Resolução do Lunar Lander do Gym

Gabriel Ribeiro Pardini, Leandro de Oliveira Peixoto e Marcelo Loiola Lopes Veras

***Abstract*— DQN (Deep-Q Network) é um algoritmo descoberto recentemente que mescla aprendizado por reforço com redes neurais profundas, exibindo feitos impressionantes como jogar jogos de Atari 2600 com desempenho sobre-humano. No presente artigo, é abordada uma resolução usando DQN do problema Lunar Lander do site Machine Learning Gym, um acervo on-line de questões sobre aprendizado de máquina. Nesse sentido foi encontrado um resultado bastante coerente, com cerca de 90% dos episódios de avaliação ocorreram de forma esperada.**

I. Introdução Teórica

O DQN, que significa Deep Q-Network, é um algoritmo específico de deep reinforcement learning (aprendizado por reforço profundo) que utiliza uma combinação do algoritmo Q-learning e redes neurais profundas para aprender ações ótimas em problemas complexos de aprendizado por reforço.

O DQN foi introduzido em 2013 por Volodymyr Mnih e colaboradores, da DeepMind, e foi um marco importante no campo do aprendizado por reforço profundo. Esse algoritimo superou o desempenho humano em vários jogos do console Atari, mostrando a capacidade das redes neurais profundas em aprender a partir de informações visuais.

A principal ideia por trás do DQN é usar uma rede neural profunda para aproximar a função Q, que é a função de valor de ação que mapeia um estado e uma ação para um valor que representa a utilidade dessa ação em relação ao estado atual. A rede neural recebe o estado como entrada e produz um vetor de valores de Q para cada ação possível. A ação com o maior valor de Q é selecionada como a ação a ser tomada, gerando uma recompensa. O esquema geral de qualquer algorítimo de aprendizado por reforço está demonstrado na Figura 1. Para se tomar uma ação, existem inúmeras politicas, uma conhecida é a ϵ-greedy, na qual a ação tomada é escolhida de forma que se tenha probabilidade ϵ de se tomar uma ação aleatória e $1-\epsilon$ de se tomar a ação "gulosa".

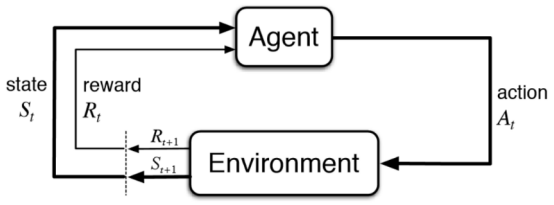

Fig. 1. Esquema básico do aprendizado por reforço [9]

Durante o treinamento, o DQN utiliza uma técnica chamada experiência de repetição (experience replay) para melhorar o aprendizado. A experiência de repetição envolve o armazenamento das transições de estado-ação-recompensa-estado seguinte em um buffer de replay. Em cada etapa de treinamento, uma amostra aleatória é retirada do buffer de replay e usada para atualizar os parâmetros da rede neural.

Além disso, o DQN utiliza uma estratégia chamada target network para estabilizar o treinamento. A target network é uma cópia da rede neural usada para estimar os valores alvo de Q. Durante o treinamento, os pesos da target network são atualizados periodicamente com os pesos da rede principal. Essa técnica ajuda a evitar a instabilidade do treinamento e melhora a convergência.

O DQN utiliza a função de perda chamada erro quadrado médio (mean squared error) para atualizar os pesos da rede neural, minimizando a diferença entre os valores de Q estimados pela rede e os valores alvo. Esses valores alvo são calculados usando a equação de Bellman, que expressa os valores de Q em termos dos valores de Q do próximo estado.

O DQN normalmente usa algo chamado de iteração de política geral, descrita como a conjunção de avaliação de política e iteração de política, para aprender políticas de entrada sensorial de alta dimensão. Por exemplo, um tipo comum de DQN abordada em publicações de tecnologia como a Nature recebe informações sensoriais dos videogames Atari 2600 para modelar os resultados. Isso é feito em um nível muito fundamental, reunindo amostras, armazenando-as e usando-as para reprodução de experiência, a fim de atualizar a rede Q.

