Lista 2 - MAC0425 - Inteligência Artificial

Lucas Paiolla Forastiere - 11221911

12 de julho de 2021

1. Os experimentos foram feitos utilizando os arquivos `qlearn.py` e `explore-qlearn.py` (basta ter `python` e a biblioteca `numpy` instalados e rodar utilizando `$ python qlearn.py` ou `$ python explore-qlearn.py`).

O programa que se chama apenas `qlearn.py` obtém os q-valores sem fazer nenhuma exploração. Apenas aplicamos a fórmula do algoritmo em cada entrada da matriz 3×4 várias vezes até convergir. Após 5 iterações, temos os seguintes q-valores:

```
[[[ 96.   96.   97.   95.]
  [ 96.   97.   98.   -3.]
  [ 97.   98.   99.   97.]
  [100.  100.  100.  100.]]

 [[ 95.   96.   -3.   94.]
  [ -4.   -2.   -2.   -4.]
  [ -3.   98.   98.   96.]
  [ 97.   99.   98.   97.]]

 [[ 94.   95.   95.   94.]
  [ 94.   -3.   96.   95.]
  [ 95.   97.   97.   96.]
  [ 96.   98.   97.   97.]]]
```

Figura 1: O `ndarrary` dos *q-values* para o exercício 1. Como cada estado possui 4 ações, então temos um tensor de três dimensões. Cada bloco de 4 arrays se refere a uma linha do mapa. Observe como a quarta linha são apenas valores 100. Isso é devido ao fato de esse ser o nosso estado final que tem recompensa 100. Cada um dos valores de uma linha representa o q-valor de ir naquela direção. Na ordem temos: esquerda, cima, direita e para baixo. Portanto, se quisermos saber o q-valor de ir para a esquerda na posição $2, 3$ da matriz, basta olhar o segundo conjunto de 4 linhas, dai a terceira linha desse conjunto e olhar o primeiro valor dessa linha. Vemos que é -3, o que faz sentido, pois a casa da esquerda leva a uma recompensa negativa.

Para facilitar a visualização, eu também coloquei para mostrar um mapa esquemático de como está a política referente aos q-valores exibidos.

O programa `explore-qlearn.py`, por sua vez, considera que temos um agente explorando esse mapa e que realiza um episódio por vez, começando no estado final e terminando

```
[['>' '>' '>' '<']
 ['^' '^' '^' '^']
 ['^' '>' '^' '^']]
```

Figura 2: A política referente à figura anterior.

quando chega ao estado final. Para fazer essa exploração, nós adotamos a estratégia chamada de *epsilon-greedy*, que sempre segue a atual política ótima a menos de uma chance ε de fazer uma ação completamente randômica (adotei $\varepsilon = 0.3$).

Além disso, foi levado em conta que o agente, após escolher uma ação, tem apenas 80% de chances de realmente conseguir fazer aquela ação e 10% de acabar indo para um dos lados adjacentes, como proposto em aula pelo professor. Quando o agente acaba indo para um lugar que ele não queria ir, ele não sabe disso. A única informação que ele recebe é a recompensa pela ação que ele pensa que tomou e, portanto, o q-valor que vamos atualizar é aquele referente a ação que o agente *acha* que tomou, não de fato a que ele acabou fazendo sem querer (indo parar em um lugar que não pretendia).

É interessante que, mesmo com essa mecânica, os q-valores convergem exatamente para os mesmos que anteriormente na figura 1, assim como a política ótima, evidentemente.

O número de episódios que precisamos para chegar nesse resultado, entretanto, é relativamente alto, cerca de 45 (como o comportamento do agente é não-determinístico, esse valor varia).

2. Para o segundo exercício, temos que o estado $3, 4$ tem uma recompensa de $+10$. Isso faz com que nossos dois algoritmos anteriores não convirgam para valor nenhum. O primeiro algoritmo (`qlearn.py`, que não explora) vai aumentando os q-valores a cada iteração, ficando todos eventualmente infinitos. Já no que exploramos, isso acontece também, mas as ações que levam ao estado de recompensa negativa ainda possuem q-valores negativos.

