

Doble Grado en Ingeniería Informática y ADE  
(Campus de Colmenarejo)

(Curso 2021/2022)

## **APRENDIZAJE AUTOMÁTICO**

### **PRACTICA 2: REINFORCEMENT LEARNING**

**Ricardo Prieto Álvarez - 100386267**

## 1. *Introducción*

A lo largo de esta memoria se irán cubriendo un total de 5 puntos que analizan y definen el proceso de desarrollo del agente pacman basado en el algoritmo Q-Learning con estrategia  $\epsilon$ -Greedy.

El primer aspecto a tratar corresponde a la selección de los atributos que definen el estado del juego, además de la función de refuerzo relacionada con las acciones a tomar dentro del entorno de entrenamiento.

A continuación, se realizará una explicación del tratamiento de datos elaborado con una breve descripción del código nuevo implementado para conseguir el objetivo descrito en el primer punto.

El tercer punto refleja la evolución del agente pacman construido, desde su concepción hasta la elección final de la Q-table definitiva. En esta sección se relatan todos los conflictos, problemas y dificultades encontradas y el intento para dar una solución eficiente a estas.

El siguiente punto consiste en la evaluación y comparación del agente basado en RL y el agente programado en el Tutorial 01 dados los resultados capturados una vez jugadas dos partidas en 5 mapas.

Por último, se presenta una breve conclusión que cubrirá observaciones acerca de esta práctica y prácticas anteriores, junto con un análisis de las aplicaciones factibles del RL en otros dominios.

## 2. *Selección de atributos y función de refuerzo*

En un inicio, se tomaron 3 tipos de atributos:

- Dirección al fantasma más cercano a pacman ('East', 'West', 'North', 'South')
- Distancia relativa al fantasma (Close, Mid, Far)
- Hay muro en mi dirección (1, 0)

Esto suponía un total de estados equivalente a  $4 \times 3 \times 2 = 24$ .

Finalmente, los atributos que definieron los estados definitivamente fueron:

- Dirección al fantasma más cercano a pacman ('East', 'West', 'North', 'South', 'North-East', 'North-West', 'South-West', 'South-East')
- Distancia relativa al fantasma (Close, Mid, Far)
- Tipo de muro (1, 2, ..., 15)

Esto equivale a un total de estados de  $8 \times 3 \times 15 = 360$ .

La función de refuerzo está diseñada de tal manera que represente la diferencia de score, es decir, cuando pacman anda sin comer ningún fantasma, gana -1, en el momento en el que come un fantasma gana  $200 - 1 = 199$ . Esta función tenía un problema en su inicio, ya que la diferencia de score para la acción 'Stop' era 0, por lo tanto, después de los entrenamientos, pacman decidía quedarse quieto constantemente. Para ello se incluyó una recompensa de -2 para las acciones 'Stop'.

Los entrenamientos se iniciarán con los siguientes valores para los parámetros  $\alpha$ ,  $\epsilon$  y  $\gamma$ :

- $\epsilon = 0.9$ . Se irá reduciendo 0.1 a medida que se realizan más episodios para priorizar la exploración al inicio.
- $\alpha = 0.4$
- $\gamma = 0.5$

### 3. Tratamiento de los datos y descripción del código

A la hora de obtener los elementos del estado del juego que definirán cada uno de los estados del algoritmo Q-Learning, se han desarrollado 3 métodos auxiliares. Estos pertenecen a la clase `GameState` de `busters.py`.

El primer método es `getDirectionNearestGhost()` y su función es poder determinar la dirección del fantasma más cercano al pacman retornando una cadena de tipo `String` con una de las 8 opciones posibles: {'East', 'West', 'North', 'South', 'North-East', 'North-West', 'South-West', 'South-East'}. Para poder realizar esto, primero determina la posición del fantasma más cercano al pacman, después calcula las distancias del eje de ordenadas y el eje de abscisas. Por ejemplo, si la distancia en x es positiva y la distancia en y equivale a 0, el método devuelve 'East'. Si por ejemplo el método devuelve 'South-East' quiere decir que la distancia en x es positiva y, sin embargo, la distancia en y es negativa.

