

Trabalho Prático II - 2024/2

Igor Lacerda Faria da Silva

1. Introdução

O Trabalho Prático II de Introdução à Inteligência Artificial consistiu na implementação (em Python) de 3 variações do algoritmo *Q-Learning* para *Path-Finding*, em mapas de um jogo simples (o mesmo jogo do TP I). As variações foram: o *Q-Learning* padrão (standard); com matriz de recompensa positiva (positive) e com movimento estocástico (stochastic). O repositório deste trabalho pode ser encontrado neste [link](#).

A documentação está dividida da seguinte maneira: esta breve introdução descreve o que foi realizado, a Seção 2 descreve a modelagem do programa, entrando nos detalhes das estruturas de dados; a Seção 3, por sua vez, foca no *Q-Learning* e nas variações implementadas. Por fim, a Seção 4 é uma análise comparativa entre as diferentes variações.

2. Estruturas de Dados e Modelagem

Similarmente ao TP I, a estrutura de dados mais prevalente foi a matriz. As matrizes foram usadas em diversos contextos: para armazenar o mapa (caracteres), as recompensas das posições e, claro, gerenciar a matriz (tensor) Q . A matriz Q é inicializada com valores aleatórios entre -1 e 1 (somente posições válidas são preenchidas). Além disso, outra estrutura auxiliar utilizada ao longo do programa foi uma tupla, para representar as coordenadas no mapa.

Também foi criado um tipo (enum) *Action* para auxiliar no mapeamento das direções em que o agente pode se mover. Ele é particularmente útil no que diz respeito à escolha de uma direção perpendicular, na variação estocástica do algoritmo. Assim como no TP I, foi tomado certo cuidado para lidar com o sistema de coordenadas. Após a leitura do mapa, os dados são transpostos e, no final, após o processamento do algoritmo, os dados são novamente transpostos.

No mais, o restante da implementação é bem similar ao pseudocódigo visto nas aulas. Começando do estado inicial, um *loop* escolhe uma ação. Geralmente esta é a melhor ação (para aquele estado), mas devido ao mecanismo de exploração ϵ -greedy, há a chance (10%) de ser alguma outra. A próxima posição é calculada, considerando que é possível não sair do lugar ao tentar realizar um movimento para uma posição inválida.

Caso o agente tenha atingido o estado objetivo (0) ou um estado com fogo (x), a simulação é reiniciada e a matriz Q é atualizada com base na recompensa do estado atingido. Caso contrário, é buscada a recompensa mais promissora do estado atingido, e é ela que é utilizada para atualizar a recompensa da transição realizada (considerando, claro, $\alpha = 0.1$ e $\gamma = 0.9$). A variação *positive* consiste apenas na troca da matriz de recompensas, enquanto a variação *stochastic* traz nuances que são detalhadas na Seção 3.

Em resumo, o estado contém todas as posições atingíveis; as ações são os 4 movimentos permitidos; o agente sempre se encontra em uma dada posição e a recompensa é calculada pela estimativa inicial (aleatória) e as sucessivas iterações, conforme descrito nos parágrafos anteriores.

3. *Q-Learning*

Uma breve descrição do *Q-Learning* implementado foi dada na seção anterior. De forma mais geral, o *Q-Learning* é um método simples de Aprendizado por Reforço, que é a área do Aprendizado de Máquina que se destaca pelo aprendizado a partir da interação com o ambiente. É feito o condicionamento do agente por meio de recompensas ou punições, sem fazer uso de dados rotulados (por exemplo). No *Q-Learning*, o agente explora o ambiente selecionando ações (que pode levar a novos estados), e incorporando recompensas, atualizando o valor de sua matriz de valor esperado $Q^\pi(s, a)$.

Como comentado na Seção 2, para realizar a troca para a variação *positive*, somente é selecionada a outra descrição das recompensas. Já a para a *stochastic*, o primeiro passo é escolher um número aleatório entre 0 e 1. Caso ele caia no intervalo (0,0.1], é realizada uma mudança na trajetória, para a perpendicular na esquerda. Caso ele caia no intervalo (0.1,0.2], é realizada uma mudança na trajetória, para a perpendicular na direita. Em outras situações a trajetória não é alterada. Para o agente, é atualizada a matriz Q com base na trajetória em que ele *acredita* estar seguindo, e não na trajetória “real”.

4. Análise Comparativa

Foi realizada uma análise comparativa nos 3 mapas distribuídos junto com a especificação do trabalho: o `mapa_teste`, o `choices` e o `maze` (apesar de o `mapa_teste` estar presente mais por propósitos de depuração). Para o `mapa_teste`, foi usado o exemplo da especificação: começando na posição (0,3), foram dados 100000 passos para a variação *standard*, obtendo-se a seguinte política:

```
v>>>0
v@@^x
v@@^<
>>>^^
```

Ela é idêntica à saída presente na especificação e é muito boa, pois consistentemente encaminha o agente para o estado objetivo. Para a variação *positive*, a posição inicial foi a mesma, mas foram considerados apenas 1000 passos. Isso se faz necessário porque frequentemente o agente fica “preso” em sequências que não terminam (ou seja, não levam nem ao estado objetivo e nem a um estado com fogo), o que “trava” o contador de passos. Assim, foi obtida a seguinte política:

```
v<>v0
<@@<x
<@@^>
<<<^<
```

Ela é consistentemente ruim, pois evita fortemente com que o agente atinga o objetivo. No entanto, isso é completamente esperado, pois o agente não possui “incentivo” para buscar o objetivo, uma vez que pode apenas alternar entre os estados que já produzem recompensa positiva. Para fechar, também foi mantida a posição inicial (0,3) na variação *stochastic*. Ela também foi executada por 100000 passos e a política obtida foi praticamente igual à do *standard*:

```
v>>>0  
v@@^x  
v@@^v  
>>>^<
```

Há apenas algumas pequenas diferenças na região próxima ao fogo. Elas poderiam estar associadas a uma questão da seleção dos números aleatórios, dado que o mapa é muito pequeno e o número de passos é relativamente grande. Mas há outra explicação plausível para a pequena diferença: o agente prefere descer na posição logo abaixo do fogo para evitar que, por engano, ao tentar ir para um dos lados, ele caia no fogo.

4.1. Maze

4.2. Choices