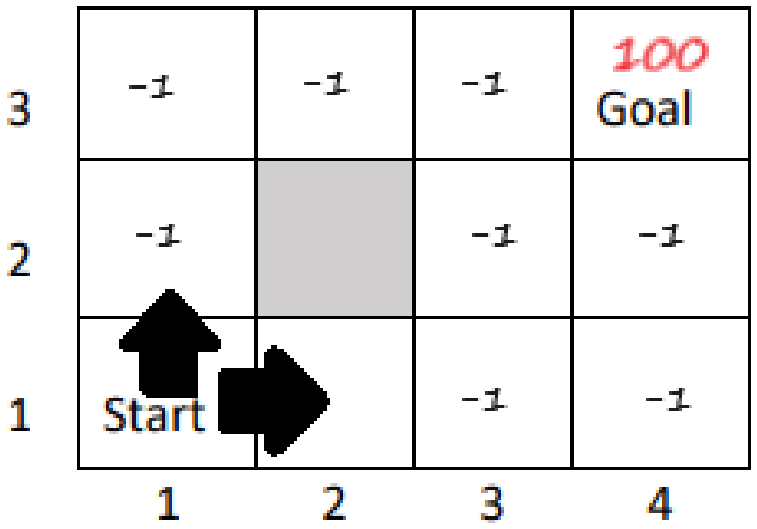
Inteligencia Artificial

Práctica 3
Reinforcement Learning



Grupo B
Xavier Marquez Cardenas
Quim Lagunas Rebollar

Introducción..... 3
Preguntas.....4
Ejercicio 1.....4
Ejercicio 2.....8
Conclusiones.....10

Introducción

En esta práctica, exploraremos los fundamentos del aprendizaje por refuerzo, centrándonos en el algoritmo de Q-learning. El Q-learning es una forma de aprendizaje automático que permite a un agente aprender a tomar decisiones secuenciales mediante la evaluación de acciones en función de su valor de utilidad esperado. En esta práctica abordaremos ejercicios que demuestran la implementación y aplicación de Q-learning, experimentamos con los diferentes paràmetros (alfa, gamma y epsilon), para ver qué valores de estos son mejores, o dan un mejor resultado y exploraremos formas de determinar si el algoritmo converge. Y finalmente veremos su aplicación en el contexto del ajedrez.

El primer ejercicio de esta práctica está implementado en el archivo P3_exercise1.py, mientras que el segundo ejercicio utiliza las classes aichess, chess, board, y piece y está implementado en la clase aichess.py.

Preguntas

Ejercicio 1

a. Implement the Q-learning algorithm to find the optimal path considering a reward of -1 everywhere except for the goal, with reward 100.

   i. (0.4 pts) Print the first, two intermediate and the final Q-table. What sequence of actions do you obtain?

   Ejemplo de ejecución:

```
First Q-table:
 [[ 0.   -0.2   0.   -0.2 ]
 [ 0.    0.   -0.2  -0.2 ]
 [ 0.    0.   -0.2  20.  ]
 [ 0.    0.    0.    0.  ]
 [-0.2  -0.36  0.    0.  ]
 [ 0.    0.    0.    0.  ]
 [-0.2  -0.2   0.    0.  ]
 [ 0.    0.    0.    0.  ]
 [-0.36  0.    0.   -0.36]
 [ 0.    0.   -0.2  -0.36]
 [-0.36  0.   -0.2  -0.2 ]
 [ 0.    0.   -0.2   0.  ]]

Second Q-table:
 [[ 0.          4.19434418  0.         62.19974464]
 [ 0.          0.         28.69644812 78.99995887]
 [ 0.          2.28064    40.75574076 99.99999461]
 [ 0.          0.          0.          0.        ]
 [48.75841512  9.99710141  0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 5.4        -0.2         0.         18.8912    ]
 [67.232      -0.2         0.          0.        ]
 [38.00110806  0.          0.          0.3095314 ]
 [ 0.          0.         11.045713   -0.83222784]
 [ 0.2736      0.         -0.488      -0.5904    ]
 [ 2.712       0.         -0.36        0.        ]]

Converged in episode  79
Final Q-table:
         UP          DOWN          LEFT         RIGHT
[[ 0.          4.19434418  0.         62.19974464]
 [ 0.          0.         28.69644812 78.99995887]
 [ 0.          2.28064    40.75574076 99.99999461]
 [ 0.          0.          0.          0.        ]
 [48.75841512  9.99710141  0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [ 5.4        -0.2         0.         18.8912    ]
 [67.232      -0.2         0.          0.        ]
 [38.00110806  0.          0.          0.3095314 ]
 [ 0.          0.         11.045713   -0.83222784]
 [ 0.2736      0.         -0.488      -0.5904    ]
 [ 2.712       0.         -0.36        0.        ]]

Optimal Path: [(2, 0), (1, 0), (0, 0), (0, 1), (0, 2), (0, 3)]
```

