

UNIVERSIDAD CARLOS III DE MADRID



APRENDIZAJE AUTOMÁTICO

GRADO EN INGENIERÍA INFORMÁTICA

GRUPO 83

---

## Práctica 3: Aprendizaje por refuerzo

---

*Autores:*

Daniel MEDINA GARCÍA  
alejandro RODRÍGUEZ SALAMANCA

9 de mayo de 2016

# Índice

1. Diagrama con los diferentes pasos de la tarea de aprendizaje realizada en la práctica	3
2. Generación del espacio de estados	3
3. Generación de la tabla Q	4
4. Agente automático	4
5. Evaluación	4
6. Agente final	4
7. Conclusiones	5

## Introducción

# 1. Diagrama con los diferentes pasos de la tarea de aprendizaje realizada en la práctica

Ni puta idea de esto, pero viene en el guión. Entiendo que será algo como Diseñar el espacio de estados - programar el agente - aprendizaje online - generar el agente final

## 2. Generación del espacio de estados

Hablar de nuestro estado (dirección en la que se encuentra el fantasma más cercano y última dirección en la que se ha movido Pacman). Justificación del conjunto de atributos final elegido y su rango para la definición de los estados. Se debe indicar la evolución histórica de los agentes implementados hasta llegar al agente final, donde cada uno de estos agentes puede tener un conjunto de atributos diferente y destacar las diferencias con el conjunto final.

Espacios usados - 1. Dirección del fantasma más cercano, si hay muros alrededor. 2. El usado (y que ha perdido) Dani 3. Dirección del fantasma mas cercano y última dirección en la que se ha movido Pacman

---

TODO: Intro de la sección

Primero, probamos un espacio de estados formado por la dirección en la que se encuentra el fantasma más cercano, y si existen muros alrededor de PacMan, esto es, si el fantasma más cercano se encuentra por encima de PacMan y a la izquierda de éste, y hay un muro justo debajo, el estado sería `North,West,False,False,True,False`. Este espacio de estados nos daba un total de 256 estados (cuatro posibles valores de los dos primeros atributos, y dos posibles valores de los cuatro siguientes). Tras ejecutar un agente con este espacio de estados, nos dimos cuenta de que la información relativa a los muros no aportaba información relevante puesto que en nuestro agente sólo elegimos una acción de entre las legales, y aquellas direcciones en las que se encuentra un muro no están incluidas en dicho conjunto.

Después, probamos a eliminar dicha información sobrante y nos quedamos con un conjunto de estados compuesto por DANI, AQUÍ TE TOCA EXPLICAR EL QUE HAS ELEGIDO TÚ.

Finalmente, el conjunto de estados elegido fue el formado por la dirección del fantasma más cercano y la dirección del último movimiento realizado por PacMan. Esto nos da un total de 64 posibles estados. Estos atributos fueron elegidos por dos razones:

- Simplifica mucho el conjunto de estados - Al tener en cuenta únicamente tres atributos, el conjunto de estados queda muy reducido, lo que era uno de los objetivos que tratábamos de cumplir, puesto que un conjunto de estados muy grande puede contener estados que no se lleguen a alcanzar nunca.
- Aporta información necesaria y genérica - Otro de los requisitos que hemos intentado cumplir ha sido obtener un agente genérico que no esté *atado* a ningún mapa, y que sea capaz de desenvolverse con soltura en diversos escenarios. Por ello, los atributos escogidos son independientes del mapa, pero aportan la información que necesita PacMan para determinar hacia qué dirección debe moverse.

### 3. Generación de la tabla Q

Aprendizaje online con epsilon-greedy. Rewards: 10 si se acerca al fantasma más cercano, 199 si se come un fantasma, -1 si no hace algo —

Para generar la tabla Q hemos hecho uso de la técnica conocida como *online learning*. Se trata de un método mediante el cual los datos se encuentran disponibles en orden secuencial (se generan mientras PacMan está jugando partidas) y son usados para actualizar la mejor predicción hasta el momento de la tabla Q.

