Universidade Federal de Minas Gerais
Departamento de Ciência da Computação

Trabalho Prático 2: Aprendizado por Reforço

Ingrid Rosselis Sant Ana da Cunha
irscunhadcc.ufmg.br

1 de Julho de 2019

1 Introdução

O segundo trabalho da disciplina Inteligência Artificial tem por objetivo aplicar os conceitos de aprendizado por reforço, vistos em sala, em uma versão simplificada do jogo *Pac-Man*. Nesta versão, o mapa do jogo é bidimensional e estático, isto é, os fantasmas e as recompensas possuem local fixo no mapa.

A ideia do problema é implementar um algoritmo *Q-Learning* para este mapa que seja parametrizável, e treinar o algoritmo para encontrar as pastilhas no mapa e evitar os fantasmas de forma ótima.

O aprendizado por reforço é um dos três tipos de aprendizados mais usados em Inteligência Artificial, onde o agente aprende interagindo com o ambiente por meio de ações e avaliando as recompensas resultantes dessas ações em determinados estados. O objetivo do aprendizado por reforço é encontrar uma estratégia ótima que maximize as recompensas recebidas.

O *Q-Learning*, algoritmo em estudo neste trabalho prático, é uma variação do *Temporal-Difference Learning*, um tipo de aprendizado por reforço tabular que não necessita de modelo e utiliza apenas os valores recebidos por ações tomadas. *Q-Learning* é um algoritmo *off-policy*, ou seja, que pode utilizar políticas de aprendizado e exploração diferentes, e que interage com o ambiente fazendo uma ação e analisando sua recompensa e qual o novo estado que ela leva. O algoritmo sempre leva em conta a melhor ação para o próximo estado e por isso sempre retorna uma política ótima para exploração.

2 Modelagem do Problema

A ideia por trás da modelagem feita para resolver o problema foi simplificar ao máximo as estruturas de dados dados usadas. O projeto é composto por métodos e definições globais das variáveis necessárias para executar o algoritmo. Entre os métodos, podemos citar a existência de um método que roda um episódio inteiro do *Q-Learning*, isto é, ele define a posição inicial do robô no mapa de maneira aleatória, desde que esta seja válida (não seja uma parede ou um estado terminal), e aplica o algoritmo até que um estado terminal tenha sido encontrado. A execução para vários episódios é feita no fluxo principal do programa. As fontes seguidas para implementar o algoritmo foram os slides da disciplina e o artigo *Simple Reinforcement Learning: Q-learning*[1].

As outras funções utilizadas são em relação à criação de mapas e tabelas e da escrita em arquivo. É interessante comentar que foram usadas dois tipos de tabelas neste trabalho que possuem forte interação: o mapa recebido e a *Q-Table*. O mapa recebido foi armazenado em forma de uma matriz Numpy e acessado para receber informação sobre os próximos estados à partir da posição atual.

Já a *Q-Table* foi estruturada em forma de uma matriz Numpy tridimensional que possui em seus indexes as linhas e colunas relativas ao mapa original e as ações possíveis. O valor armazenado nesta matriz é o resultado da Função de Valor de cada ação para o estado que se encontra na posição do mapa indicada pela linha e coluna. O motivo por trás do uso de uma matriz tridimensional foi manter a estrutura aprendida em sala, mas facilitando a atualização da tabela.

Resumindo a obtenção de um estado e um valor na *Q-Table*, temos:

$$estado = mapa\_original[linha, coluna]$$
$$Q(estado, ação) = q\_table[linha, coluna, ação]$$

Em relação às estruturas de dados usadas, algumas informações foram propositalmente repetidas para facilitar a utilização no algoritmo: os estados e ações podem facilmente ser retornados na forma de indexes ou das STRINGS que os definem e; os estados foram divididos entre todos possíveis e os terminais. Outra estrutura interessante é o dicionário MOVEMENT, que retorna duas funções *lambda* de movimentação para cada direção dada, facilitando o cálculo de próximas posições sem a necessidade de estruturas condicionais soltas pelo código ou de várias funções auxiliares.

