Melissa Robles - 201632731
Reinforcement Learning
Universidad de los Andes
Bogotá 2023

Q-Learning

1. Introducción

En este documento se presentarán resultados y detalles de implementación del algoritmo de Q-learning para tres escenarios distintos: Grid World, Laberinto de cuartos y Taxi. Para cada escenario se definieron dos clases: un ambiente y un agente. Además de esto, se creó una clase Learner que ejecuta todo el algoritmo de aprendizaje. El ambiente define los estados y acciones generales, y tiene dos métodos `start` y `stop` que determinan los estados de inicio y de finalización. Para estos casos, se utilizó un estado general de finalización llamado `final` que se llega siempre que se termina un episodio. El agente es el que ejecuta los movimientos que ayudar al Learner a aprender. En particular, tiene un estado actual que va cambiando a medida que se ejecutan acciones, y un método `do_action` que permite realizar, a partir de un estado, una acción seleccionada. Este método retorna la recompensa obtenida y el estado de llegada luego de ejecutar la acción. Si se considera un *ruido* se modela dentro de este método.

Para instanciar la clase de Q-learning se deben especificar algunos parámetros:

- **Gamma:** El factor de descuento. Determina el comportamiento del aprendizaje pues puede penalizar el número de pasos antes de finalizar un episodio.
- **Epsilon:** Determina la preferencia entre explorar y explotar.
- **Alpha:** Learning rate.
- **decrease_alpha:** Factor que determina cuánto decrece el valor de alpha en el tiempo.
- **exploration_decreasing_decay**: Factor que determina cuánto decrece el valor de epsilon en el tiempo.
- **num_episodes_batch (N):** Determina el número de episodios a comparar para la condición de parada y la actualización de los valores de alpha y epsilon. Cada N episodios

se compara el promedio de las recompensas del batch pasado y si la diferencia es menor a 0.1 se detiene el algoritmo. Además se actualizan los valores de epsilon y alpha siguiendo la fórmula:

$$\epsilon_t = \epsilon_0 \cdot e^{\lambda \cdot t}$$

donde ϵ_0 es el valor inicial de epsilon pasado por parámetro, λ es el valor definido en `exploration_decreasing_decay` y t es el número de actualización (batch actual). Esta misma fórmula se utiliza para el alpha con los valores correspondientes.

2. Grid World

Para el escenario de Grid World se utilizó la plantilla de los anteriores proyectos. Una cuadrícula de 10x10 con un estado de llegada (representado en verde), varios estados de penalización que retornan una recompensa negativa (representados en rojo) y otras casillas prohibidas (representadas en gris). Se consideró un ruido de 0.2, con lo que con 80 % de probabilidad se ejecuta la acción escogida y con 20 % de probabilidad se ejecuta alguna otra acción posible. Cuando se escoge una acción que no se puede realizar, con 80 % de probabilidad el agente se queda quieto y de lo contrario se escoge alguna otra acción posible.

Para obtener la convergencia con los valores mostrados en la figura **??**, se necesitaron 56.500 episodios.

Variable	Valor
Epsilon	0.8
Alpha	0.4
Gamma	0.9
DecreaseAlpha	0.01
DecreaseEpsilon	0.01
EpisodesBatch	500

Figura 1: Valores para convergencia de Laberinto de cuartos

Los resultados, tanto en cuanto a política como a valores se muestran en las figuras 7 y 3.