En cuanto a los refuerzos, hemos decidido dividirlos en tres tipos:

- PacMan se come un fantasma - Cuando PacMan se come un fantasma, recibe un refuerzo positivo con un valor de 199 (coincide con la diferencia de puntuación entre el estado actual y el próximo estado al que se transiciona)-
- PacMan se acerca al fantasma más cercano - En este caso, entendemos que PacMan está realizando un movimiento beneficioso, por el cual recibe un refuerzo positivo con valor 10.
- PacMan se aleja del fantasma más cercano - Si PacMan está *persiguiendo* a un fantasma, y su distancia a éste aumenta, el refuerzo que se da es negativo, cuyo valor es de -1.

Además, se ha usado la técnica  $\epsilon$ -greedy, por la cual nuestro agente toma la acción devuelta por la política obtenida hasta el momento un  $\epsilon$  % de las veces, y  $(1-\epsilon)$  % una acción aleatoria, para que el mapa sea explorado y se prueben diversas acciones para un mismo estado, con el objetivo de quedarnos con la que se considere mejor.

### 4. Agente automático

Hablar de cómo se ha implementado el agente

— Nuestro agente ha sido implementado sobre la clase `PacmanQAgent`, la cual extiende de `QLearningAgent`, por lo que trae cierto comportamiento predefinido para que resulte más sencillo implementar un agente utilizando la técnica de Q-Learning.

Primero, se inicializan los valores de `alpha`, `epsilon` y `discount` en el constructor de la clase, para ser usados posteriormente en la actualización del valor en la tabla Q. Además, se carga la tabla Q, que se encuentra en un fichero llamado `qtable.txt`. Esta tabla Q es almacenada en un objeto de tipo `Counter`, que se encuentra en `utils.py`. Dicho objeto actúa como un diccionario, en el que las claves serán tuplas estado-acción, y el valor, el correspondiente al que retorne la función Q.

Nuestra política es retornada por el método `getPolicy`. Este método recibe un estado, e itera sobre todas las acciones legales, devolviendo aquella acción cuyo valor en la tabla es el máximo para el estado dado.

Si una partida supera los 800 turnos, ésta acaba, pues entendemos que nuestra tabla no se está actualizando con valores que ayuden a nuestro agente a jugar mejor.

## 5. Evaluación

1.  $\alpha = 0.3$ ,  $\text{discount} = 0.8$ ,  $\epsilon = 0.8$  2.  $\alpha = 0.6$ ,  $\text{discount} = 0.4$ ,  $\epsilon = 0.7$

Para la evaluación no queremos actualizar la tabla ( $\alpha = 0$ ) y no tenemos en cuenta  $\epsilon$  (no random move)

Evaluar en default y finalMap?

## 6. Agente final

(El que mejores resultados obtenga de lo anterior) Descripción y análisis de los resultados producidos por el agente final implementado tras la evaluación. En este apartado es importante describir por que se cree que ha funcionado bien el agente seleccionado (si es este el caso). En cualquier caso, se deberán describir posibles mejoras que se pueden hacer para aumentar su rendimiento. Además, es importante responder a la siguiente cuestión:

¿El agente final desarrollado es capaz de superar en rendimiento a los agentes utilizados en la generación de las tuplas de experiencia? Justificar la respuesta tanto en caso afirmativo como en caso contrario.

## 7. Conclusiones

Conclusiones sobre la tarea a realizar. Apreciaciones más generales sobre las prácticas de la asignatura como: para que pueden ser útiles los modelos obtenidos y si se os ocurren otros dominios en los que aplicar aprendizaje automático, etc. • Descripción de los problemas encontrados a la hora de realizar esta práctica. • Comentarios personales. Opinión acerca de la práctica. Dificultades encontradas, críticas, etc