Em um sentido geral, as DQN treinam em entradas que representam jogadores ativos em áreas ou outras amostras experientes e aprendem a combinar esses dados com as saídas desejadas. Este é um método poderoso no desenvolvimento de inteligência artificial que pode jogar jogos como xadrez em alto nível ou realizar outras atividades cognitivas de alto nível. Como fora dito, o jogo do Atari 2600 é um bom exemplo de como a IA usa os tipos de interfaces que eram tradicionalmente usados por agentes humanos.

Em resumo, o DQN é um algoritmo de aprendizado por reforço profundo que combina o Q-learning com redes neurais profundas. Ele utiliza uma rede neural profunda para estimar os valores de Q, utiliza a técnica de experiência de repetição para treinamento eficiente e emprega uma target network para estabilizar o treinamento. O DQN tem sido aplicado com sucesso em uma variedade de problemas, desde jogos Atari até controle de robôs, mostrando o poder do aprendizado por reforço profundo. Nesses casos, com o DQN, o jogador de IA se torna mais parecido com um jogador humano ao aprender a alcançar os resultados

desejados.

II. Implementação

A. Problema a ser resolvido

O problema a ser resolvido constitui-se do Lunar Lander da biblioteca de problemas de Machine Learning Gym, o problema se resume a treinar uma rede neural para realizar o pouso de uma nave espacial dentro de um lugar especificado utilizando o algoritmo DQN, assim como especificado na Figura 2.

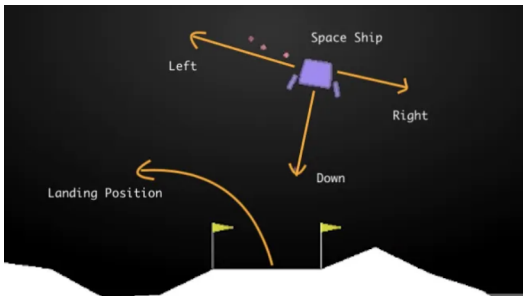

Fig. 2. Especificação do problema a ser resolvido [1]

TABLE I
Espaços de estados para o Lunar Lander

Número do Estado	Estado	Mínimo	Máximo
0	x	-1.5	1.5
1	y	-1.5	1.5
2	v_x	-5	5
3	v_y	-5	5
4	ângulo	$-\pi$	π
5	velocidade angular	-5	5
6	contato direito	-1	1
7	contato esquerdo	-1	1

TABLE II
Espaços de ações possíveis no Lunar Lander

Número da Ação	Ação
0	fazer nada
1	acionar motor direito
2	acionar motor central
3	acionar motor esquerdo

A partir das tabelas I e II temos o espaço de estados e ações possíveis para o problema do Lunar Lander.

B. Especificação da rede neural

A rede neural utilizada é semelhante à utilizada no Laboratório 12 da disciplina CT-213 (Inteligência Artificial para Robótica Móvel) do Instituto Tecnológico de Aeronáutica, entretanto tiveram que ser feitas modificações para que fosse possível realizar o treinamento da rede de forma a ser encontrado um resultado coerente. Dessa forma, de acordo com a Tabela III, pode-se ver a especificação da rede neural que foi utilizada para o treinamento da função ação valor.

A parte mais complexa do algoritmo DQN foi realizar a decisão dos hiperparâmetros da rede, dessa forma, foram testados vários e percebeu-se que o melhor grupo pode ser descrito na Tabela IV.

TABLE III
Especificação da rede neural

Layer	Neurons	Activation Function
Dense	256	ReLU
Dense	128	ReLU
Dense	4	Linear

TABLE IV
Hiperparametros utilizados

Hiperparâmetro	Valor
γ	0.990
ϵ	0.010
ϵ_{decay}	0.985
learning rate	0.001

Com a rede neural e os hiperparâmetros descritos, foi possível ter um treinamento coerente e chegou-se em um resultado bastante satisfatório. Não foi utilizada Reward Engineering pois nos testes utilizados verificou-se que os ganhos que foram colocados pioravam o treinamento.