   La sequencia de acciones cambia según la ejecución, las 4 sequencias de acciones que obtenemos son:

   1. [(2, 0), (1, 0), (0, 0), (0, 1), (0, 2), (0, 3)]
   2. [(2, 0), (2, 1), (2, 2), (1, 2), (1, 3), (0, 3)]
   3. [(2, 0), (2, 1), (2, 2), (2, 3), (1, 3), (0, 3)]
   4. [(2, 0), (2, 1), (2, 2), (1, 2), (0, 2), (0, 3)]

   ii. (0.4 pts) After trying for a bit, what is your parameter choice for alpha, gamma and epsilon? Why?

   Alpha = 0.3 Gamma = 0.7 Epsilon = 0.9

Despues de probar con varios valores para los parametros, hemos decidido fijar estos valores para los parametros ya que son los que convergen con más rapidez, tanto si cambiamos Alpha y Gamma como si cambiamos epsilon para una probabilidad menor, a veces converge en menos episodios, pero también muchas veces necesita muchos más episodios para converger. Además con un valor de alfa más grande, convergerá antes, pero sería menos preciso, con una gamma más grande, le dariamos más valor a las recompensas, y en este caso son todas -1 (excepto la meta). Y si tuviesemos una epsilon mas pequeña, aumentaria la aleatoriedad, y debido a esto, podría tardar mas episodios en converger.

iii. (0.4 pts) How do you judge convergence of the algorithm? How long does it take to converge?
Evaluamos la convergencia segun la diferencia de medias, consideramos que converge cuando la diferencia está por debajo de cierto umbral (En nuestro caso 0,0001).

Aproximadamente converge entre 40 y 300 episodios.

b. Try implementing the more accurate reward given by:

3	-3	-2	-1	100
2	-4		-2	-1
1	-5	-4	-3	-2
	1	2	3	4

i. (0.4 pts) Answer the questions of the previous section for this case.
Ejemplo de ejecución:

```
First Q-table:
 [[ 0.    0.    0.   -0.2 ]
 [ 0.    0.    0.   -0.1 ]
 [ 0.   -0.2   0.    0.  ]
 [ 0.    0.    0.    0.  ]
 [-0.3  -0.5   0.    0.  ]
 [ 0.    0.    0.    0.  ]
 [ 0.    0.    0.   -0.1 ]
 [10.    0.    0.    0.  ]
 [-0.76  0.    0.   -0.4 ]
 [ 0.    0.   -0.5   0.  ]
 [ 0.    0.    0.    0.  ]
 [ 0.    0.    0.    0.  ]]

Second Q-table:
 [[ 0.         -0.76        0.         23.80335098]
 [ 0.          0.         -0.29117652 57.25398029]
 [ 0.          5.0315359   6.11166242 94.76652367]
 [ 0.          0.          0.          0.        ]
 [ 5.39652946 -1.82661205  0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [54.50086102 -0.3         0.         -0.1       ]
 [99.9999682  41.45961809  1.41294556  0.        ]
 [-2.76318429  0.          0.         56.52961292]
 [ 0.          0.         10.79528983 67.28036824]
 [15.64583502  0.         14.58632434 78.09791282]
 [88.99970125  0.         27.50092905  0.        ]]

Converged in episode  199
Tiempo transcurrido: 0.07845926284790039 segundos
Final Q-table:
          UP          DOWN          LEFT         RIGHT
[[ 0.         -0.76        0.         28.93446794]
 [ 0.          0.         -0.29117652 62.55278091]
 [ 0.          5.0315359   6.11166242 96.90968456]
 [ 0.          0.          0.          0.        ]
 [ 8.10308896 -1.82661205  0.          0.        ]
 [ 0.          0.          0.          0.        ]
 [62.82002253 -0.3         0.         -0.1       ]
 [99.99999772 41.45961809  5.97672849  0.        ]
 [-1.95813436  0.          0.         56.55770761]
 [ 0.          0.         14.30535911 67.28915839]
 [22.37914138  0.         18.78360281 78.09981975]
 [88.99997481  0.         31.47969326  0.        ]]

Optimal Path: [(2, 0), (2, 1), (2, 2), (2, 3), (1, 3), (0, 3)]
```

