Aprendizaje automático

Tutorial 4

Grupo 84

Sabrina Riesgo Reyes	100363834	100363834@alumnos.uc3m.es
Laura Yunta García	100363785	100363785@alumnos.uc3m.es

Índice

1. INTRODUCCIÓN 3
2. EJERCICIO 1 3
3. EJERCICIO 2 7
4. DESCRIPCIÓN DE LA FUNCIÓN UPDATE 8
5. FICHEROS GENERADOS CON LAS TABLAS Q 9
6. CONCLUSIONES 9

1. INTRODUCCIÓN

El objetivo de la práctica planteada es familiarizarse con el software en el cual se implementa Q-Learning en el dominio del GridWorld. Para ello se han realizado una serie de ejercicios planteados de los cuales se han podido sacar bastantes conclusiones al respecto.
Cabe destacar que el documento se va a dividir en seis secciones distintas que serán:

- Introducción.
- Ejercicio 1: Se resolverán las distintas preguntas planteadas para el ejercicio 1.
- Ejercicio 2: Se resolverán las distintas preguntas planteadas para el ejercicio 2.
- Explicación de la función update: En este apartado se dará una explicación de la función update programada en el apartado 8 del ejercicio 1.
- Ficheros generados con las distintas tablas Q: En este apartado se podrá visualizar las diferentes tablas de valores obtenidas al ejecutar los distintos ejercicios.
- Conclusiones.

2. EJERCICIO 1

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?

El tablero contiene 11 celdas/estados, ya que a pesar de tener 12 casillas una de ellas no es accesible, por lo que no es un posible estado del tablero.
El agente puede ejecutar cinco acciones (norte, sur, este, oeste, salida), aunque no siempre se permitan todas ya que depende de la celda en la que se encuentre. Por ejemplo, en el estado inicial sólo podría ejecutar las acciones norte y este. La acción salida solo se ejecuta en los estados finales.
Para resolver el juego mediante aprendizaje por refuerzo, se entrenaría por medio de prueba y error al agente para que seleccione la acción correcta a ejecutar en el estado actual, de modo que vaya desde la casilla en la que se encuentra inicialmente hasta la casilla que contiene el 1, evitando pasar por la casilla que contiene el -1 y consiguiendo la mayor recompensa posible. Para ello se le otorgaría una recompensa al realizar una acción correcta y se penalizaría en caso de que la acción realizada no fuera correcta.

2. Abrir el fichero qlearningAgents.py y buscar la clase QLearningAgent. Describir los métodos que aparecen en ella.

MÉTODO	DESCRIPCIÓN
__init__(self, **args)	Inicializa los valores de la tabla Q.
readQtable(self)	Lee los valores de la tabla Q.
writeQtable(self)	Escribe los valores en la tabla Q.

__del__(self)	Es invocado al final de cada episodio como destructor.
computePosition(self, state)	Devuelve el número correspondiente a una casilla dado un estado. La numeración comenzaría por la casilla inferior izquierda y terminaría en la superior derecha.
getQValue(self, state, action)	Dado un estado y una acción, devuelve el valor de tabla Q correspondiente. En caso de no haber visto nunca el estado devolvería el valor 0.0.
computeValueFromQValues(self, state)	Dado un estado, devuelve el valor máximo de la tabla Q perteneciente a una de las acciones legales. En el caso de los estados finales devuelve el valor 0.0 ya que no se puede realizar ninguna de las acciones norte, sur, este y oeste.
computeActionFromQValues(self, state)	Dado un estado, devuelve la mejor acción que se puede realizar. En el caso de los estados finales devuelve “None” ya que no se puede realizar ninguna de las acciones norte, sur, este y oeste.
getAction(self, state)	Dado un estado, determina la acción a realizar. Se toma una acción aleatoria con probabilidad epsilon de entre las acciones legales. En caso contrario se tomará la acción que viene determinada por la mejor política. Finalmente, para los estados finales devuelve “None” ya que no se puede realizar ninguna de las acciones norte, sur, este y oeste.
update(self, state, action, nextState, reward)	No realiza ninguna función. Se desarrollará posteriormente.
getPolicy(self, state)	Dado un estado devuelve la mejor acción de la tabla Q.
getValue(self, state)	Dado un estado devuelve el valor más alto de la tabla Q.