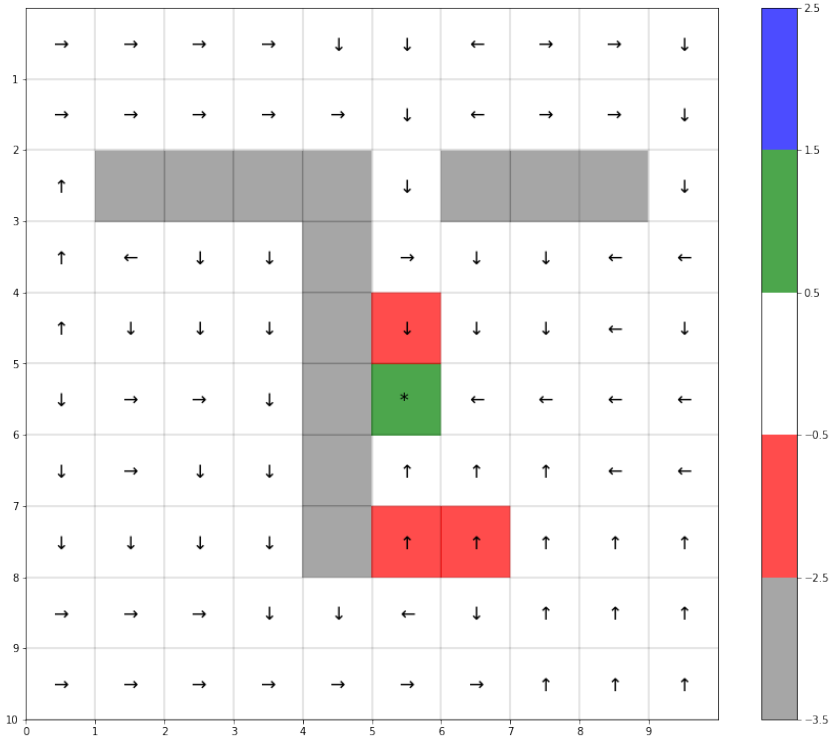

Figura 2: Resultado política Grid World

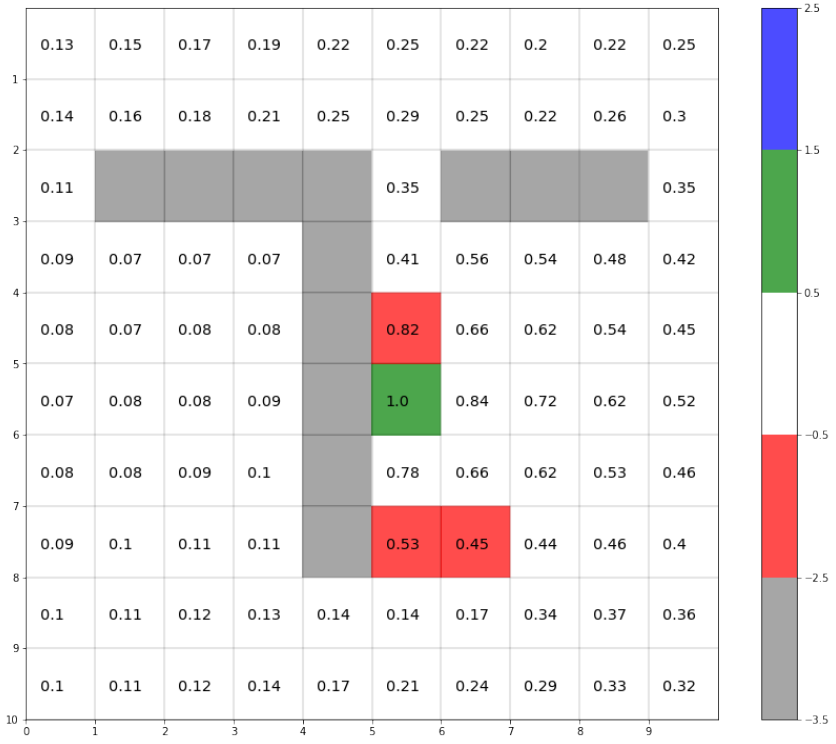

Figura 3: Resultado valores Grid World

Además de esto, se obtuvieron gráficos para resumir el comportamiento del aprendizaje.