La sequencia de acciones cambia según la ejecución:

1. [(2, 0), (1, 0), (0, 0), (0, 1), (0, 2), (0, 3)]
2. [(2, 0), (2, 1), (2, 2), (2, 3), (1, 3), (0, 3)]
3. [(2, 0), (2, 1), (2, 2), (1, 2), (0, 2), (0, 3)]
4. [(2, 0), (2, 1), (2, 2), (2, 3), (1, 3), (0, 3)]

Hemos cambiado el valor de gamma por 0.9, de esta forma damos más importancia a las recompensas a largo plazo. De esta forma converge en menos episodios.
De esta forma converge aproximadamente entre 40 y 250 episodios, pero en menos episodios en un porcentaje mayor de las veces.

ii. (0.4 pts) What is the effect of the new reward function on performance?
Con las nuevas recompensas, y gamma a 0.9, en un mayor porcentaje de las veces converge en menos episodios alrededor de 100 y son menos las veces que tarda más de 150 episodios

iii. (0.4 pts) How does this relate to the search algorithms studied in P1? Could you apply one of those in this case?
Es parecido a una búsqueda informada como la que usabamos para el A* en la práctica 1, ya que la recompensa, cada vez mayor, nos da información sobre si nos estamos acercando o alejando del objetivo

c. The main novelty in RL algorithms with respect to the search algorithms in P1 is that they can be applied in stochastic environments, where the agent doesn't fully determine the outcome of its actions.

i. (0.6 pts) Drunken sailor. Your agent is now a drunken sailor trying to get to bed after a good share of whiskey and shanties: their legs don't seem to obey them all the time. Introduce stochasticity (= randomness) by enforcing that only 99% of the steps intended by the sailor are actually taken, the rest leading randomly in any other possible direction.
Para implementar esto, hemos añadido, una vez escogida la acción las siguiente líneas de código:

```
    if not drunken_sailor():
        return action
    else:
        available_actions = available_actions[available_actions != action]
        return np.random.choice(available_actions)

    return action

def drunken_sailor(ratio=0.01):
    if np.random.rand() < ratio:
        return True
    else:
        return False
```

De esta forma, un 1% de las veces, se hará aleatoriamente un movimiento, diferente al que habia sido escogido.

ii. (1 pts) Use at least one of the two rewards proposed:
   1. (0.15 pts) What is your parameter choice? Why?
   Cómo utilizamos las recompensas del ejercicio 1a, hemos utilizado los mismos valores en los parametros:

   Alpha = 0.3 Gamma = 0.7 Epsilon = 0.9

   2. (0.15 pts) Assuming the sailor is in a state that allows learning: how many drunken nights are necessary for them to master the perilous path to bed? Compare to the previous, deterministic scenario.
   Dependiendo de la ejecucion varia el número, sin embargo, podemos observar que esta en un rango superior a los dos ejercicios anteriores, converge aproximadamente en un rango entre 100 y 400 episodios.

