Deep Q-Network: "Roteiro"

Introdução

Deep Q-Learning é uma técnica de aprendizado de máquina que combina redes neurais convolucionais com aprendizado por reforço.

O objetivo é ensinar uma inteligência artificial a obter sucesso em um ambiente a partir de imagens, ou seja, **sem saber as regras nem receber instruções explícitas**.

A rede neural extrai características relevantes da imagem e estima os melhores valores de ação, enquanto o aprendizado por reforço avalia as decisões com base nas recompensas recebidas.

Essa abordagem se assemelha ao modo como humanos aprendem a jogar um jogo novo: explorando, testando ações e aprendendo com os erros.

História e Motivação

Antes do surgimento do DQN, os algoritmos de aprendizado por reforço (como o Q-*Learning* clássico) funcionavam bem apenas em ambientes com **espaços de estados pequenos ou discretos**, pois era necessário manter uma **tabela** $Q(s, a)$ com todos os pares possíveis de estado e ação - o que se torna inviável em problemas com muitos estados (como jogos com imagens, por exemplo).

A solução para esse problema veio em **2013**, quando pesquisadores da ***DeepMind*** propuseram o uso de **redes neurais profundas para aproximar a função** Q, no artigo *Playing Atari with Deep Reinforcement Learning*. Essa abordagem deu origem ao que chamamos hoje de ***Deep Q-Network*** **(DQN)**.

Em **2015**, o artigo formal publicado na revista *Nature* consolidou o impacto da técnica ao demonstrar que uma IA poderia **aprender a jogar diversos jogos do Atari 2600 diretamente a partir dos pixels da tela e da pontuação**, **sem nenhum conhecimento prévio das regras do jogo ou engenharia de atributos**. Ou seja, **a única entrada do agente eram as imagens do jogo**, e ele aprendia a jogar **somente por tentativa e erro**, ajustando suas ações com base nas recompensas recebidas.

Essa foi a **primeira vez** que uma rede neural foi usada com sucesso em um ambiente de aprendizado por reforço de alta complexidade, superando até mesmo jogadores humanos em vários jogos. Isso marcou um **avanço histórico na Inteligência Artificial**, abrindo caminho para aplicações mais avançadas em jogos, robótica e sistemas autônomos.

Ideia Geral do Algoritmo

A principal inovação do DQN foi substituir a tabela $Q(s, a)$ por uma **rede neural** que aproxima os valores Q para cada ação, dados os estados.

> $Q(s, a)$:
>
> Onde:
>
> - s é o estado (*state*)
> - a é a ação (*action*)

Como funciona:

- O agente observa uma **imagem** (estado atual do jogo).
- Essa imagem é processada por uma **CNN (Rede Neural Convolucional)**, que extrai as principais características visuais.
- A rede retorna os valores estimados $Q(s, a)$ para cada ação possível.
- O agente escolhe a ação com maior valor (ou explora aleatoriamente com uma estratégia como ε-*greedy*).
- A ação é executada, e o agente recebe uma recompensa e observa o novo estado.
- A experiência `(estado, ação, recompensa, novo estado)` é armazenada em uma **memória de replay**.

Durante o treinamento, o agente amostra essas experiências da memória e atualiza os pesos da rede com base na **diferença entre o valor estimado e o alvo**, calculado com base no Q-*learning*:

$$\text{target} = r + \gamma \cdot \max_{a'} Q_{\text{target}}(s', a')$$

O que possibilitou o DQN?

O uso direto de redes neurais com aprendizagem por reforço (*RL - Reinforced Learning*) **não funcionava bem devido** à **instabilidade do treinamento**. O DQN superou esses problemas com três soluções principais:

Experience Replay:

- Armazena experiências anteriores em um *buffer*.
- Durante o treinamento, amostra-se um *mini-batch* aleatório.
- Isso **quebra correlações temporais** e melhora a eficiência de aprendizado.

Target Network:

- Uma cópia da rede principal (chamada rede-alvo) é usada para calcular os valores-alvo de Q.

- Ela é atualizada com menos frequência, **estabilizando o processo de atualização dos pesos**.

ε-*Greedy*:

- Estratégia de exploração: com probabilidade ε, o agente escolhe uma ação aleatória.
- Com probabilidade 1 – ε, escolhe a ação com maior valor Q estimado.
- ε é decrescido ao longo do tempo.

Definição Formal do Algoritmo

> **Algorithm 1** Deep Q-learning with Experience Replay
>
> Initialize replay memory $\mathcal{D}$ to capacity N
> Initialize action-value function Q with random weights
> **for** episode $= 1, M$ **do**
> Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
> **for** $t = 1, T$ **do**
> With probability ϵ select a random action a_t
> otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
> Execute action a_t in emulator and observe reward r_t and image x_{t+1}
> Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
> Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
> Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
> Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
> Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
> **end for**
> **end for**

1. Inicializa a rede Q com pesos aleatórios.
2. Inicializa a rede-alvo (*target*) com os mesmos pesos.
3. Para cada episódio:
   - Observa o estado.
   - Escolhe uma ação com ε-*greedy*.
   - Executa a ação e observa a recompensa e o novo estado.
   - Armazena essa transição no *replay buffer*.
   - Amostra um *mini-batch* do *buffer* e atualiza os pesos da rede Q.
   - Atualiza periodicamente a rede-alvo.

   Esse ciclo é repetido até que o agente aprenda uma política eficiente.

Conclusão e Impacto

O DQN foi um **divisor de águas no Aprendizado por Reforço**. Ele mostrou que é possível aplicar redes neurais profundas em tarefas de decisão complexas, mesmo com entradas de alta dimensão como imagens.

Graças a ele, surgiram variantes mais poderosas como:

- ***Double* DQN** (reduz superestimação dos valores Q),

- ***Dueling* DQN** (separa valor do estado e vantagem da ação),
- ***Rainbow* DQN** (combina múltiplas melhorias).

Além dos jogos Atari, a abordagem DQN foi aplicada em robótica, simulações físicas, carros autônomos e outros contextos de controle inteligente.

> Slide disponível no Canva.

Referências

Artigo de 2013 – DeepMind:
MNHI, Volodymyr et al. *Playing Atari with Deep Reinforcement Learning*. arXiv preprint arXiv:1312.5602, 2013. Disponível em: https://arxiv.org/abs/1312.5602. Acesso em: 31 maio 2025.

Artigo de 2015 – DeepMind (Nature):
MNHI, Volodymyr et al. Human-level control through deep reinforcement learning. *Nature*, v. 518, p. 529–533, 2015. DOI: 10.1038/nature14236. Disponível em: https://www.nature.com/articles/nature14236. Acesso em: 31 maio 2025.

Trabalho de Conclusão de Curso – USP (2018):
TAVARES, Vikttor Cruz. *Uma introdução ao Aprendizado por Reforço Profundo: estudo e implementação do algoritmo Deep Q-Network*. Universidade de São Paulo, Instituto de Matemática e Estatística, 2018. Disponível em: https://bccdev.ime.usp.br/tccs/2018/viktt/monografia_revisada.pdf. Acesso em: 31 maio 2025.