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?

En el laberinto se muestran las casillas estado divididas en cuatro trozos que contienen un número que indica la propagación del refuerzo (valor de la tabla Q) en función de la acción elegida.

Por terminal aparece en primer lugar “RUNNING 100 EPISODES” que indica el número de episodios de aprendizaje, valor que ha sido introducido como argumento (-k 100). A continuación, se mostrará para cada episodio el número de episodio en el que se encuentra, por ejemplo “BEGINNING EPISODE: 2”, y además se mostrará dentro de cada episodio, por cada instante la siguiente información, indicando un ejemplo:

- Started in state: (0,0) → indica el estado inicial en el que se encuentra el agente.
- Took action: north → indica la acción realizada.
- Ended in state: (0,1) → indica el estado al que se llega desde el estado inicial una vez ejecutada la acción anterior.
- Got reward: 0,.0 → indica el refuerzo.

En el caso de que “Ended in state” tome el valor “TERMINAL_STATE” significa que ha llegado a uno de los estados finales, se ha realizado la accion “exit” y se indicaría el fin del episodio y el valor medio de refuerzo este, por ejemplo “EPISODE 2 COMPLETE: RETURN WAS 0.0984770902184”.

5. ¿Qué clase de movimiento realiza el agente anterior?

El agente realiza un movimiento totalmente aleatorio. Inicialmente no se actualizan los valores de la tabla Q según las recompensas y penalizaciones ya que el método “update” de momento no realiza ninguna función.

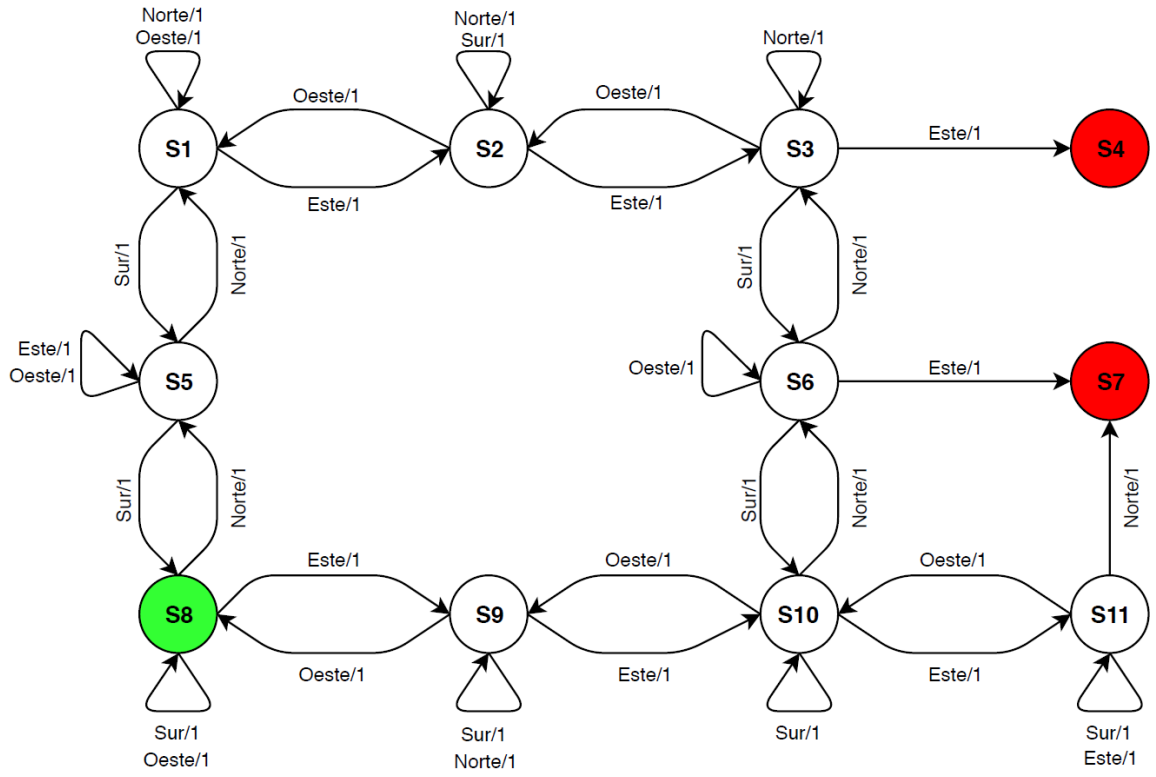
6. Dibujar el MDP considerando un entorno determinista.

