Aprendizaje Automático

GRADO EN INGENIERÍA INFORMÁTICA

Tutorial 1

Curso 2020/2021

Jorge Rodríguez Fraile, 100405951, Grupo 83, 100405951@alumnos.uc3m.es
Carlos Rubio Olivares, 100405834, Grupo 83, 100405834@alumnos.uc3m.es

Índice

Pregunta 1

En cuanto a la interfaz tenemos diferentes números en la parte inferior de la pantalla que, de izquierda a derecha representan la puntuación y la distancia a cada uno de los fantasmas (el número de color naranja nos dice la distancia de Pac-man al fantasma naranja, y así sucesivamente)
Por otro lado, cabe recalcar que, cuando Pac-man come un punto en el mapa, en la terminal aparece la palabra 'REMOVE', en este caso, como solo hay un punto, sólo aparece una vez.

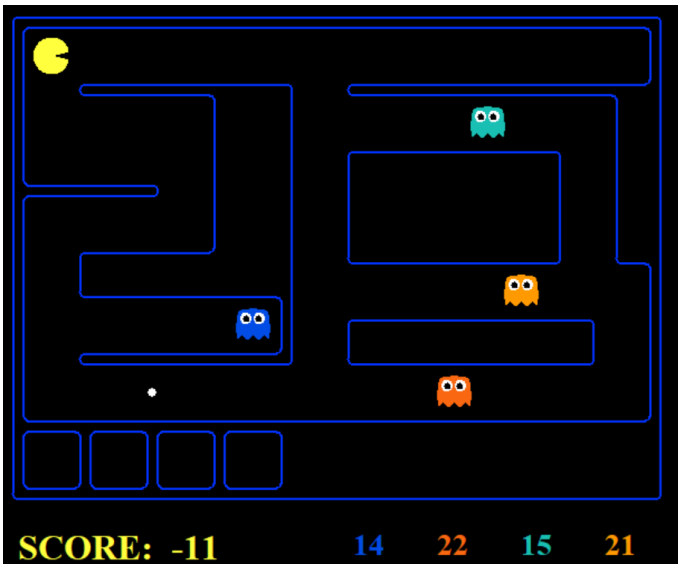
La posición que ocupa Pac-man se indica en el layout con la letra P, que en el escenario por defecto (oneHunt.lay) se encuentra en la posición (6,14) contando desde la esquina superior izquierda.

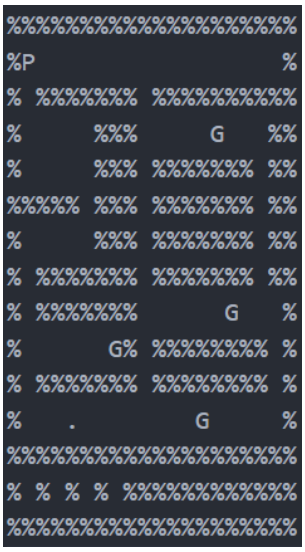
Pregunta 2

Las distancias a cada uno de los fantasmas es un dato bastante útil, ya que, en este caso, comerse todos los fantasmas es el objetivo del juego, aunque, por otro lado, también sería útil un contador con los fantasmas y puntos restantes en el tablero que Pac-man pueda comer. Si se decide cambiar la finalidad del juego a la del original (que en este caso es comer todos los puntos del tablero y evitar ser eliminado por los fantasmas) el uso de este tipo de datos sería de gran ayuda.

Pregunta 3

Los tableros tienen 5 tipos de elementos:
Muros: Se definen con '%', son las posiciones a las que no se pueden desplazar Pac-man ni los fantasmas.
Espacio de movimiento: Son posiciones vacías ' ', sin ningún símbolo o carácter, por estas posiciones pueden transitar tanto Pac-man, como los fantasmas.
Fantasmas: Se define la posición que ocuparían los fantasmas con la letra 'G', si se selecciona un determinado número de estos.
Puntos: En este caso, se diferencian los puntos normales, representados por '.' y los puntos grandes (que en el juego original dan a Pac-man invencibilidad) y vienen dados por 'o'.
Pac-man: Se indica con la letra 'P' la posición inicial del Pac-man.





En la parte inferior se han dejado huecos para que aparezcan los fantasmas muertos.

Pregunta 4

Cuando ejecutamos este comando se nos muestra en terminal cada 'tick' o turno con todos los datos que se han recogido sobre este:

Tablero: Tenemos su ancho y largo, que es estático durante todos los turnos de la partida.

Posición de Pac-Man: Tupla (x,y) que representa la posición del jugador en el tablero. La esquina inferior izquierda es el punto (0,0).

Legal actions: Tupla de los diversos movimientos que podemos realizar en ese turno en concreto. Las acciones vienen dadas por los cuatro puntos cardinales y 'Stop'.

Pac-Man direction: Dirección a la que está mirando Pac-man, puede ser cualquiera de los 4 puntos cardinales.

Número de fantasmas: Autoexplicativo, la cantidad de fantasmas que aparecen en el tablero.

Living Ghosts: Tupla booleana indicando 'True' si el fantasma está 'vivo' en el tablero y 'False' si ya ha sido comido por Pac-man. La posición 0 de la lista indica a Pac-man, que siempre es False.

Ghosts-positions: Conjunto de tuplas que definen la posición de cada fantasma.