C. Implementação

Dessa forma, a implementação foi baseada na implementação do lab 12 da disciplina de CT-213, de forma que utilizou-se muita da implementação de execução do treinamento não iria mudar muito, dessa forma a implementação focou-se em encontrar a rede e os hiper parâmetros que se encontrava um treinamento de forma adequada. Dessa forma a rede neural escolhida é utilizada para se predizer a tabela ação valor. Além disso para realizar uma ação é utilizada uma estratégia ϵ-greedy com escalonamento do ϵ de forma a no inicio gerar ações aleatórias para não focar o treinamento em apenas um caminho.

A implementação focou-se no preenchimento de 3 arquivos: dqn_agent.py na qual está uma classe que define os parâmetros do agente, assim como toda a especificação da rede neural; train_lunar_lander.py na qual ocorre o treinamento, sendo escolhido que o treinamento seria limitado em 800 episódios; e eval_Lunar_Lander.py na qual ocorre a avaliação da rede, sendo escolhidos 30 episódios para realizar a avaliação.

III. Resultados e Discussão

Com respeito à implementação supracitada, é interessante exibir os gráficos vinculados à avaliação e ao treinamento, uma vez que este apresenta a evolução da rede neural a partir de pesos aleatórios e aquela apresenta um novo conjunto de dados em contradição à amostra apresentada no treinamento, para conferir se efetivamente a rede ”aprendeu” a resolver o problema em questão para uma boa parcela dos casos. Não foi realizado a plotagem da politica aprendida devido a grande quantidade de estados correspondentes.

A. Treinamento

O objetivo desta parte é justamente treinar a rede neural para estimar a função ação valor, $Q_{s,a}$, para todas as ações em um determinado estado, e selecionar a ação que resulta na maior função ação valor, $argmax_{a \in A} Q_{s,a}$, para implementá-la no próximo instante de tempo. Os resultados pertinentes de um treinamento de 500 episódios, ou seja, após 500 simulações de pouso, estão explicitados na Figura 3.

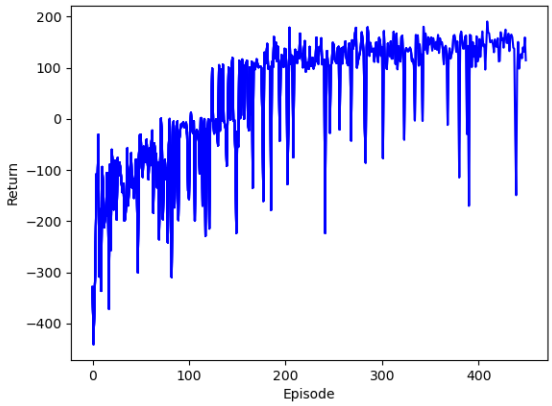

Fig. 3. Gráfico de recompensa acumulada em função do número do episódio de treinamento

A primeira informação que salta aos olhos ao observar o gráfico acima é o fato de, apesar de ruidosa, a recompensa acumulada tender a aumentar conforme o tempo passa. Isso está previsto pelos pesos aleatórios atribuídos na inicialização da rede, sem nenhuma referência inicial, havendo uma probabilidade irrisória de os pesos já estarem ajustados para o problema. Com isso, a rede se ajusta tomando como base a função de perda e ocorre um aumento considerável na recompensa acumulada com o passar do tempo.

Outro ponto que é importante ressaltar são os vales presentes no gráfico, que diminuem consoante aumento de número de episódios. Isso está previsto pela estocasticidade da função $\epsilon - greedy$, que pode vir a escolher uma ação que não seja ótima, ou seja, a que satisfaz $argmax_{a \in A} Q_{s,a}$, sendo isso feito com o intuito de aumentar $exploration$ em detrimento de $exploitation$, havendo dessa maneira chances maiores de se alcançar o ótimo global. Como o próprio valor de ϵ diminui com o passar do treinamento, a chance de uma ação sub-ótima ser escolhida pelo algoritmo DQN também diminui.

