ENTRENANDO A UN AGENTE PARA JUGAR AL TRES EN RAYA MEDIANTE Q-LEARNING

Se implementará el **algoritmo Q-Learning** para que un agente aprenda a jugar al tres en raya.

Se evaluará el desempeño del agente analizando el porcentaje de partidas ganadas frente a jugador aleatorio y experto.

GENERACIÓN DE ESTADOS

1 Se generan todos los estados posibles:

A	B	C
D	E	F
G	H	I

Las variables A,B,C,D,E,F,G,H,I tendrán los valores 0,1 ó 2. El 0 representa que no hay ficha en esa casilla, el 1 que hay una ficha del jugador 1 y el 2 del jugador 2.

Asumiendo que empieza el jugador 1 observamos que en un estado dado podrá haber como máximo una ficha más del jugador 1 que del 2.

Por lo tanto para que un estado sea legal se deberá cumplir:

cantidad(unos)==cantidad(doses) o cantidad(unos)==cantidad(doses)+1

Podemos generar usando bucles todas las combinaciones desde 000000000 hasta 222222222

```
for a in range(3):
  for b in range(3):
…
```

Descartando los estados ilegales.

REPRESENTACIÓN NUMÉRICA DE UN ESTADO

Veamos un ejemplo de cómo representar un estado de forma numérica:

0	0	1
2	1	0
2	1	2

3^8	3^7	3^6	3^5	3^4	3^3	3^2	3^1	3^0
0	0	1	2	1	0	2	1	2
A	B	C	D	E	F	G	H	I

La representación del estado X

$X=3^8x0+3^7x0+3^6x1+3^5x2+3^4x1+3^3x0+3^2x2+3^1x1+3^0x2=1319$

Dado el número que representa al estado podemos obtener los valores de las variables A,B,C,D,E,F,G,H,I usando aritmética modular:

I=1319%3=2

- Ahora restamos I a X 1319-2=1317
- Lo dividimos por 3 (observa que siempre será divisible) 1317/3 =439

H=439%3=1

- Ahora restamos H a X 439-1=438
- Lo dividimos por 3 (observa que siempre será divisible) 438/3 =146

G=146%3=2

Y seguimos realizando el mismo proceso hasta obtener todos los valores.

Los Estados se identificarán consecutivamente de 0 en adelante, y guardaremos el código que lo represente en un diccionario.

Por ejemplo el estado 920 pudiera representarse con el código de representación 1319 (visto arriba) , entonces diccionario[920]=1319

También necesitaremos un **diccionario inverso**.

TABLA DE TRANSICIONES (del jugador 1)

Para generar la tabla de transiciones (asumiendo que el jugador 1 es el AGENTE a entrenar), las acciones posibles serán 9 (poner un 1 en A,B,C,D,E,F,G,H, o I.

estado\acción	0	1	2	3	4	5	6	7	8
0									
1									
..									

Para rellernar la tabla se debe seguir el siguiente proceso:

Para cada estado s hacer

Para cada acción a de ese estado hacer

- Representación r=Diccionario[s]
- probar si se posible realizar la acción a sobre r
- si es posible, calcular la representación de estado resultante k
- usar diccionario inverso para obtener número de estado
- actualizar valor de la tabla con el número de estado

Las transiciones imposibles las marcamos con -1

TABLA DE RECOMPENSAS

Podemos usar un vector R de estados con un valor 100 para estados ganadores y 0 empates y -100 derrotas. O un array bidimensional de estados y acciones.

TURNOS

Ojo, porque en un juego por turnos este concepto del estado siguiente es un poco confuso. En principio podríamos pensar que partiendo de un tablero vacío, si el algoritmo pone una X, el estado siguiente es un tablero con una casilla cubierta. Sin embargo, esto no es así. Q-learning sólo se preocupa por aprender qué hacer cuando es su turno. Tras cubrir el tablero con una X, es el turno del rival. Cuando el rival haya colocado una O, volverá a ser el turno del algoritmo. Este es el estado del tablero que debemos usar como estado siguiente en las experiencias.

REFERENCIAS

Q-learning: Aprendizaje automático por refuerzo.3 en raya.

https://rubenlopezg.wordpress.com/2015/05/12/q-learning-aprendizaje-automatico-por-refuerzo/

ENTREGA

Se entregará un único archivo PDF con el notebook en Python que incluya capturas de ejecución y celdas markdown con los pasos y explicaiones de lo realizado.

Se mostrarán gráficas del entrenamiento indicando el porcentaje de partidas ganadas, empatadas o perdidas por el Agente, jugando contra:

(5 Puntos) Un contrincante que mueva de forma aleatoria.
(5 Puntos) Un contrincante experto (que realice la mejor jugada).