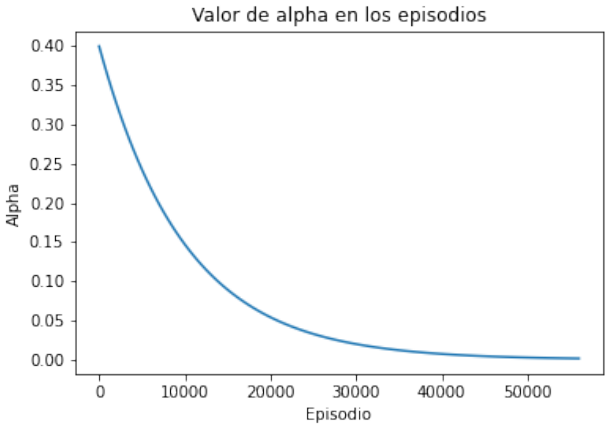

Figura 4: Cambios en el valor de alpha en los episodios

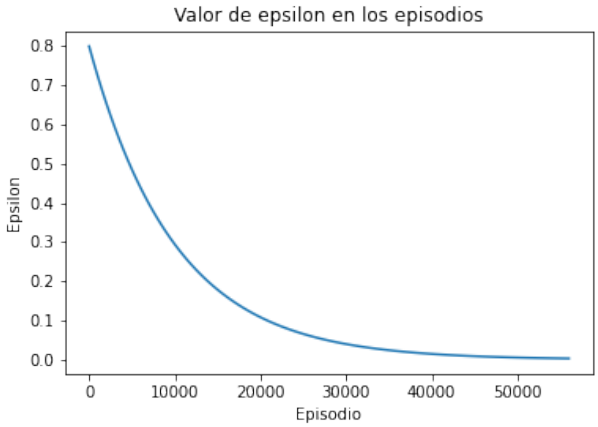

Figura 5: Cambios en el valor de epsilon en los episodios

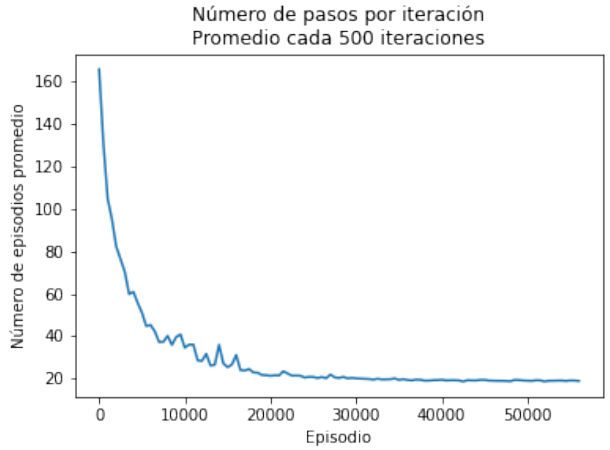

Figura 6: Numero de pasos promedio por episodio

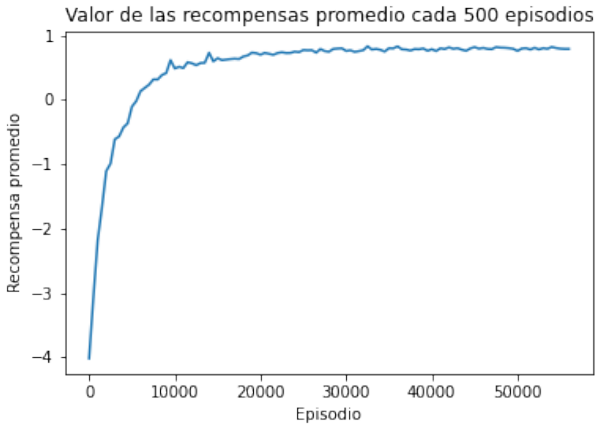

Figura 7: Recompensa promedio obtenida por episodio

3. Laberinto de cuartos

En este ambiente queremos que el agente aprenda a salir por el cuarto superior izquierdo en la menor cantidad de pasos posible comenzando desde cualquier punto de la cuadrícula. Para esto, se definieron los estados como tuplas (i,j) igual que en el ambiente de GridWorld, al igual que las acciones a tomar son $\uparrow, \downarrow, \rightarrow, \leftarrow$ y 'exit'. Esta última solo funciona si se está en el estado (0,2), pues es la única forma de salir del laberinto de cuartos. Además de esto, se consideró un ruido de 0.3 que funciona de la misma forma que en el ambiente GridWorld. Por último, se definió una recompensa de 1 si el agente logra salir del laberinto y una recompensa de -0.001 por cada paso que da sin llegar al final. Esto, junto con el valor de gamma garantiza que el agente aprenderá a llegar al final en el menor número de pasos posibles.

