Insper
Bacharelado em Engenharia da Computação

Letícia Coêlho , Matheus Oliveira e Nívea Abreu

Reinforcement Learning : Projeto 1

São Paulo
Abril de 2023

Sumário

1 Qual é a característica de Q-Learning e Deep Q-Learning destacada pelos autores do artigo?

Q-learning (Watkins 1989) é um dos algoritmos de aprendizado por reforço conhecido por aprender valores de ação irrealisticamente , pois o mesmo inclui uma etapa de maximização sobre os valores de ação estimados, que tendem a preferir valores superestimados a subestimados.[1]

O Deep Q-Learning (DQN) combina o algorítimo Q-learning com uma rede neural profunda flexível , que fornece aproximação de função flexível com potencial para um baixo erro de aproximação assintótica, e o determinismo dos ambientes evita os efeitos nocivos do ruído. Apesar disso , o DQN às vezes superestima substancialmente os valores das ações.

Uma rede Q profunda (DQN) é uma rede neural com multiplas camadas que, para um determinado estado S, gera um vetor de valores de ação, sendo esses os parâmetros de rede.[2]

Dois elementos importantes do algoritmo DQN Mnih et al. (2015) é a rede alvo e o uso de replay de experiência. A rede de destino, com parâmetros , é a mesma da rede online, exceto que seus parâmetros são copiados a cada etapa da rede online. Este mecanismo de replay de experiência consiste em armazenar as experiencias do agente ao longo do treinamento. A repetição da experiência melhoram drasticamente o desempenho do algoritmo.

2 Qual é a ideia principal do Double Deep Q-Learning e como essa ideia deve ser implementada? Mostre a ideia através de um pseudo-código.

Nos algorítimos Q-learning e DQN, o operador *max* , usa os mesmos valores tanto para selecionar uma ação e para avaliar uma ação, o que torna mais provável a seleção de valores superestimados. Para evitar isso, podemos separar a seleção da avaliação.

Dessa forma, a ideia do Double Q-learning é reduzir as superestimativas o decompondo a operador *max* em seleção de ação e avaliação da ação.

Algorithm 3 Algoritmo Deep Q-Network

Inicializar replay memory D com capacidade N
Rede neural Q inicializada com parametros θ aleatórios
for cada episodio **do**
    Inicializar o estado inicial s_t
    for $t = 1, T$ **do**
        Escolher uma ação a_t usando a politica derivada de Q (e.g., ϵ-greedy)
        Executar a ação a_t, observar r_t, s_{t+1}
        Armazenar a tupla de transições (s_t, a_t, r_t, s_{t+1}) na memoria D
        Selecionar aleatóriamente um mini-lote de D contendo K experiências (s_t, a_t, r_t, s_{t+1}). (Experience Replay)
        for cada experiência no mini-lote **do**
            if estado s_t é terminal **then**
                y $\leftarrow$ r
            else
                $y \leftarrow r + \gamma \max Q(s_{t+1}, a^{'}; \theta)$
            end if
            Atualizar os parametros do modelo com $(y - Q(s, a))^2$
        end for
        $s_t \leftarrow s_{t+1}$
    end for
end for

Figura 1: Pseudo-código Deep Q-learning [2]

O Double Q-learning (van Hasselt 2010), duas funções de valor são aprendidas atribuindo experiências aleatoriamente para atualizar uma das duas funções de valor, resultando em dois conjuntos de pesos. Para cada atualização, um conjunto de pesos é usado para determinar a política gananciosa(seleção da ação) e o outro para determinar seu valor (avaliação da ação).[1]

O Double Deep Q-Network tem seu mecanismo de aprendizagem semelhante ao DQN, sua diferença pincipal estão no fato do DDQN usa duas redes neurais denominadas online e target network, sendo que a cada ***n*** iterações¸os os pesos da rede online são copiados para target com intuito de manter a instabilidade no processo de aprendizagem do agente.[1]

Algorithm 4 Algoritmo Double Deep Q-Network

Inicializar replay memory D com capacidade N
Rede neural Q inicializada com parametros θ aleatórios
Rede neural target $\hat{Q}$ inicializada com parametros $\theta^{-} = \theta$
for cada episodio **do**
    Inicializar o estado inicial s_t
    for $t = 1, T$ **do**
        Escolher uma ação a_t usando a politica derivada de Q (e.g., ϵ-greedy)
        Executar a ação a_t, observar r_t, s_{t+1}
        Armazenar a tupla de transições (s_t, a_t, r_t, s_{t+1}) na memoria D
        Selecionar aleatóriamente um mini-lote de D contendo K experiências (s_t, a_t, r_t, s_{t+1}). (Experience Replay)
        for cada experiência no mini-lote **do**
            if estado s_t é terminal **then**
                y $\leftarrow$ r
            else
                $y \leftarrow r + \gamma \max \hat{Q}(s_{t+1}, a'; \theta^{-})$
            end if
            Atualizar os parametros do modelo com $(y - Q(s, a; \theta))^2$
        end for
        A cada K interações $Q(s, a; \theta)$ = $\hat{Q}(s, a; \theta^{-})$
        $s_t \leftarrow s_{t+1}$
    end for
end for

