Trabalho Prático 2: Aprendizado por reforço

Felipe Louzada Mingote[1]

[1]Departamento de Ciência da Computação - Universidade Federal de Minas Gerais

1. Introdução

Os objetivos deste trabalho consistem na prática de conceitos e algoritmos de aprendizado por reforço, em especial, o algoritmo Q-learning. Além disso, proporcionar um entendimento de como funções de recompensa e o ambiente afetam a política gerada pelo agente.[Chaimowicz and Negrisoli 2022]

2. Solução do Problema

Dado a docuemntação do trabalho, o método Q-learning foi implementado com as seguintes variações:

- Recompensas Negativas
- Recompensas Positivas
- Recompensas Negativas Estocásticas

As estruturas utilizadas para modelar o problema foram 3 matrizes, sendo elas ”mapa”, ”Q”e ”R”. A matriz mapa contém as informações arquivo .map passado na entrada já modelado, ou seja, o mapa já construído em memória. Já a matriz Q representa os Q-valores do algoritmo Q-learning (explicados mais a frente), divididos por ação em cada estado, ou seja, para cada par de coordenadas (x,y) teremos quatro possíveis direções para a movimentação do nosso agente, sendo elas ’cima’, ’baixo’, ’esaquerda’ e ’direita’, e cada direção tem seu próprio Q-valor. Por sua vez, a matriz R carrega o valor da recompensa que o agente vai receber naquele estado. Todas as variações do algoritmo possuem como base essa modelagem inicial. Por questões de implementação e facilidade de acesso na matriz, o programa lê as coordenadas iniciais do nosso agente invertidas, ou seja, o valor X na entrada do programa corresponde a coluna Y em uma matriz e, de maneira análoga, a entrada Y representa a linha X em uma matriz com início (0, 0).

2.1. Q-Learning

O algoritmo Q-Learning é uma técnica de aprendizado por reforço onde, para cada estado, toda possível ação é mapeada e, a partir de cada ação dado um estado, calculamos o Q-valor daquela ação dentro daquele estado, utilizando a seguinte equação:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma max(Q(s',a')) - Q(s,a))$$

- $Q(s,a)$: O valor atual do Q-value no estado atual com a ação desejada
- r: o valor da recompensa que será obtida a partir da ação a
- α: a taxa de aprendizado, nesse trabalho, definida em 0.1
- γ: a taxa de desconto nesse trabalho, definida em 0.9
- $max(Q(s',a'))$: Q-value da ação que maximiza o ganho do estado futuro

A fim de facilitar a implementação, os Q-valores dos estados finais, definidos pelo objetivo e pelo fogo, são a própria recompensa de cada um, para que o agente entenda que eles são, no final das contas, os valores mais relevantes para calcular os próximos Q-valores. Assim que o agente atingir um dos estados objetivos, ele reinicia sua exploração a partir do estado inicial, e isso se dá até o agente completar a quantidade de passos passada na entrada do programa. Outra implementação exigida na especificação do trabalho foi um fator de exploração, ϵ com um valor igual a 0.1. Esse fator de exploração corresponde a probabilidade do agente, ao invés de escolher a ação que maximiza o seu Q-valor, escolhe uma ação aleatória. Esse fator é muito importante para que o agente explore possíveis rotas, ditas, escondidas, e sair de máximos locais.

2.1.1. Recompensas Negativas

A primeira variação do método Q-learning implementado foi o de recompensas negativas, o que faz com que, a cada ação, o agente receba uma penalidade na sua recompensa, ou seja, ele tende a buscar o caminho com a menor penalidade possível. Como a única recompensa positiva do mapa é o objetivo final, o agente é muito motivado a buscar esse objetivo, uma vez que qualquer outra busca nesse mapa levará a uma penalidade. As recompensas implementadas nessa variação, assim como nas outras variações, estão descritas na especificação inicial do trabalho.

