CI-0148 Aprendizaje Máquina
Prof. Pablo Sauma Chacón

Laboratorio 7

Objetivo: Familiarizar al estudiante con la creación y el uso del algoritmo de aprendizaje por refuerzo Q-Learning en un entorno controlado.

Enunciado: Lea las instrucciones del documento y resuelva el enunciado en Python. Al inicio de su documento adjunte su nombre y carnet como un comentario. Este trabajo es de carácter individual, tampoco se aceptarán códigos que no hayan sido desarrollados por su persona.

Para este laboratorio se trabajará con un juego de laberintos ideado por su profesor. La interfaz gráfica para dicho programa la puede encontrar aquí.

El código permite recorrer laberintos autogenerados lo que permite contar con una diversidad de escenarios dónde probar su agente, así como botones mediante los cuáles podrá modificar los parámetros de su agente. El objetivo consiste en que su agente (celda roja) logre salir del laberinto (celda celeste) desplazándose por medio de los pasillos del mismo (celdas blancas). También se cuenta con una opción adicional que permite agregar un requerimiento previo a salir del laberinto que consiste en encontrar la llave para la puerta (celda dorada). Y la alternativa de activar los "tesoros" en el nivel, que le permitirán a su agente adquirir más puntos. En caso de utilizar el caso con llave, su agente deberá obtener la llave previo para poder "salir" del laberinto; si la llave no se encuentra en su posesión, alcanzar la celda de salida no terminará la simulación. Para fines del entorno, caminar contra una celda negra equivale a perder y por lo tanto termina la simulación.

Dentro del código encontrará la clase `Agent` con tres funciones que son las que debe implementar:

1. Método de inicialización `__init__(self, seed, state_dims, actions, learning_rate, discount_factor, eps_greedy, decay)` que inicializa el agente que utiliza Q-Learning. El parámetro `seed` corresponde a un número entero que se utilizará para sembrar la pseudoaleatoriedad del agente (note que cualquier llamado a la clase random deberá hacerse al objeto prng en su lugar). El parámetro `state_dims` corresponde a las dimensiones del espacio del problema en forma de tupla (altura,ancho). El uso de llaves o tesoros agregan valores adicionales a la tupla/dimensiones superiores que representan la posesión de la llave/tesoro respectivo (será un 0 o un 1 dependiendo de si ya se posee el ítem o no). El parámetro `actions` será una lista con las acciones posibles, seguido de `learning_rate`, `discount_factor`, `eps_greedy` y `decay` que son parámetros propios del algoritmo. Para fines de la implementación se asumirá que el `decay` aplica exclusivamente al valor de epsilon-greedy y la tasa de aprendizaje se mantendrá constante a lo largo de las simulaciones.

2. Método de simulación `simulation(self, env)` que ejecuta una simulación del agente dentro del laberinto. El parámetro `env` es la variable de tipo `Maze` que corresponde al entorno de nuestro agente. Recuerde que para ejecutar una simulación debe reiniciar el entorno, y luego seguir ejecutando acciones (en este caso un llamado al método `step` del punto 3) hasta que se haya alcanzado un estado terminal. Note que el objeto `env` posee llamados para reiniciar el entorno y para preguntar si el estado actual es un estado final. Al final de cada simulación es necesario reajustar el valor epsilon-greedy acorde a la tasa de decaimiento.

3. Método de ejecución `step(self, env, learn=True)` que ejecuta un paso del agente. El parámetro `env` es la variable de tipo `Maze` que corresponde al entorno de nuestro agente. El parámetro `learn` indica si el agente está ejecutando en modo aprendizaje o no. De estar ejecutando en modo aprendizaje, el agente deberá generar una variable aleatoria y comparar su valor con el epsilon-greedy actual. De ser menor deberá elegir una acción al azar (modo exploración), de ser mayor (o si `learn=False`) elegirá la mejor acción conocida hasta ahora. Note que el objeto `env` posee un método para obtener el estado actual en forma de tupla (altura, ancho) o (altura,ancho,llave) dependiendo del problema, además de un método para ejecutar una acción, la cuál retorna tanto la recompensa obtenida como el nuevo estado alcanzado. De estar ejecutando en modo aprendizaje deberá actualizar los pesos de la tabla Q.

4. Finalmente, pruebe su agente para los siguientes escenarios: `Map Seed=17`, `Key=Off` y `Map Seed=21`, `Key=On`; ejecute múltiples iteraciones, pruebe con diferentes combinaciones de parámetros y describa cualquier patrón o comportamiento interesante que haya encontrado. Luego juegue con las semillas de los mapas para encontrar alguno que le parezca “interesante”, documente su semilla y la combinación de parámetros que utilizó para resolverlo. Finalmente, documente cualquier otro aprendizaje que este laboratorio le haya dejado.
    a. También debe probar algunos casos aplicando tesoros. ¿Qué cambia en el comportamiento?
    b. Nota: como parte del laboratorio deberá modificar los valores de las diversas recompensas, así como los parámetros del algoritmo.