Figura 2: Pseudo-código Double Deep Q-learning [2]

3 Como foram implementados os testes empíricos neste artigo?

Os teste empíricos consistem em jogos Atari 2600, usando o Arcade Learning Environmente. O objetivo é que um único algoritmo, com um conjunto fixo de hiperparâmetros, aprenda a jogar cada um dos jogos separadamente da interação, tendo somente os pixels da tela como entrada. Isso trata-se de um teste robusto, visto que as entradas possuem alta dimensão e a mecânica dos jogos varia consideravelmente de jogo para jogo. As melhores soluções encontradas depende fortemente do algorítimo de aprendizagem implementado.[1]

As condições de treinamento dos algorítimos DQN e DDQN foram as exatas condições descritas descritas por Mnih et al , onde a arquitetura da rede é uma rede neural convolucional com 3 camadas de convolução e uma camada oculta totalmente conectada, com aproximadamente 1,5 milhões de parâmetros no total.[1]

A rede recebe os últimos quatro quadros como entrada e gera um valor de ação para cada ação tomada. A rede foi treinada em uma única GPU para 200 milhões de quadros em todos os jogos Atari 2600.[1]

4 Quais foram as principais contribuições do artigo?

Segundo o artigo, o mesmo possui 5 contribuições:

1. Foi mostrado que o Q-learning pode ser superotimista em problemas de grande escala (mesmo em ambiente determinísticos), devido aos erros de estimativa inerentes ao aprendizado.[1]

2. Analisando o jogo Atari observou-se que esse superotimismo é mais comum e grave do que se imaginava.[1]

3. Mostrou-se que o algorítimo Double Q-learning pode ser usado em escala para reduzir com esse excesso de otimismo. Isso resulta aprendizado mais conviável e com maior estabilidade.[1]

4. Criação de uma implementação chamada Double Deep Q-learning (DDQN), que utiliza a arquitetura existente e a rede neural profunda do algoritmo DQN, não sendo necessário reder ou parâmetros adicionais.[1]

5. Mostrou-se que DDQN encontra as melhores políticas, obtendo novos resultados nos jogos Atari 2600.[1]

5 Como podemos verificar se o Double Deep Q-Learning tem um aprendizado mais estável e pode encontrar melhores políticas?

Em geral, o DDQN é considerado mais estável do que o DQN porque ele usa duas redes neurais separadas: uma rede para avaliar o valor de ação e outra rede para selecionar a ação a ser executada. Isso ajuda a reduzir a sobrevalorização de valores de ação, que é um problema comum no DQN. Conforme podemos conferir nos dois gráficos do DQN, quanto do DQQN, este último tem uma oscilação bem menor de valores, o que permite a conclus ao que a tendência de um aprendizado neste modelo é mais sólida, devido a segmentação de etapas causada pela divisão de tarefas entre o armazenamento de Q valores e a seleção da próxima ação, permitindo um aprendizado mais conciso do agente.

6 Implemente o algoritmo Double Deep Q-Learning e valide nos seguintes ambientes: Cart Pole e Lunar Lander.Você obteve o comportamento esperado? Justifique sua resposta. Se não, na opinião do grupo, o que faltou implementar para atingir o comportamento esperado?

Não foi obitdo o comportamento esperado, isto é, os agentes conseguirem executar de maneira completa o enviroment. Provavelmente não foi possível pela quantidade pequena de episódios que utilizamos (devido a falta de otimização do keras com a máquina do grupo), que não permitiu ao agente, tempo suficiente para conseguir adequar os pesos da rede para conseguir consolidar um aprendizado que seja adequado para este conseguir executar bem o ambiente. Para possíveis melhorias, poderiamos aumentar o número de episódios para treinamento (1000 para o LunarLander por exemplo) e também diminuir a frequência de atualização dos pesos da rede do modelo, permitindo ao modelo explorar melhor os Q valores antes da atualização, em um valor aproximado de 100-150 episódios.

Referências

[1] Hado van Hasselt, Arthur Guez e David Silver. "Deep Reinforcement Learning with Double Q-Learning". Em: Proceedings of the AAAI Conference on Artificial Intelligence 30.1 (mar. de 2016). URL: `https://ojs.aaai.org/index.php/AAAI/article/view/10295`.

[2] Natalia Freitas Araújo Rodrigo Moraes Rodrigues Otavio Noura Teixeira. "Tecnicas de Aprendizagem por Reforço na resolução do Mundo de Wumpus". Em: Faculdade de Engenharia de Computaçao – Universidade Federal do Pará (UFPA) (), pp. 0–18.