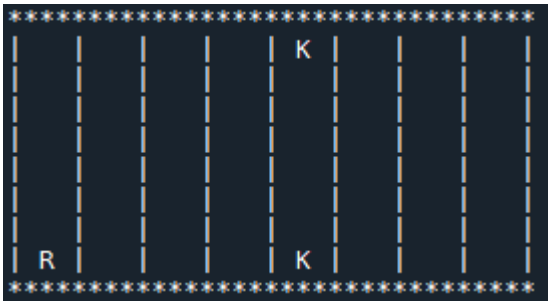
   3. (0.2 pts) What is the optimal path found? If we watched the sailor try to follow it, would they always follow the same path?
   Dependiendo de la ejecucion encontramos estos caminos optimos:
   [(2, 0), (1, 0), (0, 0), (0, 1), (0, 2), (0, 3)]
   [(2, 0), (2, 1), (2, 2), (1, 2), (1, 3), (0, 3)]
   [(2, 0), (2, 1), (2, 2), (1, 2), (0, 2), (0, 3)]
   [(2, 0), (2, 1), (2, 2), (2, 3), (1, 3), (0, 3)]

4. (0.5 pts) Could we apply one of the algorithms in P1 here? Why? Hint: think of the notions of deterministic vs random and of path vs policy.
Podriamos utilizar una heurística para determinar si estamos más cerca o no del objetivo y fijar las recompensas en función de ésta.

Ejercicio 2

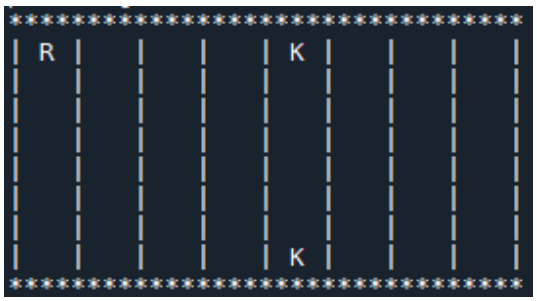
a. Adapt your Q-learning implementation to find the optimal path to a check mate considering a reward of -1 everywhere except for the goal (check mate for the whites), with reward 100.

   i. (0.5 pts) What sequence of actions do you obtain?
   Dependiendo de la ejecución obtendremos una sequencia de acciones diferente, ya que hay varios caminos para hacer jaque mate en 6 movimientos.
   Una posible sequencia sería:
   [[[7, 0, 2], [7, 4, 6]], [[0, 0, 2], [7, 4, 6]], [[6, 3, 6], [0, 0, 2]], [[5, 2, 6], [0, 0, 2]], [[4, 2, 6], [0, 0, 2]], [[3, 3, 6], [0, 0, 2]], [[2, 4, 6], [0, 0, 2]]]
   Que corresponde a los movimientos:



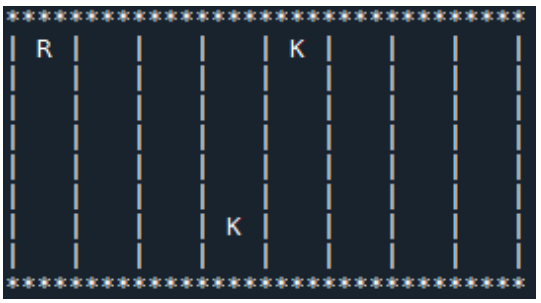
[[7,0,2],[7,4,6]]