Para la convergencia del algoritmo se utilizaron los valores mostrados en la figura 8 y se necesitaron 20.800 episodios. Los resultados de los valores encontrados y la política óptima se muestran en las figuras 9 y 10. Por último, en las figuras 11, 12, 13 y 14 se muestran los cambios de los valores de epsilon, alpha, número promedio de pasos por episodio y recompensa promedio.

Variable	**Valor**
Epsilon	0.8
Alpha	0.4
Gamma	0.9
DecreaseAlpha	0.01
DecreaseEpsilon	0.01
EpisodesBatch	200

Figura 8: Valores para convergencia de Grid World

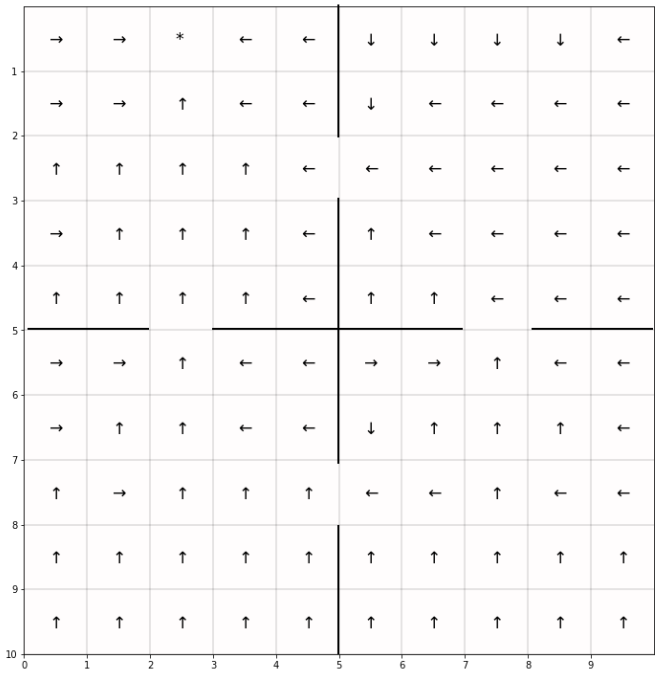

Figura 9: Resultado política Laberinto de cuartos

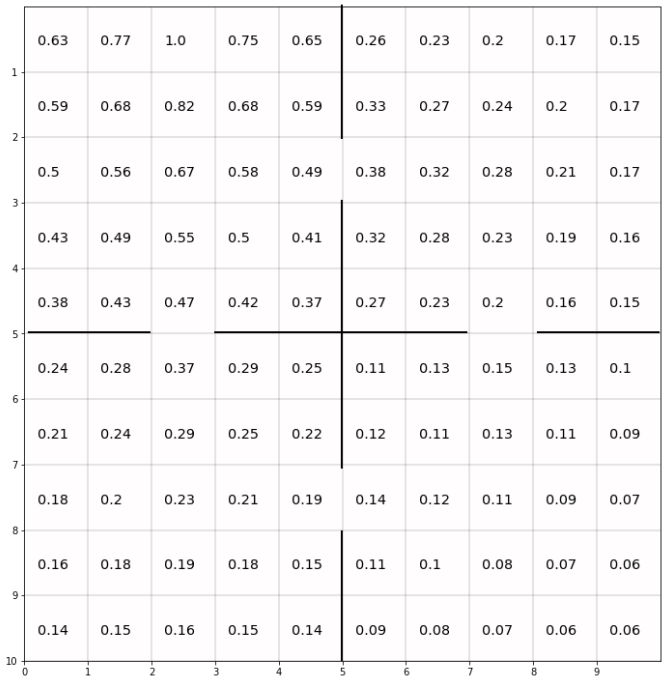

Figura 10: Resultado valores Laberinto de cuartos