El segundo método, `getRelativeDistanceNearestGhost()` consiste en determinar la distancia relativa entre el fantasma más cercano y el pacman, para ello obtiene la máxima distancia Manhattan presente en el tablero y la divide en 3. Si por ejemplo la distancia entre el fantasma más cercano y el pacman se encuentra en el intervalo entre  $\text{maxDist}/3$  y  $2*\text{maxDist}/3$ , el resultado es 'Mid', es decir, la distancia relativa es media.

El tercer método es `getTypeOfWall()`, y se encarga de determinar cuál es el tipo de muro en el que se encuentra pacman en un momento determinado. Para esto, se genera una lista de 4 elementos booleanos que representan los muros alrededor de Pacman, de tal manera que cada combinación de estos corresponde con un muro. Esto se explica con mayor profundidad en el siguiente apartado.

Además de los métodos previamente explicados, se ha desarrollado un módulo que contiene un único método llamado `createStates()` que se encarga de generar una lista de tuplas que contengan todas las combinaciones posibles para los valores de los atributos seleccionados que definen los estados de la Q-table y se almacena en la variable atributo `states` de la clase `QLearningAgent`, esto es posible gracias a la compresión de listas de python. Esto facilitará determinar la fila de la tabla de valores Q por medio del método `computePosition()`.

En el método `computePosition()` se pasa el estado del juego completo, pero se necesita una tupla con los 3 valores de los atributos anteriormente descritos, para ello se genera esta misma llamando a los 3 primeros métodos explicados y se determina la fila de la Q-table obteniendo el índice que corresponde a la tupla de atributos del estado en ese momento en la lista generada a partir del método `createStates()`.






#### 4. Descripción del agente implementado y su evolución

Tras haber realizado los correspondientes entrenamientos se consiguió un agente capaz de acabar con todos los fantasmas estáticos y dinámicos para los mapas 1 y 2, sin embargo, el mayor problema encontrado surgía en los mapas 3, 4 y 5, por causa de los “pasillos” en los que se encuentran algunos enemigos. El agente realizado hasta ahora sólo tenía información genérica sobre los fantasmas, de tal manera que, pacman era incapaz de determinar cuándo sí y cuando no se encontraba con ciertos muros.






Para solucionar esto, se decidió realizar un cambio en el tercer atributo de los estados, añadiendo información del tipo de muro en el que se encuentra el agente. Para poder determinar este tipo de muro se ha desarrollado un método que clasifica una tupla de 4 elementos booleanos que determina si hay un muro en su posición Este/Oeste/Norte/Sur más inmediata. Esta tupla tendría la siguiente estructura:

(<MuroIzq>, <MuroDecha>, <MuroArriba>, <MuroAbajo>)






En cuanto a la clasificación de los muros, esto se indica en el esquema inferior:

Tipo de muro	1	2	3	4	5
Representación					

Tipo de muro	6	7	8	9	10
Representación					

Caso Práctico 2: Agente Pacman basado en Aprendizaje por refuerzo

Tipo de muro	11	12	13	14	15
Representación					

Tras realizar el entrenamiento correspondiente utilizando una estrategia  $\epsilon$ -Greedy, se encontró un conflicto en el agente. Este se provocaba debido a que un fantasma al suroeste era considerado en la dirección oeste, de tal manera que, si el agente pacman se encontraba con un enemigo posicionado exactamente al oeste (altura 0 en el eje y) y otro posicionado en el suroeste, estos eran considerados como en la misma posición. Al elaborar las pruebas con el mapa 3, no se consiguió establecer una Q-table que definiera una política óptima que resultara eficaz.

Para solucionar esto, se consideró necesario añadir un poco más de información a los estados, más específicamente a la dirección relativa al fantasma. Por lo tanto, los estados quedarían de la siguiente manera:

- Dirección al fantasma más cercano a pacman ('East', 'West', 'North', 'South', 'North-East', 'North-West', 'South-West', 'South-East')
- Distancia relativa al fantasma (Close, Mid, Far)
- Tipo de muro (1, 2, ..., 15)

Este cambio supone un total de  $8 \times 3 \times 15 = 360$  estados, posiblemente algunos de ellos no serán usados nunca, pero es la única manera que puede ayudar a evitar los conflictos en las posiciones más complicadas de los mapas.

