Solucionando o FrozenLake do OpenAi com Q-Learning

Víctor C. Colombo[1]

[1]Departamento de Computação – Universidade Federal De São Carlos(UFSCar)
São Carlos – SP – Brazil

victorcora@hotmail.com

Resumo. *Reinforcement Learning (RL) pode ser usado para criar inteligências artificiais capaz de resolver jogos em cenários que préviamente não se havia conhecimento sobre a solução, tendo apenas a informação sobre suas regras e condições de vitória. É apresentada uma inteligência artificial capaz de resolver o problema do FrozenLake por meio da técnica RL Q-Learning.*

1. Introdução

Reinforcement Learning voltou a ter destaque recentemente após a publicação de um artigo, em 2014 (Mnih et. al.), que propôs o uso de redes neurais profundas para criar bots capazes de jogar diversos jogos de Atari de forma que suparessem os humanos em muitos deles. Nesse artigo, foi usado uma variação do Q-Learning, um método de Reinforcement Learning, para treinar a rede neural.

Reinforcement Learning é um tipo de aprendizado de máquina em que não há inicialmente um conjunto de dados para treino, como no aprendizado supervisionado. Para aprender, o sistema usa de tentativa e erro, testando soluções aleatórias para o problema, até que uma delas dê um resultado positivo. Quando isso acontece, o algoritmo guarda as ações que levaram a esse estado considerado positivo e os repete com maior probabilidade em iterações futuras. Após muitas tentativas e erros, o agente passa a conhecer o cenário e tomar ações que o levem à conclusão do jogo.

Como não há conhecimento prévio de quais ações em cada momento são benéficas, algoritmos de RL precisam de uma função conhecida como reward. Essa função retorna um valor que indica ao agente se a ação que ele tomou é positiva no cenário ou não.

O método Q-Learning é, a princípio, uma forma de solucionar um problema de RL com estados discretos (não contínuos). Ele se dá pelo uso de uma tabela, chamada de Q-Table, em que cada entrada dessa tabela é o máximo de reward esperado no futuro se o agente tomar uma ação no estado atual. Com o treinamento desse modelo, a tabela vai passar a prever ações que no futuro resultem em resultados positivos, mesmo que imediatamente não.

2. O Problema

O FrozenLake é um cenário contido no pacote gym do OpenAi. O OpenAi se tornou famoso após criar uma inteligência artificial capaz de vencer os melhores jogadores do mundo em DOTA2 com extrema facilidade.

Nesse cenário, o agente deve atravessar um lago congelado de tamanho 8x8 até

encontrar seu Frisbie. Porém, nesse cenário há locais em que existem buracos. Caso o agente se movimente para uma posição de buraco, ele morre, cujo reward é -1. Caso o agente consiga chegar no objetivo, o reward é 1.

O problema desse cenário se dá no fato de que existe um chance alta da ação tomada não ter o efeito desejado. Há a possibilidade do agente 'escorregar' para a direita ou esquerda do seu movimento, o que pode causar sua morte, e esse detalhe significará resultados interessantes para a Inteligência Artificial treinada.

3. Solucionando o Problema

Para começar, vamos definir o problema como sendo Markoviano. Isso significa que a ação que deve ser tomada no estado presente depende apenas do presente e do futuro. Ações tomadas no passado não devem influenciar a escolha atual. Dessa forma podemos simplificar muito a solução.

Para começar, iniciamos a tabela Q com zeros, que no futuro serão os valores máximos esperados dado cada ação no estado atual. Nesse momento temos um problema. Todos os estados futuros estão marcados como 0, o que fará o algoritmo não saber que caminho tomar.

Para isso, a ideia de exploration vs exploitation é introduzida. No começo, queremos explorar aleatoriamente todo o cenário e ver o que acontece em cada estado e ação tomada. Após algum tempo, as ações se tornarão menos aleatórias e o conhecimento adquirido passará a ser usado para solucionar o problema, ou seja, passaremos à faze do exploitation. Esse método se chama 'epsilon greedy', pois epsilon será nossa taxa de aleatoriedade.