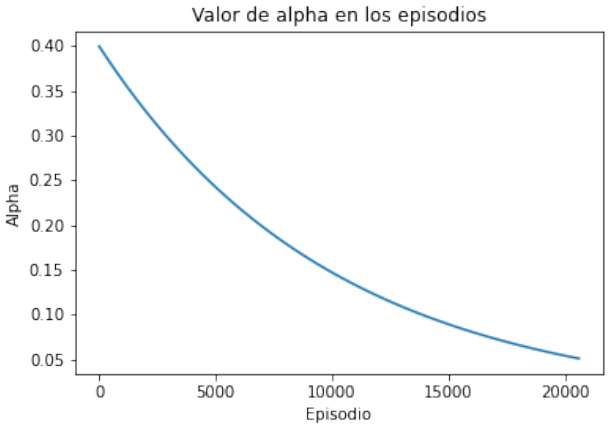

Figura 11: Cambios en el valor de alpha en los episodios

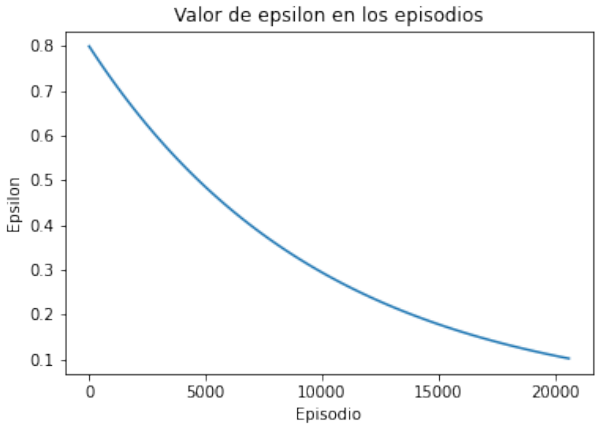

Figura 12: Cambios en el valor de epsilon en los episodios

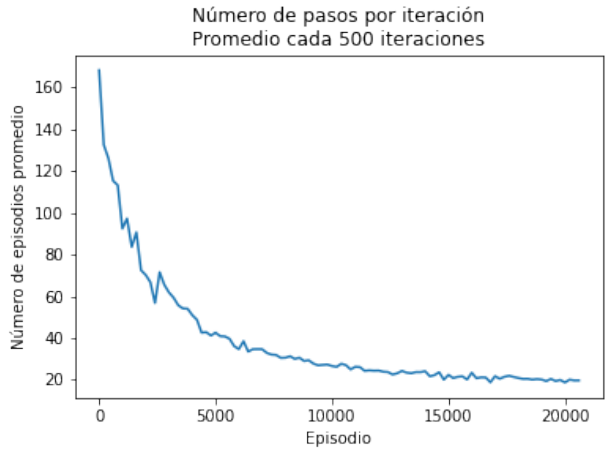

Figura 13: Numero de pasos promedio por episodio

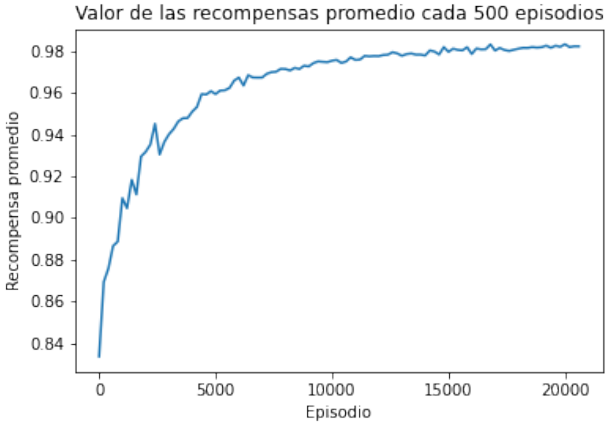

Figura 14: Recompensa promedio obtenida por episodio

4. Taxi

El problema del taxi consiste en un taxi (agente) que debe recoger a un pasajero y dejarlo en el lugar de destino. El lugar de recogida y de llegada debían ser paraderos que se encuentran en las esquinas del tablero. Además de esto, hay paredes que no permiten que el taxi se mueva libremente.