2.1.2. Recompensas Positivas

Mais difícil de modelar, a variação com recompensas positivas para o nosso agente faz com que ele ganhe pontos só por andar, e é bem possível que ele possa caia em um *loop*, ou mesmo fique batendo contra a parede para tentar maximizar a quantidade de pontos ao invés de explorar o mapa em busca do objetivo final. Para tentar contornar essa situação, tentamos aumentar o ϵ (nossa taxa de exploração), pois assim, com um caráter exploratório maior, muitos desses loops eram evitados, mas ainda sim visíveis quando comparávamos seus Q-values.

2.1.3. Recompensas Negativas Estocásticas

Utilizando a mesma recompensa negativa já implementada, a variação estocástica utiliza de um fator de aleatoriedade para o nosso agente se movimentar, ou seja, quando ele escolhe uma ação, existe uma probabilidade de 0.2 de chance do agente escolher as direções perpendiculares à deireção desejada, sendo 0.1 para a esquerda e 0.1 para a direita. Essa variação aumenta muito o fator de explorabilidade do nosso agente, uma vez que ele pode escolher caminhos não visitados e, assim, poder conhecer melhor o mapa mais rápido. Outra observação que foi observada em alguns testes é que, com esse fator estocástico, o agente tende a escolher caminhos mais seguros e longe do fogo, uma vez que ele pode se movimentar para aquela posição quando ele queria passar do lado para chegar no objetivo final.

3. Análise Experimental

Utilizando os mapas e os gabaritos dos mesmos disponibilizados, comparamos as políticas resultantes quando utilizamos o método padrão com recompensas negativas e com o método estocástico, contando 300.000 passos para cada versão do algoritmo e para cada mapa. Um adendo, a variação com recompensas positivas não foi integrada na comparação, uma vez que ela não convergia no objetivo sem a alteração dos parâmetros α, γ e ϵ.

A seguir temos os mapas, a posição inicial de cada mapa, e a variação utilizada:

```
mapa_teste.map, x=0, y=3
standart
>>>>O
v@@^x
v@@^<
>>>^<

stochastic
>>>>O
v@@^x
v@@^<
>>>^^
```

```
choices.map, x=5, y=0
standart
@v<>>>>v<<@
@>@vx@xv@^@
@v@v@@xv@^@
@v@vx@xv@<@
@v>v@@xv<<@
@<@vx@xv@^@
@v@v@@xv@^@
@^>>>O<<<<@

stochastic
@><>>>>v<<@
@<@^x@xv@^@
@v@v@@xv@^@
@^@vx@xv@v@
@^>v@@xv<<@
@^@vx@xv@^@
@^@v@@xv@v@
@>>>>O<<<<@
```

```
maze.map, x =10, y=0
standart
x@x@x@x@x@v@
```

```
v<<<<<<<<<<@
vx@@@@@@@@^@
vx@<>>^v^v^@
vx@>@@@@@@@@
vx@vv<<^<^<x
vx@@@@@@@@<@
O<<<<<<<<<^@
```

stochastic

```
x@x@x@x@x@v@
v<<<<<<<<<<@
vx@@@@@@@@^@
vx@^>^v>v>^@
vx@>@@@@@@@@
vx@<<<<v<<vx
vx@@@@@@@@^@
O<<<<<<<v>>@
```

4. Conclusão

A modelagem de um problema sempre foi, e continua sendo, um dos maiores desafios que os algoritmos de IA enfrentam, e com o aprendizado por reforço do Q-learning não foi diferente. É visível como a implementação de recompensas positivas faz com que o algoritmo demore muito a encontrar (quando encontra) a solução. Um outro ponto observado nesse trabalho foi com que, com a implementação estocástica, muitas vezes o algoritmo encontrava a primeira solução muito mais rápida, porém, ele acaba fazendo o nosso agente cair no fogo, o que acaba penalizando muito um caminho que poderia ser muito perto do ótimo.

O aprendizado por reforço é, sem dúvidas, uma ferramenta muito poderosa, mas também é muito sensível a sua definição e modelagem do mundo, principalmente nas recompensas, uma vez que elas podem acabar viciando nosso agente em loops e movimentações desnecessárias, por isso, o maior desafio é achar essa modelagem órima da recompensa para cada contexto.

Referências

Chaimowicz, L. and Negrisoli, T. (2022). Trabalho pratico ii: Aprendizado por reforço. *Belo Horizonte, Brasil.*