Además de los cambios previamente explicados, se ha añadido un limitador de 1000 ticks en las partidas, de tal manera que se evite un exceso de tiempo para el entrenamiento que pueda suponer ruido para los valores de las tuplas acción-estado.

El entrenamiento en el mapa 3 fue más complicado. Para ello fue necesario tomar dos copias de este mapa, ambas con un fantasma entre muros en la esquina superior izquierda y siendo el otro fantasma el que cambiaba en función del mapa que se estaba utilizando. Si no se hacía así, el entrenamiento era infructuoso y con una duración por episodio excesiva. Aún así, tras tomar estos dos mapas como fuente de entrenamiento, al cambiar de uno a otro, los resultados eran poco satisfactorios.

Para intentar solucionar lo anteriormente mencionado, se tomó el mapa 3 original y se inició el entrenamiento con una tabla de estados que ofrecía buenos resultados a la hora de completar el mapa 3 con sólo dos fantasmas.

Se prueba a aumentar el factor de descuento a 0.7. La mejora es sustancial, siendo capaz de completar los 4 primeros mapas con fantasmas estáticos y fantasmas con movimiento aleatorio. Sin embargo, al hacer la prueba del agente con la Q-table de ese momento sobre el mapa 5, este se queda atascado en la esquina inferior derecha, para evitar esto, es necesario más

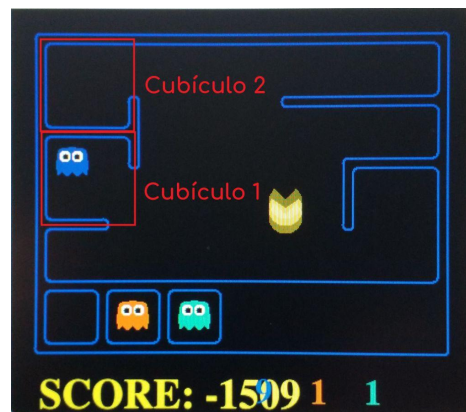
## Caso Práctico 2: Agente Pacman basado en Aprendizaje por refuerzo

entrenamiento, por lo que se guarda la Q-table que se posee hasta este momento por seguridad y se comienzan las pruebas con un  $\gamma$  de 0.8 y un valor de  $\epsilon$  equivalente a 0.5.

A pesar de estos cambios, no consigue terminar los episodios, por lo que se decide entrenar para 3 tipos de copias del mapa 5, cada una de las copias tiene a uno de los fantasmas en una de sus posiciones concretas. Se intuye que esto puede dar información para cada caso.

Una vez elaborada la experimentación que anteriormente se explicó, fue imposible conseguir que pacman terminara la partida satisfactoriamente. Esto se debe a que, los fantasmas de la esquina superior izquierda sólo los separa un muro y, dados los atributos de los estados, esto supone un conflicto.

Un ejemplo de esto: si pacman determina que el fantasma más cercano está al ‘Noroeste’ y ‘Cerca’, consigue aprender a terminar con él adentrándose al cubículo 1, una vez dentro detecta que el fantasma más cercano está al ‘Norte’ y la distancia relativa es ‘Cerca’, por lo tanto, sabe que debe retirarse, sin embargo, al realizar la acción, vuelve a tener en la información de su estado que el fantasma más cercano se encuentra al ‘Noroeste’ y ‘Cerca’ de manera relativa, por lo que, vuelve a adentrarse al cubículo 1, a pesar de que el fantasma que queda está en el cubículo 2, todo esto para los tipos de muros que determinan el espacio de los cubículos 1 y 2. Esto sucede de la misma manera para el caso en el que pacman se adentra primero al cubículo 2.



Aunque se cambie el atributo que mide la distancia relativa al fantasma más cercano y se añada más información, este problema seguiría presentándose, ya que la diferencia de distancia entre estos dos fantasmas es mínima y los tipos de muros no ofrecen suficiente información a los estados.

