Reinforcement Learning – Assignment 6

Fecha de Entrega: 10 de abril de 2023
Presentado por: Sara María Cachique Leguizamón

Algoritmo de Q-Learning

El algoritmo Q-learning se utiliza para entrenar un agente a tomar decisiones en un ambiente, es decir que el agente aprende de las interacciones que tiene de su entorno. La función Q-value sirve para estimar la calidad de una acción en un estado y acción especifica, su objetivo es maximizar la recompensa acumulada a largo plazo.

1. Se define el ambiente, se inicializan los estados (espacio de estados) y acciones del ambiente.
2. Selección de acción: el agente selecciona una acción a tomar de acuerdo con el estado actual con estrategia de exploración.
3. Toma de decisión: el agente toma una acción y de eso obtiene el estado y la recompensa asociada a la acción.
4. Actualización de la función Q: El agente actualiza la función Q para el par (estado, acción) basándose en el nuevo q-value y la recompensa asociada. Sigue la siguiente ecuación.

$$Q(s,a) = Q(s,a) + \alpha * (R + \gamma * \max(\mathrm{Q}(\mathrm{s}',\mathrm{a}')) - Q(s,a))$$

Donde:
Q(s,a) es el valor de la función Q para el estado s y la acción a.
α es la tasa de aprendizaje
R es la recompensa recibida por el agente al tomar la acción
γ es el factor de descuento
Q(s', a´) la función Q para el estado próximo.

5. Se itera los pasos dos al cuatro un numero determinado de episodios.

Ambiente Gridworld

Para el ambiente de gridworld se logra convergencia al cabo de 100 episodios. Esto se puede observar en la implementación. Los parámetros seleccionados para es escenario son:

$\alpha = 0.1,\ \gamma = 0.6,\ epsilon = 0.5$ (50 y 50 en exploración y explotación)

El programa imprime, la acción ejecutada (right (1), left(0), up(2), down(3)) en el estado actual (i,j), estado actual, acción seleccionada y estado próximo, también imprime las acciones posibles (que pueden ser 0, 1, 2, 3) de acuerdo a cada estado, para verificar el correcto funcionamiento del mismo. Como se muestra a continuación.

```
Executed action: left at state (5, 1)
(5, 1), 0, (5, 0)
[1, 2, 3]
```

Ilustración 1. Estado inicial, acción y estado proximo

Finalmente imprime la Q- table, como se muestra a continuación, estado y su q-value de acuerdo con cada acción mencionada.

```
{(0, 0): array([0., 0., 0., 0.]), (0, 1): array([0., 0., 0., 0.]), (0, 2): array([0., 0., 0., 0.]), (0, 3): array([0., 0., 0., 0.]),
```

Ilustración 2. Q-table

Se toma como base el algoritmo, presentado por el profesor. Los únicos cambios sustanciales en el mismo son:

La función de las recompensas: En el entorno, la recompensa es de 1 para la casilla objetivo y -1 para las casillas trampa y para moverse es de 0.

La función del entorno denominada Gridworld: se establece el entorno del gridworld, espacio por el cual se moverá el agente.

La función del agente: Las acciones que puede realizar el agente para llegar un nuevo estado hasta que este llegue a la casilla objetivo.

Ambiente Laberinto

Para este ambiente se quiere que el agente aprenda a salir por el cuarto superior izquierdo en la menor cantidad de pasos posible. La única restricción de este ambiente es que al final de cada episodio el agente comienza nuevamente en cualquier posición valida del laberinto.

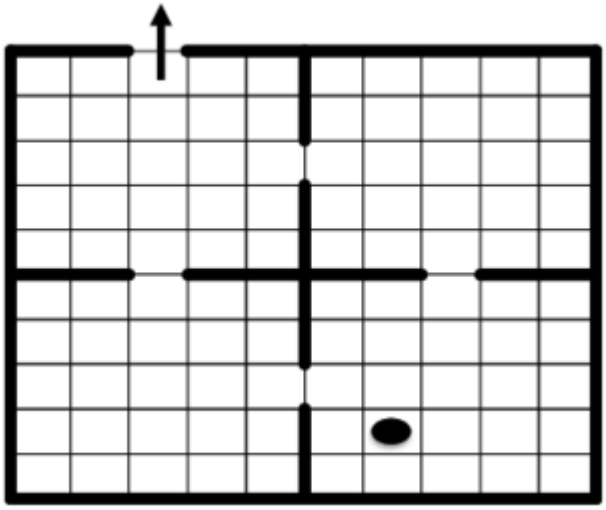

Ilustración 3. Tablero Laberinto

Para este caso se deja el inicio y destino fijos, la posición para el inicio es (8,6), al igual que con el gridworld se establece en entorno del laberinto. Las acciones se basan en las restricciones que se

tienen en los muros del laberinto (esto se establece en los movimientos del agente) y finalmente el algoritmo empieza a converger después de 500 episodios con un épsilon de 0.2, una tasa de aprendizaje de 0.1 y un descuento de 0.6, como se muestra a continuación.

