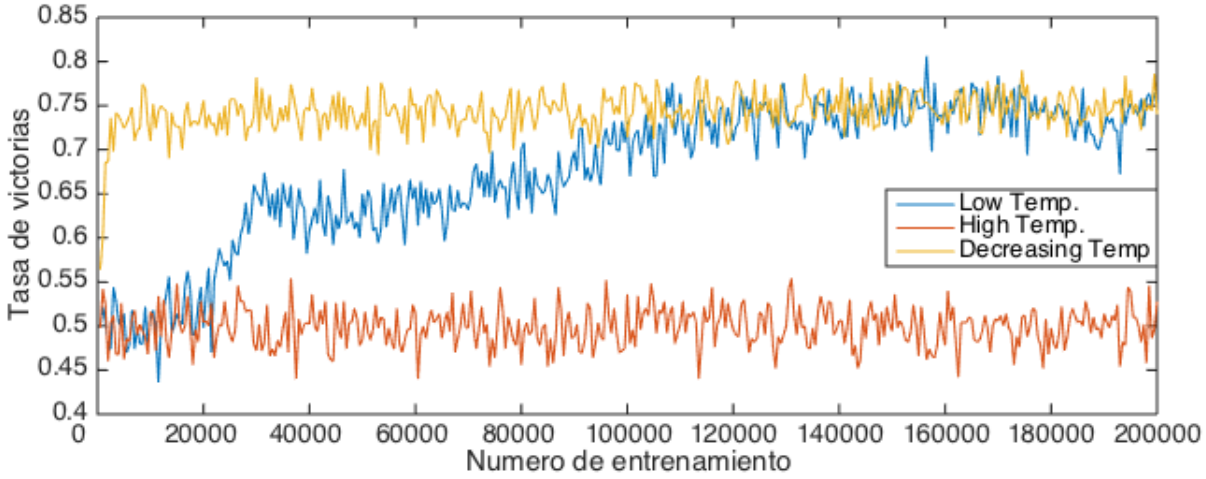

Figura 3.13: Parámetro de temperatura (τ) en Softmax

4. Discusión

El algoritmo de Q-Learning es un algoritmo muy interesante, sobre todo, por su simpleza y sus buenos resultados. En muy pocas líneas de código, uno logra tener un agente que puede aprender con la experiencia y tener una buena performance después de su entrenamiento.
Cuando la cantidad de estados se hace muy grande (como pudimos notar cuando duplicamos el tamaño del tablero), Q-Learning empieza a tener desventajas prácticas ya que requiere de mucho entrenamiento para poder explorar todos los estados al no tener un modelo sobre el mundo.
Por otro lado, la elección de refuerzos (como en todo algoritmo de Aprendizaje por Refuerzos) es central ya que esto repercute en la estrategia que será elegida por el agente ya que su fin último es optimizar la recompensa recibida. Por eso, es importante elegir una política acorde de valores iniciales de la función estado-valor Q, respecto de los refuerzos. O en todo caso, inicializarlos al azar si se desconoce el dominio del problema.
La tasa de aprendizaje α es otro factor central sobre el que prestar atención ya que regula el modo en que se incorpora nueva experiencia, respecto de la anterior. Valores muy cercanos a cero, o muy cercanos a 1, no tienen demasiado sentido porque dejan de lado las ventajas del algoritmo de Q-learning por no considerar la experiencia nueva o por darle demasiado peso, haciendo que no se converja al valor esperado.
Por último, la política de explotación-exploración presenta otro dilema central en el algoritmo que debe decidir cuándo optar por una acción que le de la mayor recompensa según su conocimiento actual versus cuándo explorar nuevos estados para mejorar su conocimiento del problema. Para esto, la política ϵ-greedy es una buena opción sencilla que permite regular el porcentaje de exploración de manera probabilística. Por otro lado, la política softmax que enfría el sistema siguiendo la distribución de Boltzmann obtiene un buena compensación permitiendo una exploración más agresiva al principio y una explotación más agresiva sobre el final.

Referencias

[1] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1.