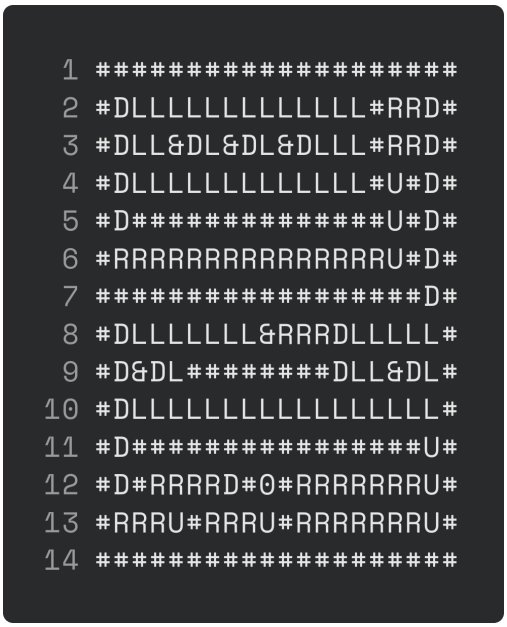
3 Resultados Obtidos

Abaixo seguem os resultados das políticas obtidas pelo algoritmo para os três arquivos de testes disponibilizados. Este resultado é o mesmo impresso no arquivos PI.TXT, como especificado.

3.1 Política: Arquivo 1 – 7x14

```
1 ##############
2 #RRRRR0LLLLLL#
3 #######U##U#U#
4 #RRRRRRU&RU#U#
5 #U##########&#
6 #ULLLLLLLLLL#
7 ##############
```

3.2 Política: Arquivo 2 – 14x20



3.3 Política: Arquivo 2 – 11x20

4 Análise Experimental

A análise feita para este trabalho foi a relação entre a média de valores Q(estado, ação) ao longo do tempo, este medido por episódios. Para cada um dos três arquivos de teste passados, foram variados a taxa de aprendizado e o fator de exploração pelos valores 0.1, 0.5 e 0.9. O número de episódios foi fixado em 30000, um número grande o suficiente para que o algoritmo consiga atingir a política ótima em todos os 3 casos de teste.

Esta última observação é constada analisando os 3 gráficos abaixo: em quase todas as variações feitas, o algoritmo conseguiu convergir para o mesmo valor de recompensa média. A única exceção é para o terceiro mapa com os valores de taxa de aprendizado e de fator de exploração em 0.1, que ficaram muito próximos de convergir.

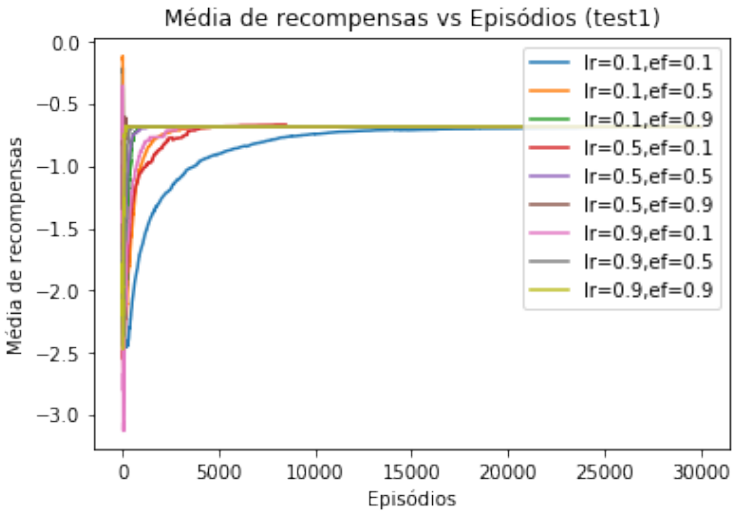


Figura 1: Gráfico para o mapa 1 – 7x14

O gráfico 1 deixa bem claro a importância do *tuning* do fator de exploração: 3 das 4 curvas que demoram mais a convergir são as que possuem o fator de exploração igual a 1. Isso significa que o algoritmo demora muito mais a encontrar a política ótima e pode acabar ficando "preso"a ótimos locais (muito *exploit*).

