Práctica 1

Laura Rodríguez Navas
rodrigueznavas@posgrado.uimp.es

5 de marzo de 2021

Ejercicio 1

1. Descargar el código fuente para esta práctica, *softpractica1.zip*, de la página web de la asignatura
2. Descomprimir el fichero anterior.
3. Abrir un terminal o consola de comandos y entrar dentro de la carpeta *softpractica1*.
4. Para empezar vamos a ejecutar GridWorld en el modo de control manual, que utiliza las teclas de flecha. *python gridworld.py -m -n 0*
5. El objetivo es lograr llegar lo antes posible a la celda etiquetada con un 1, evitando caer en la celda con un -1.

Preguntas

1. ¿Cuántas celdas/estados aparecen en el tablero? ¿Cuántas acciones puede ejecutar el agente? Si quisieras resolver el juego mediante aprendizaje por refuerzo, ¿cómo lo harías?
2. Abrir el fichero *qlearningAgents.py* y buscar la clase *QLearningAgent*. Describir los métodos que aparecen en ella.
3. Ejecuta ahora el agente anterior con:
   python gridworld.py -a q -k 100 -n 0
4. ¿Qué información se muestra en el laberinto? ¿Qué aparece por terminal cuando se realizan los movimientos en el laberinto?
5. ¿Qué clase de movimiento realiza el agente anterior?
6. ¿Se pueden sacar varias políticas óptimas? Describe todas las políticas óptimas para este problema.
7. Escribir el método *update* de la clase *QLearningAgent* utilizando las funciones de actualización del algoritmo *Q-Learning*. Para ello, inserta el código necesario allí donde aparezca la etiqueta INSERTA TU CÓDIGO AQUÍ siguiendo las instrucciones que se proporcionan, con el fin de conseguir el comportamiento deseado.

8. Establece en el constructor de la clase *QLearningAgent* el valor de la variable *epsilon* a 0,05. Ejecuta nuevamente con:
   python gridworld.py -a q -k 100 -n 0
   ¿Qué sucede?

9. Después de la ejecución anterior, abrir el fichero *qtable.txt*. ¿Qué contiene?

Ejercicio 2

En el ejercicio anterior, siempre que el agente decid´ıa moverse hacia una dirección se movía en esa dirección con probabilidad 1. Es decir, se trataba de un MDP determinista. Ahora vamos a crear un MDP estocástico:

1. Ejecuta y juega un par de partidas con el agente manual:
   python gridworld.py -m -n 0.3
   ¿Qué sucede? ¿Crees que el agente *QLearningAgent* será capaz de aprender en este nuevo escenario?

2. Reiniciar los valores de la tabla Q del fichero *qtable.txt*. Para ello ejecutar desde el terminal:
   cp qtable.ini.txt qtable.txt

3. Ejecutar el agente *QLearningAgent*:
   python gridworld.py -a q -k 100 -n 0.3

4. Tras unas cuantos episodios, ¿se genera la política óptima? Y si se genera, ¿se tarda más o menos que en el caso determinista?