Notación seguida para los estados en el proceso de decisión de Markov:

S1	S2	S3	S4
S5		S6	S7
S8	S9	S10	S11

Proceso de decisión de Markov:

- Conjunto de estados = {s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8, s_9, s_{10}, s_{11}}
- Conjunto de acciones = {norte, sur, este, oeste}



7. ¿Se pueden sacar varias políticas óptimas? Describe todas las políticas óptimas para este problema.

Sí, se pueden sacar dos políticas óptimas. En este caso el objetivo sería llegar a la casilla que contiene un 1, realizado las acciones que permitan llegar a esta de la mejor forma posible. La primera política consiste en utilizar la distancia de Manhattan para llegar a la casilla objetivo. De este modo se obtiene el camino más corto entre los puntos de inicio y fin y las acciones a realizar por el agente.
La segunda política consiste en realizar el camino más eficiente evitando pasar por la casilla que contiene un -1, ya que de este modo se obtendría el mayor refuerzo posible.
Teniendo en cuenta las políticas descritas anteriormente, se mostrarán las acciones a realizar por el agente desde el estado inicial al estado final bueno:

Estado Actual (columna, fila)	Acción	Siguiente Estado (columna, fila)
(0,0)	Norte	(0,1)
(0,1)	Norte	(0,2)
(0,2)	Este	(1,2)
(1,2)	Este	(2,2)
(2,2)	Este	(3,2)
(3,2)	Salida	--

8. Escribir el método update de la clase QLearningAgent utilizando las funciones de actualización del algoritmo Q-Learning.

9. 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?

El agente va realizando distintas acciones hasta llegar a los 100 episodios. Durante este período de tiempo se puede apreciar como se va moviendo en función de los valores de la tabla Q que aparecen reflejados en cada una de las casillas. El agente va realizando acciones y probado en función de dichos valores hasta que finalmente se puede apreciar como realiza solo aquellas acciones que le permiten ir desde el estado inicial hasta el estado final por el camino más eficiente. Además, se puede apreciar el camino marcado en verde resaltando los valores de la tabla Q correspondientes a aquellas acciones pertenecientes al camino óptimo.

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

El fichero qtable.txt contiene los nuevos valores de la tabla Q organizados. El orden de las casillas sería:

9	10	11	12
5	6	7	8
1	2	3	4

En el fichero cada línea se corresponde con una casilla del tablero, de modo que la línea 1 tendrá los valores de la casilla 1, la línea 2 los de la casilla 2 y así sucesivamente. En cada línea aparecen 5 valores, cada uno de ellos corresponde a la acción en el siguiente orden: norte, este, sur, oeste, salida.

3. EJERCICIO 2

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?

En este caso al haber realizado una ejecución con un nivel de ruido de 0.3 hace que el agente no tenga unos movimientos deterministas y por tanto no va a tener los valores exactos de las acciones a realizar para llegar a su meta. De hecho, en ocasiones se ha observado que al pulsar cualquier flecha de movimiento el agente se mueve a una dirección que no es la indicada. Esto ocurre debido a que la acción que se ejecuta, aunque sea la elegida puede no ser legal e incluso puede que la acción que se ejecuta puede no ser la elegida y además no ser legal, por lo tanto, el agente se queda en su posición actual.
Independientemente de todo esto, consideramos que el agente QLearningAgent sí que será capaz de aprender en el nuevo escenario ya que utilizará las técnicas de aprendizaje por refuerzo para calcular perfectamente las políticas óptimas que le harán llegar al objetivo deseado.