Na prática, quando rodamos o algoritmo em que há um agente explorando, cada epsódio dele vai ficando mais e mais demorado, pois ele tenta passar o máximo de vezes que ele consegue pelo estado de recompensa 10 antes de terminar o epsódio.

A pergunta feita na lista é se isso corresponde ao esperado. Minha resposta é que sim, pois esse é de fato o comportamento que *se espera* caso nós não colocamos nenhuma punição no agente por ficar andando muito tempo antes de terminar um epsódio. Todavia, esse não é o comportamento *desejado*. Desejamos uma trajetória em que o agente passa pelo 10 uma única vez e vai direto para o estado final.

Para isso, precisamos modificar o valor de γ, que é o coeficiente de penalização do agente por levar muitos passos para finalizar um epsódio. Nos casos anteriores, γ era igual a 1.

Tomando um valor de γ muito pequeno, como 0.5, temos que o agente quer terminar o epsódio tão rápido, que ele nem passa pelo 10, levando aos q-valores da figura 3 e política da figura 4.

Por outro lado, se colocamos um valor não tão grande de penalidade, como $\gamma = 0.9$, temos os resultados das figuras 5 e 6.

Portanto, vimos que para fazer o agente ter o resultado que gostaríamos, basta modificar o valor de γ de forma a punir mais ou menos ele por ficar andando demais.

Para convergir, tomamos muito mais tempo agora, cerca de 80 episódios.

```
[[[   4.375      4.375     10.75       1.1875 ]
  [   4.375     10.75      23.5      -45.125  ]
  [  10.75      23.5       49.        10.75   ]
  [ 100.       100.       100.       100.     ]]

 [[   1.1875     4.375    -45.125     -0.40625]
  [ -97.8125  -100.5      -88.25     -97.8125 ]
  [ -45.125     23.5        4.375      4.375  ]
  [  10.75       0.        4.375      6.6875 ]]

 [[  -0.40625    1.1875     1.1875    -0.40625]
  [  -0.40625  -51.         4.375      1.1875 ]
  [   1.1875    10.75       6.6875     4.375  ]
  [  15.375     15.375      0.         0.     ]]]
```

Figura 3: Q-valores para o exercício 2 com $\gamma = 0.5$.

```
[['>' '>' '>' '<']
 ['^' '>' '^' '<']
 ['^' '>' '^' '<']]
```

Figura 4: Política para o exercício 2 com $\gamma = 0.5$. Vemos que o agente sequer passa pela casa $3, 4$, que é onde temos uma recompensa de $+10$, pois o desconto por andar mais é tão grande que não vale a pena.

```
[[[ 62.171       -2.71        70.19        -3.439      ]
  [ 62.171       70.19        79.1        -26.929      ]
  [ 70.19        79.1         89.          70.19       ]
  [100.         100.         100.         100.         ]]

 [[ -3.439       62.171      -26.929       62.171      ]
  [-44.0461     -28.81      -28.81       -28.81000043]
  [-26.929       79.1         79.1         62.171      ]
  [ 70.19        89.          79.1         89.         ]]

 [[ 62.171       54.9539      70.19        62.171      ]
  [ 62.171      -26.929      79.1         70.19       ]
  [ 70.19        70.19        89.          79.1        ]
  [ 90.1         90.1        100.         100.         ]]]
```

Figura 5: Q-valores para o exercício 2 com $\gamma = 0.9$.

```
[['>' '>' '>' '<']
 ['^' '^' '^' '^']
 ['>' '>' '>' '>']]
```

Figura 6: Política para o exercício 2 com $\gamma = 0.9$. Vemos que agora o agente escolhe ir para a direita sempre que possível na última linha, pegando a recompensa de +10 e se está na posição $3, 4$, ele também escolhe ir para a direita, permanecendo no lugar e ganhando uma recomepensa de +10 novamente. E caso ele acabe indo para cima sem querer, então ele termina seu epsódio.