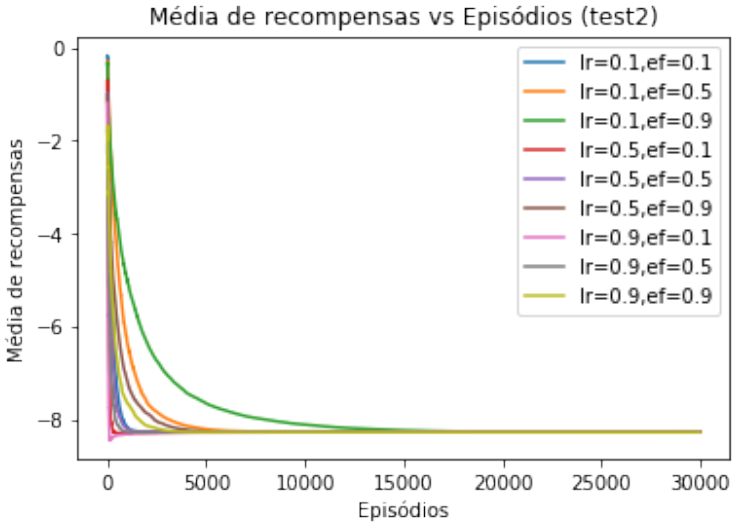


Figura 2: Gráfico para o mapa 2 – 14x20

O gráfico 2 referente mapa 2 foi o que obteve piores médias por episódios. Isso pode ser explicado por uma característica já conhecida no algoritmo *Q-Learning* e notada no mapa: a recompensa média depende muito do tipo de mapa. Neste caso, o mapa possui linhas quase completamente compostas por paredes, o que faz com que o robô precise se deslocar mais e sempre só tenha uma opção de caminho por esta área.

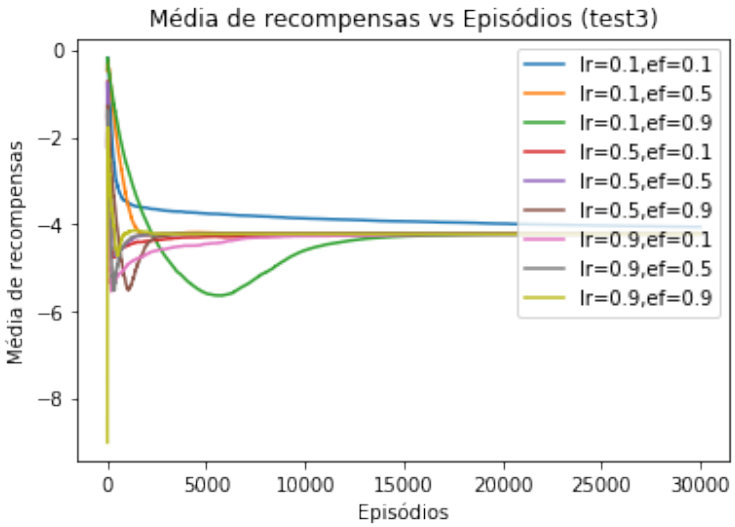


Figura 3: Gráfico para o mapa 3 – 14x20

Já o gráfico 3 do mapa 3 possui uma característica interessante que pode explicar o porquê da variação nas médias: ele é composto por muito mais fantasmas que os outros. Logo, faz sentido que haja mais variação, porque existem mais pontos onde o algoritmo perde e recebe uma recompensa negativa.

5 Conclusão

O trabalho prático 2 da disciplina Inteligência Artificial teve como objetivo implementar o algoritmo *Q-Learning* para uma variação do jogo *Pac-Man*. Esta variação contém um mapa estático, onde nem fantasmas nem pastilhas podem se mover ou trocar de posições. No entanto, em um mesmo mapa podem haver inúmeros fantasmas e pastilhas.

O algoritmo implementado é bem simples, e por isso não houveram grandes dificuldades na parte de codificação do trabalho. Neste trabalho, pode ser constatado também que o algoritmo é, de fato, ótimo, pois os arquivos de saída contendo as saídas foram comparados com os de outros colegas de turma, obtendo-se os mesmos resultados.

Por fim, este trabalho permitiu aplicar os conceitos vistos em sala de aula sobre aprendizado por reforço. Foi interessante notar que as características sobre *Q-Learning* e métodos tabulares puderam ser vistas na prática, como a demora na convergência do algoritmo, a diferença nas políticas de aprendizado e exploração e o sempre retorno da política ótima.

Referências

[1] Andre Violante. Simple Reinforcement Learning: Q-learning. `https://towardsdatascience.com/simple-reinforcement-learning-q-learning-fcddc4b6fe56`.