Aprendizaje por refuerzo aplicado al juego del Super Mario de la competición Mario AI

Isaac Pérez Borrero
Escuela Técnica Superior de Ingeniería
Universidad de Huelva

Objetivos

1. Crear un agente que aprenda a jugar al Super Mario aplicando un algoritmo de aprendizaje por refuerzo.
2. Implementar el algoritmo *Q-learning* [1] para desarrollar el aprendizaje.
3. Elegir los mejores valores para los parámetros del algoritmo, con el objetivo de que el agente aprenda correctamente.

Introducción

- Nuestro agente interactúa con un entorno donde realiza acciones y del que recibe cierta información. Esta información será empleada por el agente para realizar el proceso de aprendizaje.
- Con esta información, se definirá un estado para el agente y el cálculo de la recompensa en cada instante, ambas requeridas por el algoritmo *Q-learning*.
- El aprendizaje consistirá en decidir en un estado dado, que acción realizar. Se hará uso de una tabla (*Q-tabla*) que almacenará para cada posible estado y acción, cual es la bondad de hacer la acción en dicho estado, también llamado *Q valor*.

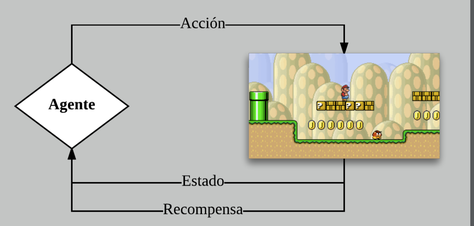

Figura 1: Esquema general del proceso de aprendizaje

- El *Q valor* para un estado s_t y una acción a_t en un instante de tiempo t, $Q(s_t, a_t)$, es calculado de la siguiente forma:

$$Q(s_t, a_t)' = \delta Q(s_t, a_t) + \alpha [r(s_t, a_t) + \gamma max_{a_t+1} Q(s_{t+1}, a_{t+1})]$$

Donde:

- δ es $\mathbf{1} - \alpha$.
- α es el factor de aprendizaje, controla la variación de Q.
- $r(s_t, a_t)$ es la recompensa para el estado s_t y la acción a_t.
- γ controla como influyen las recompensas de los estados futuros.
- $max_{a_t+1} Q(s_{t+1}, a_{t+1})$ es la acción con mayor Q para el estado posterior al actual, s_{t+1} .

Materiales

- Se hace uso de una plataforma que emula el juego del Super Mario.
  - La plataforma envía cada unidad de tiempo la información del entorno, transcurridos 40 ms, obtiene del agente la acción a realizar.
  - Disponemos de las acciones presentes en la figura 2 y de la información que nos suministra el entorno, usamos la indicada en la figura 3.

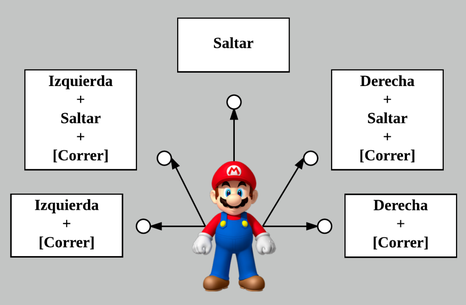

Figura 2: Acciones disponibles para el agente

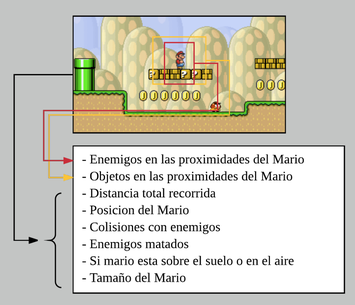

Figura 3: Información utilizada del entorno

Metodología

- A continuación se detalla como definimos nuestro estado y el cálculo de la recompensa, que será calculada como la suma de una serie de observaciones multiplicadas por un valor de ponderación.

Tamaño del Mario	Puede saltar	Obstaculos en los alrededores	Esta en el suelo	Sentido del movimiento	Enemigos en los alrededores	Esta atascado
0 - 2	0 - 1	0 - 2^{18}	0 - 1	0 - 2	0 - 2^8	0 - 1

Figura 4: Representación del estado

Observación	Ponderación
Colisión con enemigo	-100*
Mario avanza hacia la derecha	10
Mario avanza hacia la izquierda	-10
Mario atascado	-50**
Mario mata un enemigo	20
Mario esta en el aire y avanza a la derecha	70

** Si se produce, se ignoran las demás observaciones.*
*** Si se ha matado un enemigo no se penaliza el estar atascado.*

Figura 5: Información usada para calcular la recompensa

Experimentación

- Para el proceso de aprendizaje hacemos uso de una política $\epsilon - greedy$ inversa, lo que significa que con una probabilidad ϵ se hará la mejor acción disponible en la *Q-tabla* y con una probabilidad $\mathbf{1} - \epsilon$ se hará una acción aleatoria.
- Realizamos una primera experimentación para determinar los mejores valores de α, γ y ϵ. Para ello, se les asigna todas las combinaciones posibles de los siguientes valores: $\mathbf{0.2}$, $\mathbf{0.4}$, $\mathbf{0.6}$ y $\mathbf{0.8}$.
- Desarrollamos un entrenamiento de 20 ejecuciones para cada modo del Mario seguidas de un cambio de escenario hasta llegar a los 100 escenarios.
- Cada vez que se cambia de escenario se realiza una evaluación que consiste en una ejecución del Mario en el modo *Fire* sobre 100 escenarios nunca vistos.
- Del proceso de evaluación almacenamos la media en **%** de distancia recorrida sobre los escenarios. Será nuestro criterio de selección.