Ghosts-directions: Dirección a la que están mirando los fantasmas, en este caso, siempre están puestas en 'Stop'.

Ghosts-distances: Distancia Manhattan entre Pac-man y cada uno de los fantasmas.

Pac dots: Nº de puntos restantes en el tablero.

Distance nearest pac dots: Distancia Manhattan al punto más cercano disponible en el tablero.

Map: Mapa booleano que muestra con 'T' los muros y 'F' el espacio en el que el jugador puede moverse.

Score: Puntuación del jugador en ese turno.

Pregunta 5

Los parámetros que hemos considerado relevantes que almacenar en cada iteración han sido:

Las coordenadas de Pac-man en el tablero, primero la x y después la y.

getPacmanPosition()[0] y getPacmanPosition()[1]

El número de fantasmas vivos, que no se han comido. getLivingGhosts

La distancia de Manhattan al fantasma vivo más cercano. ghostDistances

El número de iteración del estado. countActions

Formato:

pacmanx,pacmany,numFantasmasVivos,distanciaMinimaAFantasma,numeroIteracion

Ejemplo del estado inicial:

12,10,4,3,0

Estos datos nos aportan la información más importante para encontrar una solución óptima al problema, ya que nos indica cómo de cerca nos encontramos de que no nos queden más fantasmas, en cuanto a fantasmas vivos y a como de cerca nos encontramos. Además, el número de pasos nos permite analizar cuantos pasos nos ha llevado eliminar los distintos fantasmas.

Lo primero antes de escribir en el fichero de salida es abrirlo, para ello usamos la función open de os, indicando donde y como queremos el fichero "./infofile.txt" y el tipo de operación que haremos "a", que se refiere a append (para que escriba a continuación de lo que haya).
Ahora sí, escribimos en el fichero de salida los datos que devuelve la función printLineData para ese estado del juego y añadimos un salto de línea. Al encontrarse dentro del bucle principal de game.py es necesario indicar que queremos llamar al agente de Pac-man y no al de los fantasmas, para conseguirlo se usa self.agents[0], y pasamos como parámetro es estado de juego actual.
Cuando hayamos acabado la escritura liberamos el descriptor del fichero.

Pregunta 6

Al principio, se había pensado en implementar una búsqueda como A* , BFS, pero se requería de la ayuda de queues y árboles, algo para lo que se necesitaría algo más de tiempo y que obviamente es más complicado de implementar.
Por todo esto, hemos decidido usar una implementación más simple, y nos hemos ayudado de dos fuentes de información básicas: la primera es que conocemos el mapa en el que vamos a ejecutar el agente de búsqueda, y la segunda es la posición del fantasma más cercano.

Para obtener la tupla (x,y) del fantasma más cercano simplemente hemos accedido a la lista de fantasmas vivos y hemos recorrido la lista de las distancias de Pac-Man a dichos fantasmas. Si la distancia que leemos recorriendo la lista es menor que la que tenemos guardada la sustituimos y también nos quedamos con el fantasma que es.
Al terminar el bucle accedemos a la posición del fantasma que haya resultado y la almacenamos. Una vez hecho esto primero ajustamos la posición del Pac-Man con el fantasma en el eje de coordenadas X y después en el eje Y.

El funcionamiento del comportamiento es bastante básico, y nos va a dar máximos y mínimos locales, es decir, en laberintos más complejos es posible que no encuentre la solución, pero como comienzo es algo que puede servir.
Ahora, se procede a comparar el comportamiento de este agente con fantasmas estáticos y en movimiento aleatorio:

- En cuanto al movimiento estático el número de turnos (o ticks) es siempre de 24, debido a la simplicidad del problema, ya que las distancias son siempre las mismas y no tendrá que modificar su camino en ninguna de las iteraciones.
- Con movimiento aleatorio de fantasmas tenemos 38'2 turnos de media, el resultado se explica por sí solo, al cambiar las distancias constantemente Pac-man tendrá que modificar su camino muy a menudo, incluso teniendo que cambiar de sentido, perdiendo así algunos turnos.

Pregunta 7

El aprendizaje automático puede hacer que a base de resolver un gran número de veces un mismo problema (pudiendo fallar en el proceso) sacar un 'algoritmo' o manera de resolver el problema de manera eficiente y óptima. Pongamos el ejemplo del mapa de Pac-man original, donde de nuevo, el objetivo será el de eliminar a todos los fantasmas. Si jugamos nosotros podemos pensar que lo más óptimo es empezar por las esquinas, o ir al centro, las posibilidades son muchas. El aprendizaje automático toma todas estas opciones y las evalúa, ve que ha salido mal y que puede mejorar hasta tener una ejecución casi perfecta, de esta manera podemos optimizar el problema de búsqueda de Pac-man al máximo.

Conclusiones y dificultades encontradas

En definitiva, este trabajo nos ha ayudado a entender mejor la estructura del código en Python y a entrar en contacto con el aprendizaje automático. Ha sido bastante complicado entender la función de cada una de las clases y qué llamaba a cada una, pero con constancia hemos podido entender el 'esqueleto' del código y continuar con el Tutorial 1. Por otro lado, nos ha resultado interesante ver cómo funcionaban los agentes del Pac-Man y de los fantasmas, ya que con pocas líneas de código se podía controlar el movimiento de los elementos en el tablero, e investigar con ello ha sido uno de los mayores 'pros' del trabajo.