B. Avaliação

Com a rede neural treinada, parte-se para a avaliação do algoritmo, em que serão usadas situações inéditas à rede para ver como ela se comporta perante elas. Com resultados mostrados na Figura 4.

Como pode ser visto, a maioria dos casos retornou um valor positivo para a recompensa acumulada, sinal de que a rede realmente foi bem treinada, havendo mínimas chances de problemas como $Overfitting$.

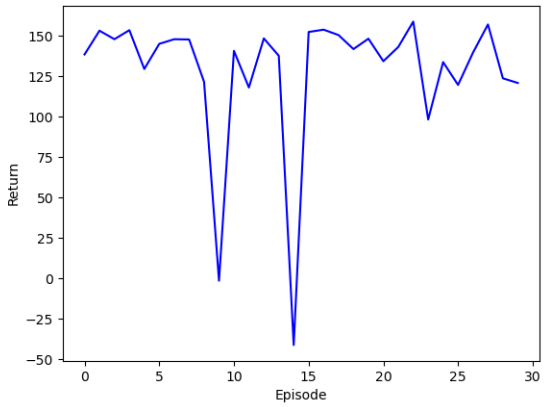

Fig. 4. Gráfico de recompensa acumulada em função do número do episódio de avaliação

IV. Conclusão

Diante dos resultados expostos, o algoritmo DQN se mostra bastante eficaz para a resolução desse problema, uma vez que a seção de treinamento ocorreu conforme previsto na literatura e na seção de avaliação foi observado sucesso em mais de 90% dos episódios.

Isto demonstra que redes neurais conseguem se comunicar bem com algoritmos de aprendizado por reforço, desde que os valores dos hiperparâmetros dessa rede, assim com sua própria arquitetura, sejam escolhidos com cautela.

References

[1] VERNA, Shiva. Train Your Lunar-Lander — Reinforcement Learning — OpenAIGYM. 29, Abril e 2019. Disponível em: https://shiva-verma.medium.com/solving-lunar-lander-openaigym-reinforcement-learning-785675066197. Acesso em: 10 de julho de 2023.

[2] Github — OpenAI Spinning Up. Disponível em: https://spinningup.openai.com/en/latest/. Acesso em: 07 de julho de 2023.

[3] Kohl, Nath and Stone, Peter. Reinforcement Learning for Fast Quadrupedal Locomotion. — ICRA 2004 — Disponível em: http://www.cs.utexas.edu/users/pstone/Papers/bib2html-links/icra04.pdf. Acesso em: 07 de julho de 2023.

[4] S. Sutton, Richard ; G. Barto, Andrew: Reinforcement Learning: An Introduction Second Edition: 2017. MIT Press.

[5] Silver, David — Lectures on Reinforcement Learning, 2015 — Disponível em: https://www.davidsilver.uk/teaching/. Acesso em: 08 de julho de 2023.

[6] Silver, David — Gradient Temporal Difference Networks — EWRL, 2012.

[7] Maddison, Chris / Huang, Aja / Sutskever, Ilya / Silver, David — Move Evaluation in Go Using Deep Convolutional Neural Networks — ICLR, 2015.

[8] Sun, Ron / Silver, David / Tesauro, Gerald / Huang, Guang-Bin — Introduction to the special issue on deep reinforcement learning:An editorial — Neural Networks, Vol. 107, p. 1-2 — 2018.

[9] MAXIMO, Marcos. Inteligência Artificial para Robótica Móvel: Introdução ao Aprendizado por Reforço. 28 de Maio de 2023. Disponível em: https://drive.google.com/file/d/1dC7RNeetKRIxfAMJlI08OKgR3YLji3W3 /view. Acesso em: 17 de Julho de 2023.