Fazemos então o agente tomar uma ação no cenário que se encontra. Isso gera um possível novo estado, com um reward associado à ação. Temos então que definir uma formula para que a tabela Q seja atualizada e a ação tomada resulte em um aprendizado. Essa fórmula será a equação de Bellman:

$$Q^*(s, a) = Q(s, a) + alpha * [R(s, a) + gamma * Q'(s',a') - Q(s, a)]$$

Sendo *s* o estado atual e *a* a ação tomada. *Gamma* é um fator que leva em conta quanto ao futuro o agente deve 'pensar' antes de tomar uma ação. Valores altos de gamma resultam em o agente fazer escolhas que pareçam ruins a curto prazo mas permita a vitória a longo. Por fim, alpha é a taxa de aprendizado por episódio. Taxas altas podem causar problemas de convergência, então o ideal é manter esse valor pequeno.

Com essas informações podemos então propor a solução para o problema do FrozenLake. Iteramos sob 1 milhão de episódios, até que o reward médio seja no mínimo 78% (valor definido pelo problema para ser considerado finalizado). Não é possível solucionar o problema 100% das vezes devido a questão da aleatoriedade do cenário.

A cada iteração, que chamaremos de episódio, é feita uma ação com probabilidade epsilon de ser aleatória, e 1 – epsilon de ser a ação ideal já achada em Q. Comparamos então o estado final com o estado anterior por meio da equação de bellman e atualizamos Q(s, a).

A cada dez mil episódios, fazemos uma simulação e verificamos se nosso Q

treinado é capaz de solucionar o problema com satisfação na maior parte dos casos. Se sim, finalizamos o aprendizado e temos Q a tabela com as ações que devem ser tomadas em cada estado para garantir bons resultados com frequência.

4. Resultados

Esse algoritmo não é o estado da arte para o RL hoje. O paper sobre o Atari introduz o uso de redes neurais convolucionais para tal problema, o que não é feito aqui. Isso resulta em menor qualidade nos resultados. Além disso, a tabela Q precisa guardar todas as informações de estados e ações possíveis. Nesse caso, são poucos, mas em outros problemas, principalmente contínuos, isso pode gerar bilhões de posições na tabela Q.

Quanto as ações que o algoritmo obtém para solucionar o FrozenLake, vemos que ele toma caminhos que um humano talvez não fizesse inicialmente. Sua primeira ação é tentar se mover em direção à parede superior. Isso pode parecer estranho, mas pelas regras do jogo essa tentativa garante que em algum momento o agente irá se mover para a direita. Isso impede que ele tome o caminho para baixo, que pode ser ruim para o resultado final.

Isso demonstra como algoritmos de Reinforcement Learning podem ser interessantes para achar soluções em problemas que humanos inicialmente não pensariam ou conseguiriam fazer (como vencer as famosas “fases impossíveis” do Mário).

References

OpenAI Gym (04 Dez 2018)

David Silver (2016) “Tutorial: Deep Reinforcement Learning”, https://icml.cc/2016/tutorials/deep_rl_tutorial.pdf

Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2013). Playing Atari with Deep Reinforcement Learning. . http://arxiv.org/abs/1312.5602.

Gaskett, C., Wettergreen, D., Zelinsky, A. (?) “Q-Learning in Continuous State and Action Spaces”

Ameisen, E. (07 Jun 2018) “Reinforcement Learning from scratch” https://blog.insightdatascience.com/reinforcement-learning-from-scratch-819b65f074d8?gi=dc522d5aea91

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Van Hoof, C. (2018) “Learn by example Reinforcement Learning with Gym” https://www.kaggle.com/charel/learn-by-example-reinforcement-learning-with-gym/notebook

Rana, A. (21 Set 2018) “Introduction: Reinforcement Learning with OpenAI Gym https://towardsdatascience.com/reinforcement-learning-with-openai-d445c2c687d2

Gosavi, A. (?) “Neural Networks and Reinforcement Learning” http://web.mst.edu/~gosavia/neural_networks_RL.pdf

Simonini, T. (10 Abr 2018) “Diving deeper into Reinforcement Learning with Q-

Learning” https://medium.freecodecamp.org/diving-deeper-into-reinforcement-learning-with-q-learning-c18d0db58efe