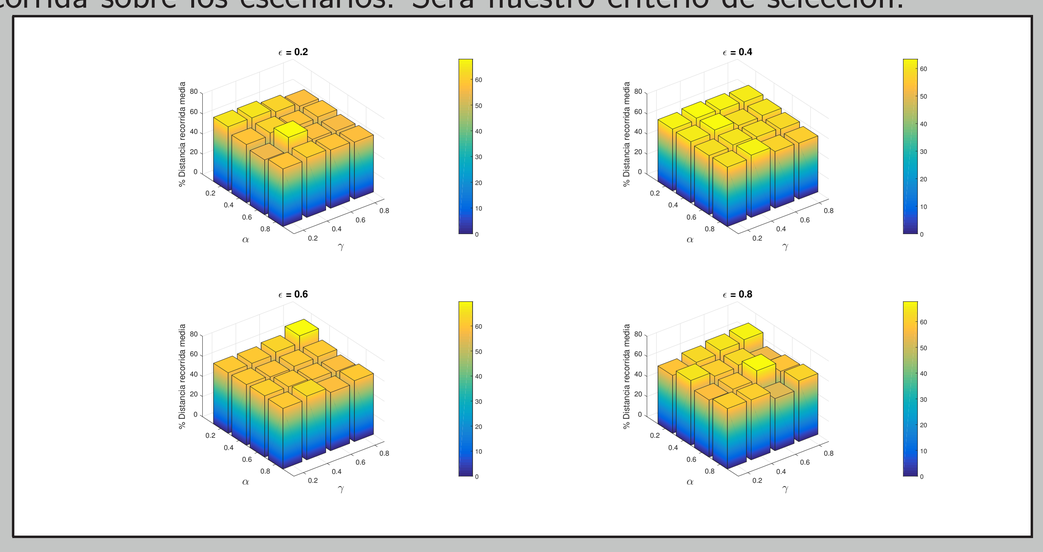

Figura 6: Resultados de la experimentación

- De la experimentación anterior nos quedamos con los 4 mejores valores, siendo los siguientes:

$$\alpha = \mathbf{0.6}\ \ \gamma = \mathbf{0.4}\ \ \epsilon = \mathbf{0.2},\ \alpha = \mathbf{0.2}\ \ \gamma = \mathbf{0.8}\ \ \epsilon = \mathbf{0.6}$$
$$\alpha = \mathbf{0.6}\ \ \gamma = \mathbf{0.6}\ \ \epsilon = \mathbf{0.8},\ \alpha = \mathbf{0.2}\ \ \gamma = \mathbf{0.8}\ \ \epsilon = \mathbf{0.8}$$

- A continuación realizamos un entrenamiento mas extensivo sobre estos valores, entrenando sobre 600 escenarios y recogiendo además el **%** de escenarios acabados con éxito durante la evaluación.

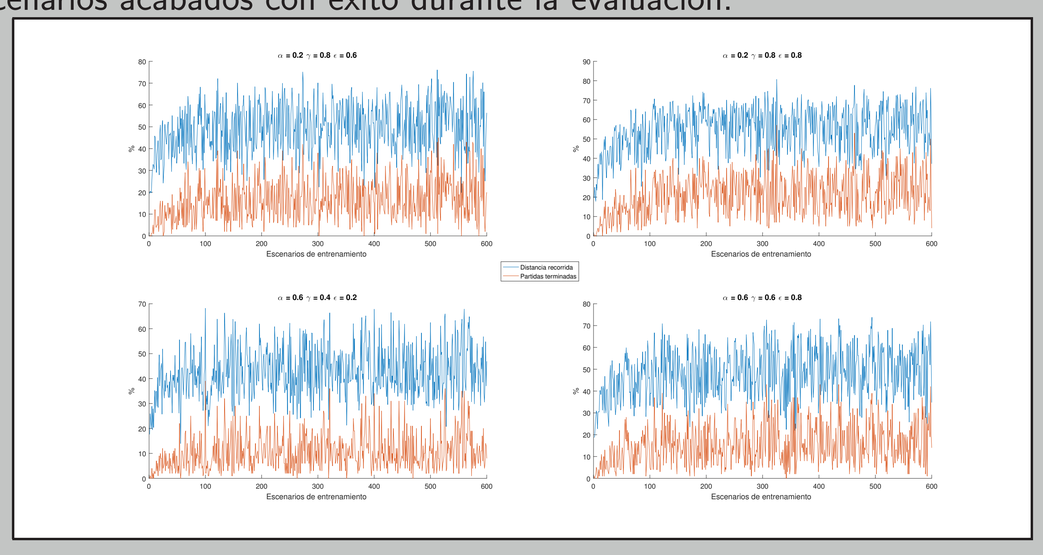

Figura 7: Resultados para los mejores parámetros

- De los resultados anteriores obtenemos que para los valores: $\alpha = \mathbf{0.2}\ \ \gamma = \mathbf{0.8}\ \ \epsilon = \mathbf{0.8}$, logramos los mejores resultados, permitiendo alcanzar un **81 %** de distancia recorrida y un **57 %** de escenarios terminados.

Conclusiones

- El algoritmo *Q-learning* nos permite obtener un buen desempeño con poca complejidad de diseño.
- Como contrapartida requiere la elección de varios parámetros que se deben elegir de forma experimental, necesitando un tiempo elevado de experimentación.
- La mayor dificultad reside en elegir una buena representación para el estado y para el cálculo de la función de recompensa.

Referencias

[1] Watkins, C.J.C.H. (1989), *Learning from Delayed Rewards*, PhD thesis, Cambridge University, Cambridge, England.

isaac.perez@alu.uhu.es