1

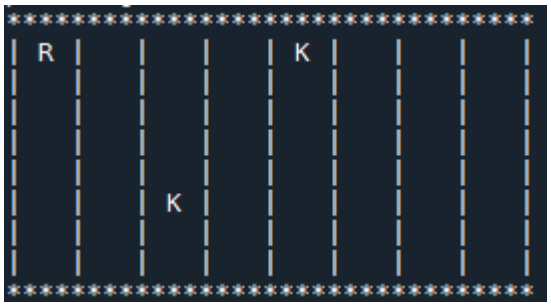
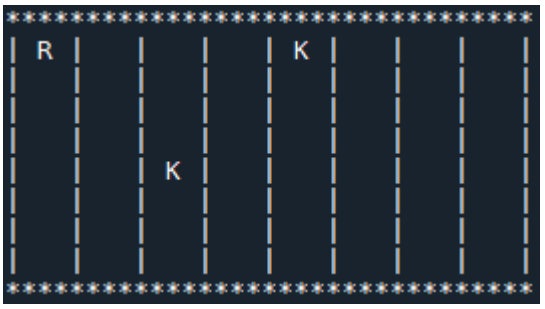
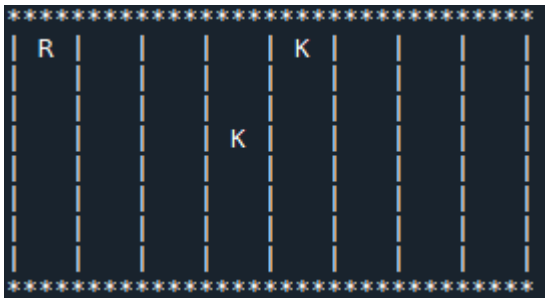
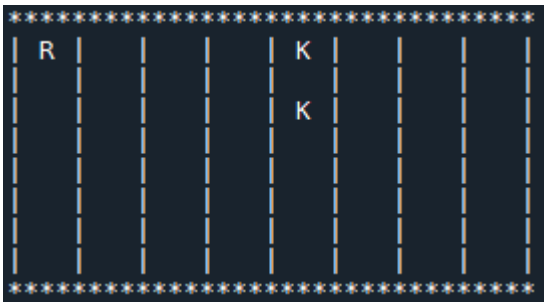


[[0,0,2],[7,4,6]]

2



[[6,3,6],[0,0,2]]

3  [[5,2,6],[0,0,2]]	**4**  [[4,2,6],[0,0,2]]
5  [[3,3,6],[0,0,2]]	**6**  [[2,4,6],[0,0,2]] **JAQUE MATE**

ii. (0.5 pts) After trying for a bit, what is your parameter choice for alpha, gamma and epsilon? Why?
Como hemos usado la misma recompensa que en el ejercicio 1 (-1 en todos menos en el jaque, que es 100). Hemos decidido usar los mismos valores:
Alpha = 0.3 Gamma = 0.7 Epsilon = 0.9

iii. (0.5 pts) How do you judge convergence of the algorithm? How long does it take to converge?
Hemos implementado un algoritmo que hace 1000 iteraciones, aun así, nuestra idea era implementar un criterio de convergencia teniendo en cuenta la diferencia entre las medias, aunque con un tolerancia mucho menor que en el ejercicio 1 ya que la q-table es mucho más grande y la media no cambia tanto como en el ejercicio 1.

b. Try now with a more sensible reward function adapted from the heuristic used for the A* search:

i. (0.5 pts) Answer the questions of the previous section for this case.

Igual que en el apartado anterior hay varias soluciones dependiendo de la ejecucion, una podria ser la mencionada en el apartado anterior:
[[[7, 0, 2], [7, 4, 6]], [[0, 0, 2], [7, 4, 6]], [[6, 3, 6], [0, 0, 2]], [[5, 2, 6], [0, 0, 2]], [[4, 2, 6], [0, 0, 2]], [[3, 3, 6], [0, 0, 2]], [[2, 4, 6], [0, 0, 2]]]
Hemos utilizado
alpha = 0.2 gamma = 0.9 epsilon = 0.9
Aumentando gamma nos hacemos que las recompensas tengan mas importancia, cómo ahora las recompensas no son todas -1 (excepto el jaque que es 100) sino que cambian según nos acercamos al jaque, necesitamos que estas cobren más importancia, para encontrar la solución antes

ii. (0.5 pts) What is the effect of the new reward function on performance?
Al utilizar unas recompensas adaptadas a partir de la heurística, el algoritmo convergerá más rapido, ya que estamos aprovechando el conocimiento para guiar el proceso de aprendizaje(Por eso cambiamos gamma, para tener mas en cuenta estas recompensas). Esto conduce a una exploración más eficiente del espacio de estados, acelerando la convergencia del Q-learning.