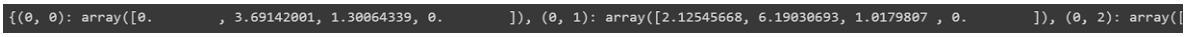

Ilustración 4. Q-table en posición (0,0) y (0,1) a los 400 episodios

Ilustración 5. Q-table a los 500 episodios

De lo anterior se puede visualizar la política que se esta generando, en el estado (0,0) el agente se mueve con mayor probabilidad a la derecha, y es la acción que más realizará. Ahora para el punto de partida (8,6) el agente es más probable que baje, como se muestra a continuación para llegar a las primeras casillas (entiéndase que bajar es el componente 4 del arreglo y que es restar en filas y mantener la columna, ósea ir al estado (7,6) del tablero).

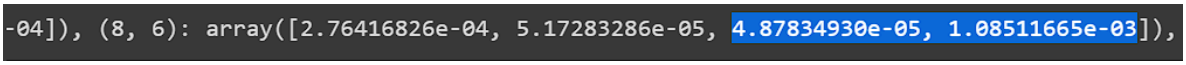

Ilustración 6. Posición de inicio (8,6) política determinada

Ambiente Taxi

Para el ambiente del Taxi se tiene en cuenta un estado adiciona al de la posición y es si el agente tiene o no tiene pasajero. Este es de vital importancia y se adiciona a q- table, para determinar los q-values, de cada acción, también se adiciona las acciones recoger y dejar como acciones relacionadas a el estado, entiéndase que este estado esta definido como tripleta de (x,y, p=pasajero), esta tercer componente puede ser 1 y 0 para los casos de si tiene o no pasajero respectivamente. Como destinos finales para el agente se definen tres ubicaciones (R, G, Y) y como estación del pasajero la posición en B. Es importante resaltar que el agente en este algoritmo es el taxi, debido a que este puede interactuar con el entorno y tomas las decisiones con el objetivo de cumplir o maximizar la recompensa, el pasajero no es un agente y solo se ubica en una determinada posición del tablero, esta distinción es importante para poder crear el ambiente, las posibles acciones y el comportamiento en general del algoritmo.

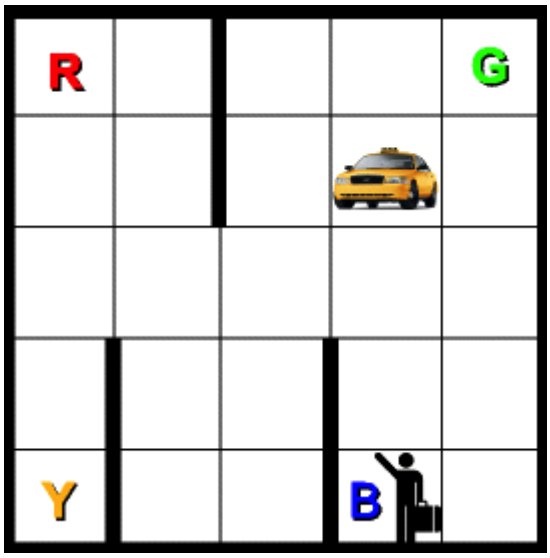

Ilustración 7. Tablero Taxi

El programa imprime lo siguiente.

```
Executed action: left at state (2, 1, 0)
(2, 1, 0), 0, (2, 0, 0)
```

Ilustración 8. Estado Inicial (x, y, pasajero), Acción y Estado próximo(x, y, pasajero)

Después de 100 iteracciones empieza a converger el algoritmo, los valores no cambian significativamente en este punto. Esto se puede observar en la implementación. Esto se puede visualizar en la siguiente imagen: el estado es la tripleta y las acciones son (0 (izq), 1 (derecha), 2 (arriba), 3 (abajo), 4(drop o dejar) y 5(peek o recoger)), para el estado (4,3,0) que significa coordenadas x, y, sin pasajero, el agente en esa posición deseable es que recoja y este comportamiento se puede visualizar en la los valores que se tienen en q-table.

```
0.00000000e+00, 0.00000000e+00]), (4, 2, 0): array([0., 0., 0., 0., 0., 0.]), (4, 2, 1): array([ 1.08324044e-06,  0.00000000e+00,
-1.00000000e+00,  0.00000000e+00]), (4, 3, 0): array([0.        , 0.35748443, 0.35999699, 0.        , 0.        ,
1.00139423]), (4, 3, 1): array([ 0.00000000e+00,  2.79695405e-05,  3.34784950e-03,  0.00000000e+00,
-9.39966731e+00,  0.00000000e+00]), (4, 4, 0): array([0.59960407, 0.        , 0.20007725, 0.        , 0.        ,
```

Ilustración 9. Comportamiento del estado (4,3,0)