Los estados para este problema se modelaron como triplas (i, j, k) en donde (i, j) representan la posición del taxi en un momento del tiempo y $k \in \{0, 1\}$ representa si el taxi está ocupado o no. Además, se determinó una recompensa de 1 si se recogía a un pasajero, de 5 si se dejaba el pasajero en su destino y de -10 si se intentaba dejar o recoger a un pasajero de forma equivocada.

Para la ejecución del algoritmo de Q-Learning para el problema del taxi, inicialmente se tomó como un punto fijo de recogida del pasajero en las coordenadas (0,0) y punto de llegada las coordenadas (4,0), esto con el fin de reducir el número de iteraciones necesarias para la convergencia puesto que estaba tomando mucho tiempo para la convergencia del problema original. Con esta modificación y tomando las decisiones como determinísticas, el algoritmo tomó 40500 episodios en converger con los valores iniciales mostrados en la figura 15.

Para mostrar los resultados se muestra la política y los valores obtenidos por estado en dos cuadrículas: una que refleja los estados del taxi antes de recoger al pasajero y una después de recogerlo. Esto se puede evidenciar en las figuras 21 y 17.

Variable	Valor
Epsilon	0.8
Alpha	0.4
Gamma	0.9
DecreaseAlpha	0.01
DecreaseEpsilon	0.01
EpisodesBatch	500

Figura 15: Valores para convergencia de Taxi

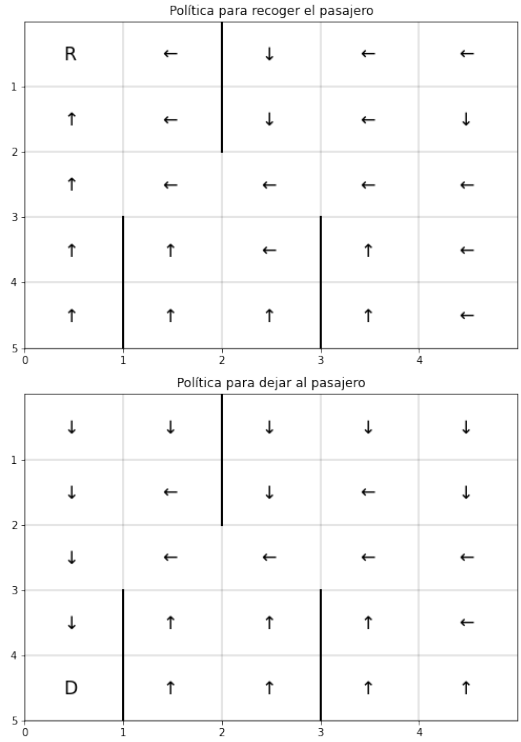

Figura 16: Resultado política Taxi

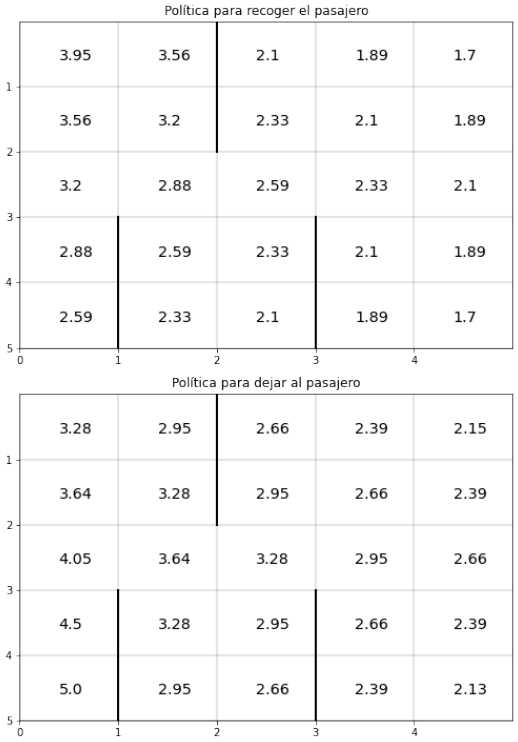

Figura 17: Resultado valores Taxi

Además, al igual que en las otras secciones se muestran los cambios de los valores de epsilon, alpha, número de pasos por iteración y recompensa obtenida a lo largo del aprendizaje. Se puede evidenciar que el agente va aprendiendo a medida que se ejecutan las iteraciones y va disminuyendo la tasa de aprendizaje.

