OpenAI - Cartpole

Universidade Federal da Fronteira Sul
Ciência da Computação

Fernanda Bonetti*, Matheus Henrique Trichez†

Abril 2018

1 Resumo

O presente trabalho[1] é uma tentativa de resolução do problema proposto por [Barto et al., 1983] através da implementação do algoritmo de aprendizagem supervisionada ativa, *Q learning*. Estão presentes neste, uma breve descrição do algoritmo implementado e da abordagem de construção dos estados para resolução do mesmo. Além de uma comparação de desempenho da implementação feita, em relação ao desempenho de outras duas abordagens mais simplistas.

*fernandasbonetti@gmail.com
†mh.trichez@gmail.com

[1] Submetido como conteúdo avaliativo no Componente Curricular *Inteligência Artificial*

2 Execução

No terminal será possível observar os resultados em tempo de execução. Para executar o trabalho, use o seguinte comando:

```
# python cartpole.py
```

3 Metodologia & Desenvolvimento

A primeira etapa executada foi a discretização dos parâmetros da *observation* utilizados pelo OpenAI Gym, afim de criar os estados utilizados para popular a tabela Q.

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

Figura 1: Parâmentros do OpenAI Gym - Cartpole-V0 environment

A discretização dos quatro parâmetros foi feita como o exemplificado na Figura 2, gerando um total de 62500 estados.

```
[-2 -1  0  1  2]
[-2 -1  0  1  2]
[-2.  -1.9 -1.8 ...,  2.7  2.8  2.9]
[-2.  -1.9 -1.8 ...,  2.7  2.8  2.9]
```

Figura 2: Estados discretizados.

Em seguida a tabela Q é inicializada, onde cada combinação dos estados representa uma linha da tabela, e os valores Q são calculados para as ações de direita e esquerda.

Os estados são representados na tabela Q da mesma ordem como são obtidos através das observations, com valores separados para facilitação da busca posterior desses valores.

Q(state) = Car Position | Car Velocity | Pole Angle | Pole Velocity

A combinação dos valores para cada componente do estado, gerou um total de 62500 estados após a discretização.

$$Total = \underbrace{5}_{carPosition} \cdot \underbrace{5}_{velocity} \cdot \underbrace{50}_{angle} \cdot \underbrace{50}_{angveloc}$$

A tabela Q pode ser lida e reutilizada de outras execuções, ou inicializada para a execução atual com valores previamente estipulados, que nesse caso foram utilizados valores iniciais de 1, tanto para ações esquerda, como direita. Ela é salva em um arquivo CSV (Comma-separated Values), contendo o estado discretizado, na forma previamente mostrada, e os valores de ambas as ações (esquerda e direita), como abaixo:

```
'-1.0|-1.0|2.3|-1.4' = (1,1)
```

A fórmula de atualização dos valores Q utilizada é demonstrada abaixo:

$$Q(s_t, a_t) \leftarrow (1-\alpha) \cdot \underbrace{Q(s_t, a_t)}_{oldValue} + \underbrace{\alpha}_{learningRate} \cdot \overbrace{(r_t + \gamma \cdot max_a Q(s_{t+1}, a))}^{LearnedValue}$$

Afim de não favorecer ações recentes sobre ações futuras, a taxa de desconto γ é utilizada como 1. Já a taxa de aprendizado α, leva em conta quanto o modelo deve aprender das novas informações considerando as informações que ele já possui (por exemplo, uma taxa de 1 faria o modelo sempre favorecer as informações novas), é utilizada como 0.05. O valor aprendido é composto pela recompensa (r_t) somada à taxa de desconto da ação com maior valor. A atualização dos valores é feita assim que uma ação é feita e um novo estado (s_{t+1}) é calculado. Para mais detalhes sobre a implementação e sobre o funcionamento do código, veja o arquivo *documentation.py* o qual é a documentação do código segundo as convenções [David Goodger, 2001].

4 Resultados & Conclusões

Uma vez implementado o *Q learning*, foram feitas execuções de 5000 episódios consecutivos partindo de uma tabela *zerada* (*i.e.* todos os estados e ações possuem um valor inicial arbitrário como em [Russell and Norvig, 2016]). Dessa maneira, o agente partiu de uma situação onde nada sabia sobre os estados, e aprendeu a partir da experiência de passar pelos estados encontrados ao longo dos 5000 episódios na tarefa de equilibrar o pêndulo. Como era esperado, após o treinamento o agente alcançou uma média de tempo muito maior do que quando começara o treinamento.