Además de esto, se creó un mapa alternativo al mapa 5 en el cual se eliminaron todos los paddots presentes, ya que, al no tener atributos que identifiquen estos mismos en los estados, supone ruido para el aprendizaje. Básicamente está imitando como jugaría yo personalmente, es decir, priorizar comer fantasmas para obtener mayor puntuación.

Cabe destacar que una vez el agente pacman era capaz de obtener buenos resultados para el mapa 5, perdió especialización para los mapas anteriores. Por lo tanto, se dispone de dos agentes distintos, uno realmente eficaz para fantasmas estáticos y en movimiento en los mapas 1 al 4, y para fantasmas en movimientos en el mapa 5, y otro capaz de ofrecer resultados suficientes para los mapas 1 a 5 para fantasmas en movimiento. Este último agente es el resultado de haber realizado el entrenamiento por cada uno de los mapas hasta terminar con el 5º, donde, parece que en el momento en el que consigue resolver este último mapa, “olvida” cómo jugar los 4 anteriores.

Finalmente, a la hora de seleccionar el agente definitivo, se ha optado por el agente que tiene la habilidad de terminar con buenos resultados para fantasmas estáticos y dinámicos en los 4

Caso Práctico 2: Agente Pacman basado en Aprendizaje por refuerzo

primeros mapas, y para fantasmas dinámicos en el mapa 5. Dada esta elección, se procede a realizar la evaluación de este jugador pacman en el próximo apartado.

## 5. Resultados obtenidos

### Análisis QLearningAgent

Tipo de Fantasmas	Laberinto	Partida	Fantasmas comidos	Puntos de comida	Score	Ticks
Estáticos	labAA1	partida 1	1	0	183	34
		partida 2	1	0	183	34
	labAA2	partida 1	2	0	383	51
		partida 2	2	0	383	51
	labAA3	partida 1	3	0	569	124
		partida 2	3	0	569	124
	labAA4	partida 1	3	0	562	152
		partida 2	3	0	550	200
	labAA5	partida 1	#NO COMPLETO#	#NO COMPLETO#	#NO COMPLETO#	#NO COMPLETO#
		partida 2	#NO COMPLETO#	#NO COMPLETO#	#NO COMPLETO#	#NO COMPLETO#
Dinámicos	labAA1	partida 1	1	0	182	36
		partida 2	1	0	181	38
	labAA2	partida 1	2	0	377	69
		partida 2	2	0	383	52
	labAA3	partida 1	3	0	575	100
		partida 2	3	0	569	124
	labAA4	partida 1	3	0	576	96
		partida 2	3	0	560	160
	labAA5	partida 1	3	200	529	284
		partida 2	3	300	500	400

### Análisis agente Tutorial 01

Tipo de Fantasmas	Laberinto	Partida	Fantasmas comidos	Puntos de comida	Score	Ticks
Estáticos	labAA1	partida 1	1	0	183	34
		partida 2	1	0	183	34
	labAA2	partida 1	2	0	383	51
		partida 2	2	0	383	51
	labAA3	partida 1	3	0	569	124
		partida 2	3	0	569	124
	labAA4	partida 1	3	200	565	140
		partida 2	3	200	565	140
	labAA5	partida 1	#NO COMPLETO#	#NO COMPLETO#	#NO COMPLETO#	#NO COMPLETO#
		partida 2	#NO COMPLETO#	#NO COMPLETO#	#NO COMPLETO#	#NO COMPLETO#
Dinámicos	labAA1	partida 1	1	0	184	32
		partida 2	1	0	185	30
	labAA2	partida 1	2	0	375	75
		partida 2	2	0	381	58
	labAA3	partida 1	3	0	581	76
		partida 2	3	0	565	140
	labAA4	partida 1	3	0	574	104
		partida 2	3	0	583	68
	labAA5	partida 1	3	200	563	148
		partida 2	3	300	544	224



Caso Práctico 2: Agente Pacman basado en Aprendizaje por refuerzo

		Score Medio	
Tipo de fantasma	Laberinto	QLearning	Tutorial 01
Estático	labAA1	183	183
	labAA2	383	383
	labAA3	569	569
	labAA4	556	565
	labAA5	#NO COMPLETO#	#NO COMPLETO#
Dinámico	labAA1	181.5	184.5
	labAA2	380	378
	labAA3	572	573
	labAA4	568	578.5
	labAA5	514.5	553.5

Como se puede observar, marcados de color verde se encuentran los valores que han obtenido mayor resultado en cuanto a Score Medio. Parece que el agente realizado en el tutorial 01 es vencedor en un entorno donde los fantasmas se encuentran en movimiento, sin embargo, la diferencia entre puntuaciones no es elevada, si no todo lo contrario, ya que la diferencia máxima de Score entre los dos agentes ha sido sólo de 39 puntos. Por lo tanto, ambos agentes logran sus objetivos de manera muy similar.

La posible causa por la que el agente implementado con Q-Learning no ha conseguido superar el mapa 5 con fantasmas estáticos es la falta de información en el estado. Una mayor información acerca de este puede resolver los conflictos de aprendizaje entre mapas.

Un posible atributo a mejorar sería la distancia relativa al fantasma más cercano, proporcionando más categorías que las 3 utilizadas en esta práctica. Esto también supone un aumento sustancial en el número de estados, de los cuales, muchos de ellos no serán visitados.

Otra idea, la cual se intuye como la que más potencial posee para resolver el problema, sería que tuviera en consideración a 2 fantasmas más cercanos por orden de cercanía en vez de 1, de tal manera que en el caso del mapa 5, el agente sabría identificar con mayor precisión su estado, teniendo a un fantasma más cerca que otro.

## 6. Conclusiones

Esta práctica la he considerado más sencilla que la primera ya que exigía un menor tratamiento inicial de los datos. En la práctica 1 fue realmente tedioso realizar un elevado número de pruebas para poder determinar qué modelo de los estudiados devolvía los mejores resultados para cada combinación de atributos y sus transformaciones.

Respecto a la práctica 2, esto no fue necesario, el único tratamiento realizado ha sido dar forma a la información acerca de un estado determinado del juego para que pudiera construirse la Q-table. Finalmente, en cuanto a problemas encontrados durante la elaboración de esta última práctica, se debe destacar la falta de conocimiento profundo del algoritmo de Q-Learning que pudiera ayudar a la hora de escoger los parámetros  $\epsilon$ ,  $\alpha$  y  $\gamma$  indicados para que el agente pacman construido obtuviera resultados satisfactorios para todos los mapas de estudio. Dada esta situación, la elección de los valores para dichos parámetros se ha basado en el ejercicio de prueba y error constante.

Otro aspecto a destacar que provocó dificultades fue el entrenamiento con el mapa 5, ya que una vez que el agente dominaba los 4 primeros mapas y se comenzaba el entrenamiento del número 5, el agente parecía “olvidar” los primeros tableros dando lugar a una situación donde resultó complicado establecer un pacman genérico para todos los 5 entornos.

Finalmente, considero que esta última práctica ha resultado la más entretenida de la asignatura. Agradezco haber realizado los proyectos propuestos este curso ya que me han otorgado una visión más práctica de los algoritmos estudiados en la teoría y creo que esto es fundamental para una disciplina como esta.

Haciendo una reflexión rápida acerca de las posibles aplicaciones de los algoritmos de Aprendizaje por Refuerzo en dominios distintos a los del enfoque de la práctica 2, podríamos encontrar el caso del Deep RL, donde se mezcla Deep Neural Networks con Reinforcement Learning lo que puede ser de mucha ayuda para sistemas de navegación de coches autónomos o drones, especialización de las decisiones comerciales y de marketing para cada cliente, optimización sobre los parámetros de experimentación en reacciones químicas, encontrando la política óptima que disminuya el tiempo utilizado ó poder tener un control eficiente del consumo de recursos energéticos de ciertos sistemas, como el algoritmo DeepMind utilizado por Google.