c. Drunken sailor. On their way to bed, our drunken sailor sees a chessboard on a table, coincidentally configured as in the previous section. They have seen the captain play with the first mate and want to give it a try, but only have a rudimentary knowledge of the rules (they know how each piece moves and what is a check mate, but not that blacks move as well).

i. (0.5 pts) Introduce stochasticity (= randomness) by enforcing that only a given percentage of the moves intended by the sailor are actually taken, the rest taken randomly from all other possibilities.
Al igual que en el ejercicio 1 esto esta representado en nuestro codigo por las lineas de código

```
    if not drunken_sailor():
        return action
    else:
        available_actions = available_actions[available_actions != action]
        return np.random.choice(available_actions)

    return action

def drunken_sailor(ratio=0.01):
    if np.random.rand() < ratio:
        return True
    else:
        return False
```

Que con una probabilidad del 99% realiza la accion que elegimos, sin embargo el 1% restante, elige una accion diferente de entre las posibles

ii. (1 pts) Use any reward you prefer:
   1. (0.5 pts) What is your parameter choice? Why?
      Ha
   2. (0.5 pts) Assuming our obsessive sailor is in a state that allows learning: how many games do they have to play before they are satisfied that they have found the best strategy and can go to bed? Compare to the previous, deterministic scenario.

3. (0.5 pts) What is the optimal path found? If we watched the sailor try to follow it, would they always follow the same path?

d. (0.5 pts) Compare the application of Q-learning in this chess scenario with that of the grid of exercise 1. How do the two scenarios differ? How does that translate into your results?

   En el primer escenario habia 4 posibles movimientos, mientras que en el 2o hay muchos más, hay que tener en cuenta que el rey se puede mover en diagonal y que la torre, ademas de arriba, abajo, izquierda y derecha, puede moverse varias casillas (por ejemplo arriba 5 casillas). MAS ESTADOS

e. (1 pts) Compare the use of Q-learning with that of search algorithms of P1 for the chess scenario seen here, both in the deterministic and stochastic case.

   Hay algunos parecidos en el uso de Q-learning y los algoritmos de búsqueda, ambos comprueban locerca que estan del resultado, para decidir la acción a realizar (los algoritmos de búsqueda con una heurística, y el Q-learning utilizando las recompensas. Dicho esto, podemos observar que los algoritmos de búsqueda requieren una gran potencia computacional, especialmente en escenarios deterministas con búsquedas profundas. Por otro lado el Q-learning puede ser computacionalmente exigente durante el entrenamiento, pero es más eficiente durante la toma de decisiones. Los algoritmos de búsqueda funcionan bien con reglas deterministas, mientras el Q-learning puede lidiar con incertidumbres es más versátil y puede manejar entornos estocásticos, donde las acciones pueden tener resultados probabilísticos.

Conclusiones

En esta práctica sobre el Q-Learning, a través de los ejercicios, hemos explorado desde una implementación básica en un entorno de cuadrícula hasta su aplicación en el dominio del ajedrez. Al implementar Q-learning en una cuadrícula, hemos aprendido a configurar el problema, definir recompensas y obstáculos, y ajustar los parametros clave como la tasa de aprendizaje y el factor de descuento. Hemos experimentado con la convergencia del algoritmo, comprendiendo cómo evaluar la estabilidad de los valores Q y ajustar los parametros para obtener un rendimiento óptimo.Este ejercicio nos ha permitido visualizar el proceso de aprendizaje y evaluar la efectividad de las políticas aprendidas.
Desde representar estados de juego hasta definir acciones y recompensas, hemos visto cómo el Q-learning podría utilizarse en problemas más complejos y estratégicos cómo el ajedrez.