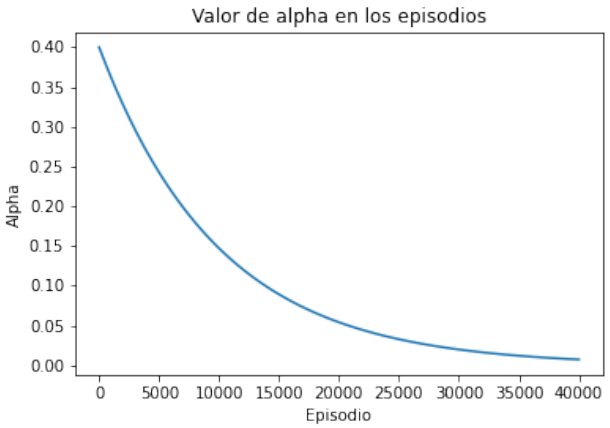

Figura 18: Cambios en el valor de alpha en los episodios

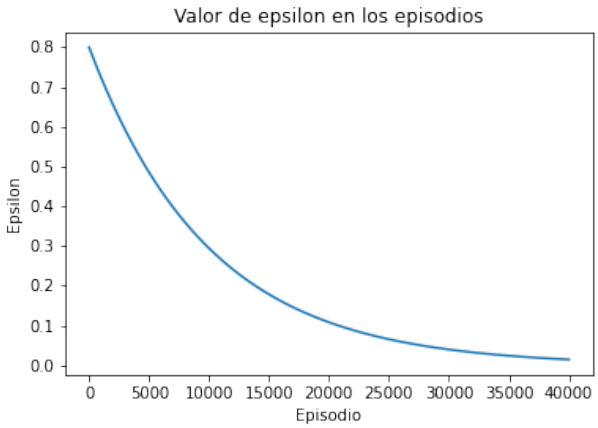

Figura 19: Cambios en el valor de epsilon en los episodios

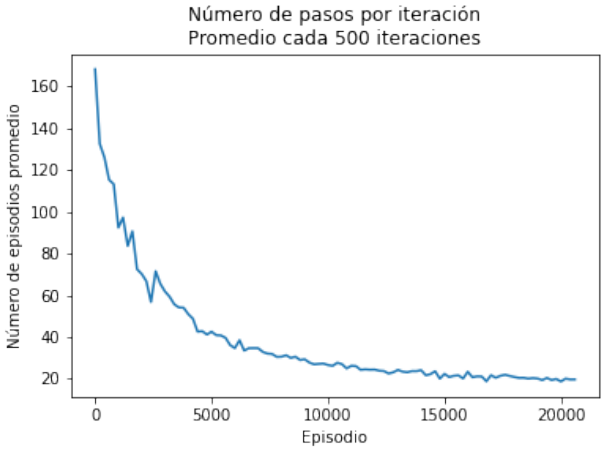

Figura 20: Numero de pasos promedio por episodio

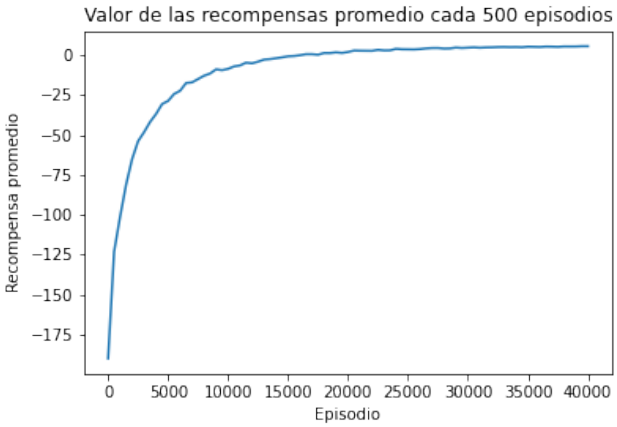

Figura 21: Recompensa promedio obtenida por episodio