Introdução à Inteligência Artificial

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 divida 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 $\boldsymbol{Q^{\pi}(s, a)}$.

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 100 mil 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 `posisitve`, 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 100 mil 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, caia no fogo se tentasse ir para um dos lados.

4.1. `Maze`

O `maze` é um mapa muito mais complexo do que o mapa de testes. Sua estrutura é semelhante a um pequeno labirinto, contando com muitas paredes e regiões com fogo. O caso de estudo foi inspirado no exemplo distribuído junto com os mapas: a posição inicial é (10, 0), mas são dados 1 milhão de passos para o `standard` (ao invés de apenas 300 mil, como no exemplo original). O intuito deste número alto foi lidar com regiões em que a política indicava ou uma colisão com as paredes, ou uma colisão com a borda do campo. Foi obtida a seguinte política:

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

Ela é bastante consistente, apesar de ainda ter duas regiões, no meio do labirinto, que indicam colisões com as paredes. Para a variação `positive`, foi usada a mesma posição inicial, mas devido à restrição de tempo comentada anteriormente, foram dados apenas 2000 passos. A política obtida foi:

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

Primeiramente, a discrepância mais clara dessa execução foi a extrema variância no tempo de execução. Algumas execuções foram muito demoradas, outras nem chegaram a terminar em tempo hábil. De fato, é possível que este exemplo de política não seja realmente "representativo" das execuções médias com 2000 passos, para a variação `positive` no mapa `maze`. De qualquer modo, a política obtida não é muito boa, chegando a incluir até alguns "pulos no fogo". Isso provavelmente está associado à baixa quantidade de passos (que, novamente, não pode ser muito mais elevada devido à variância e restrições de tempo).

Mesmo assim, alguns padrões emergiram: a segunda linha apresenta um comportamento muito "natural", semelhante à execução `standard` (provavelmente um mecanismo para fugir da alta concentração de fogos). Apesar disso, não é possível deixar de notar a quantidade escancarada de posições de colisão, seja com as paredes ou com as bordas do mapa, que também podem estar associadas à baixa quantidade de passos, bem como à tendência do `positive` de "não querer" atingir o objetivo, preferindo gerar ciclos em regiões que sejam minimamente "seguras" (sem fogos). A última observação que vale destacar nesse caso, é o fato de o agente realmente tentar "se esforçar" para não atingir o objetivo, visto que nenhuma das regiões próximas do objetivo está apontada para ele (como é de se esperar em um `positive`).

Para o estocástico, na mesma configuração do `standard`, foi obtida a seguinte política:

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

Ela é relativamente parecida com a do `standard`, no entanto, é possível observar alguns comportamentos peculiares: o agente passa a ter um grande medo de fogo, preferindo constantemente esbarrar nas paredes. Isso é consistente, dado a incerteza do movimento: pode ser mais seguro apenas se mover por engano, do que tentar realizar um movimento mas cair no fogo (como observado inicialmente para o mapa de teste). Além disso, é notável que mesmo em situações em que não há fogo por perto, o agente acaba "esbarrando" em posições proibidas com mais frequência, o que também é atribuído à incerteza.

4.2. `Choices`

O `choices` é também um mapa muito mais complexo do que o mapa de testes. Diferentemente do `maze`, a dificuldade introduzida é, em partes, por existirem muitas possibilidades de caminhos a serem seguidos e o mapa não ser muito simétrico. Similarmente ao `maze`, o caso de estudo foi inspirado no exemplo distribuído junto com o mapa: manteve-se a posição inicial (5, 0), mas foram executados 1 milhão de passos, também para reduzir o número de colisões. Para o `standard`, obteve-se:

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

A dificuldade do mapa fica evidente devido ao aumento no número de regiões de colisão, chegando a um total de 4 para esta política. Ela contém uma "volta" ao redor da parede central, que pode ter impactado no aprendizado (de fato, as regiões problemáticas estão todas à direita da "volta"). Esta política não é ideal, mas ainda é fortemente consistente.

Mais uma vez, a execução para a variação `positive` foi um tanto reduzida, considerando apenas 3 mil passos.

```
@<<<<<<<<<@
@^@vx@xv@^@
@^@<@@x^@^@
@^@<x@x^@v@
@^><@@xvv>@
@^@^x@x>@v@
@>@^@@xv@v@
@<<^^0>^><@
```

Tal qual no `maze`, o objetivo é evitado, dado que há preferência para se manter uma região "segura", acumulando uma recompensa maior. Há muitas regiões de colisão, mas que no geral podem estar associadas a essa ideia de acumulação de recompensa. Também de forma semelhante ao `maze`, alguns padrões emergiram, como na segunda coluna e na primeira linha, com a política indicando a mesma ação para a maioria das posições.

Para a variação `stochastic`, também foi usada a mesma configuração do que a padrão. Ao final, obteve-se:

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

Mais uma vez, o padrão de evitar fortemente o fogo foi observado: o agente prefere uma constante sequência de colisões com as paredes (ver coluna 8 da política). De maneira geral, as colisões, mesmo não próximas ao fogo, também são mais frequentes, devido à incerteza do movimento. Para esta política, a parte de baixo do mapa ficou super consistente, apresentando "defeitos" apenas nas primeiras linhas.