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

Sí, tras realizar bastantes episodios el agente por fin consigue obtener la política óptima, aunque en este caso tarda más que en el caso determinista, esto ocurre debido a que comienza realizando movimientos aleatorios que finalmente con el refuerzo obtenido de todos los episodios consigue mejorar la ejecución de los mismos; cosa que no ocurre en el caso determinista.
La política óptima obtenida es la siguiente:

Estado Actual (columna, fila)	Acción	Siguiente Estado (columna, fila)
(0,0)	Este	(1,0)
(1,0)	Este	(2,0)
(2,0)	Norte	(2,1)
(2,1)	Norte	(2,2)
(2,2)	Este	(3,2)
(3,2)	Salida	--

4. DESCRIPCIÓN DE LA FUNCIÓN UPDATE

En este apartado vamos a proceder a explicar la función “update” implementada para la ejecución final del ejercicio 1 y la del ejercicio 2.
La función comienza adquiriendo la posición actual del agente además de la siguiente acción a ejecutar por el mismo. A continuación, la ejecución del algoritmo entra en un condicional que depende directamente de si el siguiente estado al que se va a ir es el estado final o no lo es.
En caso de que sí lo sea, se actualizará el valor de la tabla Q utilizando el valor de reward como 1; en caso contrario, igualmente se actualizará el valor de la tabla Q con el valor de reward -1.

5. FICHEROS GENERADOS CON LAS TABLAS Q

Valores de la tabla Q del ejercicio 1:

	Norte	Este	Sur	Oeste	Salida
Casilla 1	0.59049	0.00000	0.26572	0.44294	0.00000
Casilla 2	0.00000	0.08457	0.00000	0.00000	0.00000
Casilla 3	0.28761	0.00000	0.00000	0.02537	0.00000
Casilla 4	0.00000	0.00000	0.00000	0.00000	0.00000
Casilla 5	0.65610	0.00000	0.26571	0.00000	0.00000
Casilla 6	0.00000	0.00000	0.00000	0.00000	0.00000
Casilla 7	0.77124	0.00000	0.00000	0.00000	0.00000
Casilla 8	0.00000	0.00000	0.00000	0.00000	-0.75000
Casilla 9	0.31405	0.72900	0.44271	0.42031	0.00000
Casilla 10	0.00000	0.81000	0.54527	0.49207	0.00000
Casilla 11	0.00000	0.90000	0.47698	0.54675	0.00000
Casilla 12	0.00000	0.00000	0.00000	0.00000	1.00000

Valores de la tabla Q del ejercicio 2:

	Norte	Este	Sur	Oeste	Salida
Casilla 1	0.30846	0.34785	0.26529	0.13972	0.00000
Casilla 2	0.26673	0.37374	0.15832	0.16988	0.00000
Casilla 3	0.42697	0.13618	0.00000	0.00000	0.00000
Casilla 4	0.00000	0.00000	0.00000	0.33178	0.00000
Casilla 5	0.00000	0.02936	0.24691	0.07493	0.00000
Casilla 6	0.00000	0.00000	0.00000	0.00000	0.00000
Casilla 7	0.61969	0.05016	0.10201	0.11849	0.00000
Casilla 8	0.00000	0.00000	0.00000	0.00000	-0.99999
Casilla 9	0.00000	0.00000	0.11337	0.00000	0.00000
Casilla 10	0.00000	0.34270	0.00000	0.00000	0.00000
Casilla 11	0.43465	0.85311	0.32881	0.31142	0.00000
Casilla 12	0.00000	0.00000	0.00000	0.00000	0.00000

Nota: para la identificación de casillas se ha utilizado la tabla del apartado 10 del ejercicio 1.

6. CONCLUSIONES

Con el desarrollo de la práctica hemos conseguido mejorar nuestro conocimiento y técnica respecto al uso del aprendizaje por refuerzo. Dado que se trata de un tutorial relativamente sencillo no hemos tenido dificultades para la realización del mismo; pero sí queremos destacar que el proceso que más tiempo nos ha llevado ha sido el de comprender correctamente el funcionamiento de los métodos para su posterior uso.