A melhor configuração de parâmetros encontrada para o modelo foram: $\alpha = 0.05$ e $\gamma = 1$. Em comparação à esta, é possível observar na Figura 3, o desempenho de outras execuções do agente com diferentes configurações para os parâmetros α e γ. Quando aumentada a *taxa de aprendizagem* (α) de 0.05 para 0.3 os resultados mostraram uma curva de aprendizado rápida através da média dos primeiros 1000 episódios. Se tornando, porém, muito inconstante e decrescente no decorrer do tempo após esta etapa, o que acabou por baixar as médias de tempo de equilíbrio do pêndulo.

Outro destaque importante feito a partir da Figura 3, é a representação da execução do agente com uma diminuição no parâmetro γ, ou *taxa de desconto*. Alterando o valor deste de 1 para 0.9, a capacidade de aprender do agente tendeu à zero, o que pode ser concluído observando a inclinação praticamente nula da curva gerada pelas médias de tempo de vida do pêndulo. O que por consequência gerou baixíssimas médias de tempo de vida do pêndulo, encontrando-se muito próxima às médias de tempo de vida geradas a partir de escolhas aleatórias para equilibrar o pêndulo. Dado o exposto, scredita-se que conjuntura formada pelo problema e pela abordagem de solução, requer que os estados mais próximos sejam levados em conta sob o mesmo peso em que são levados em conta os estados futuros.

Para fins de comparação foi implementado também um algoritmo ingênuo e sem aprendizado. Este apenas levava em conta o ângulo do pêndulo e reagia de acordo (*i.e.* se ângulo < 0 move-se o carro para esquerda, e caso contrário, move para direita). Como esperado, a curva dada pelas médias do tempo de equilíbrio do pêndulo, para esse caso não teve inclinação. Pois, de fato, não houve aprendizado. Contudo, esta é importante, tendo em vista que é um parâmetro para compararmos uma abordagem apenas reativa à uma implementação que se propõe a aprender sobre o problema. Fica visível a eficácia do aprendizado através de uma implementação do algoritmo

Q learning. Pois é possível observar que se dermos o mesmo peso para estados atuais e futuros ($\gamma = 1$), ainda que a taxa de aprendizagem varie, são obtidos melhores resultados no equilíbrio do pendulo do que em uma abordagem puramente reativa ou não-informada.

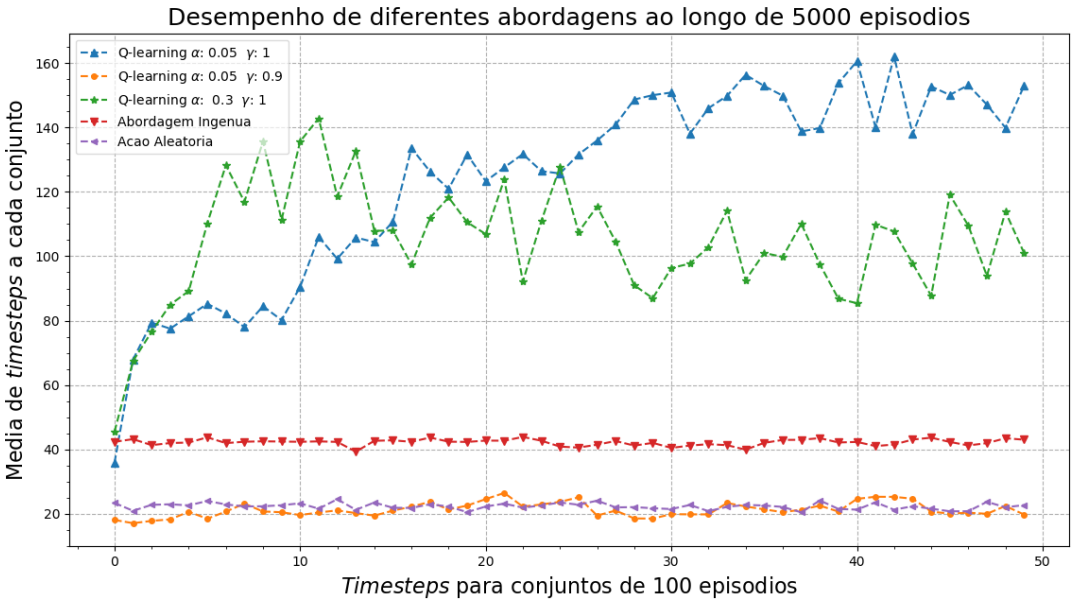

Figura 3: Média tempo de equilíbrio do pêndulo calculada sobre cada 100 episódios consecutivos distintos.

Referências

[Barto et al., 1983] Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846.

[Coppin,] Coppin, B. *Inteligência Artificial*.

[David Goodger, 2001] David Goodger, G. v. R. (2001). Pep 257 – docstring conventions. `https://www.python.org/dev/peps/pep-0257/`. [Online; acessado 29-Abril-2018].

[Luger,] Luger, G. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*.

[Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition.

[Russell and Norvig, 2016] Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited.