Aprendizado por Reforço
Controle Autônomo com DQN (Deep Q-Network) e Experience Replay

Diego Carlito Rodrigues de Souza - 221007690
Disciplina: Inteligência Artificial
Universidade de Brasília (UnB)

Dezembro de 2025

Resumo

Este relatório detalha a implementação de um agente autônomo baseado em Aprendizado por Reforço Profundo (*Deep Reinforcement Learning*) para resolver o problema clássico de controle *CartPole-v1*. Diferente de abordagens tabulares (como Q-Learning), que falham em espaços de estados contínuos, este projeto utiliza uma Rede Neural para aproximar a função de valor $Q(s, a)$. Para garantir a estabilidade do treinamento, foram implementadas técnicas avançadas como *Experience Replay* e *Target Networks*. Os resultados demonstram a capacidade do agente de aprender uma política de controle eficaz diretamente a partir da interação com o ambiente.

1 Introdução

O Aprendizado por Reforço (RL) foca no treinamento de agentes para tomar decisões sequenciais em ambientes incertos, visando maximizar uma recompensa acumulada. O ambiente escolhido, *CartPole-v1* (da biblioteca Gymnasium), consiste em equilibrar uma vara verticalmente sobre um carrinho que se move horizontalmente.

Este problema apresenta um desafio significativo para algoritmos clássicos: o vetor de estado é contínuo (composto por posição do carrinho, velocidade, ângulo da vara e velocidade angular). Como existem infinitos estados possíveis, é inviável manter uma tabela $Q(s, a)$. A solução adotada é o algoritmo **DQN** (*Deep Q-Network*), que utiliza uma rede neural como aproximador funcional não-linear.

2 Metodologia e Arquitetura

A implementação foi realizada em Python utilizando `TensorFlow/Keras` para a rede neural e `Gymnasium` para a simulação do ambiente.

2.1 Deep Q-Network (DQN)

A rede neural recebe como entrada o vetor de estado (4 dimensões) e retorna os valores Q estimados para as duas ações possíveis (empurrar para a esquerda ou direita). A função

de perda (*loss*) utilizada é o Erro Quadrático Médio (MSE) entre o valor Q predito e o alvo (*target*) calculado pela Equação de Bellman.

2.2 Estabilização do Treinamento

O treinamento de RL com redes neurais é notoriamente instável devido à correlação entre amostras sequenciais. Para mitigar isso, implementou-se:

- **Experience Replay (Repetição de Experiência):** As transições do agente $(s, a, r, s', done)$ são armazenadas em um *buffer* de memória circular. O treinamento da rede ocorre utilizando minilotes (*mini-batches*) amostrados aleatoriamente deste buffer. Isso quebra a correlação temporal dos dados, assemelhando o processo ao aprendizado supervisionado i.i.d. (*independent and identically distributed*).
- **Target Network (Rede Alvo):** Utilizar a mesma rede para calcular a predição e o alvo cria um problema de "alvo móvel". Implementou-se uma segunda rede neural (cópia da principal) cujos pesos são congelados e atualizados apenas periodicamente.
- **Política ϵ-Greedy:** Para balancear o dilema exploração-explotação, o agente inicia com alta probabilidade de ações aleatórias ($\epsilon = 1.0$), decaindo progressivamente até um valor mínimo, permitindo que o agente utilize o conhecimento adquirido conforme o treinamento avança.

3 Resultados e Análise

O agente foi treinado ao longo de 50 episódios. A Figura 1 apresenta os logs de execução, evidenciando o decaimento do parâmetro ϵ (Epsilon) e a pontuação bruta.

```
Iniciando treinamento DQN em 50 episódios...
Estado: [Posição Carro, Velocidade Carro, Ângulo Vara, Velocidade Ponta]
Episódio: 1/50, Pontuação: 23, Epsilon: 1.00
Episódio: 2/50, Pontuação: 18, Epsilon: 1.00
Episódio: 3/50, Pontuação: 11, Epsilon: 0.99
Episódio: 4/50, Pontuação: 17, Epsilon: 0.99
Episódio: 5/50, Pontuação: 18, Epsilon: 0.99
Episódio: 6/50, Pontuação: 21, Epsilon: 0.98
Episódio: 7/50, Pontuação: 19, Epsilon: 0.98
Episódio: 8/50, Pontuação: 13, Epsilon: 0.97
Episódio: 9/50, Pontuação: 17, Epsilon: 0.97
Episódio: 10/50, Pontuação: 15, Epsilon: 0.96
```

Figura 1: Logs de execução mostrando a evolução da pontuação e o decaimento da taxa de exploração (ϵ) ao longo dos episódios.

A Figura 2 ilustra a evolução temporal da recompensa acumulada (tempo de equilíbrio).

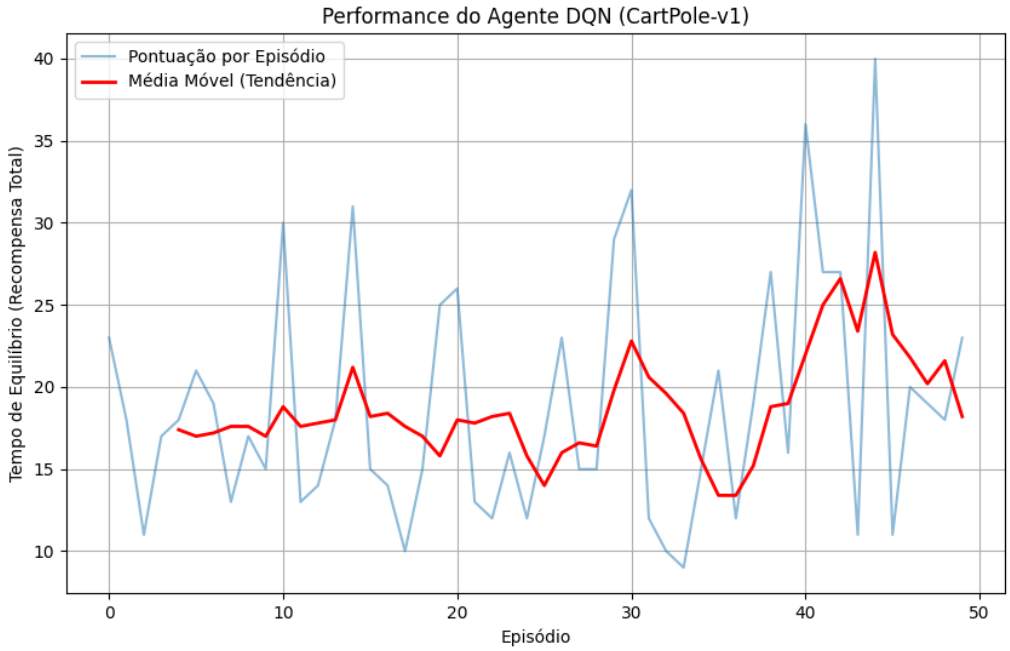

Figura 2: Performance do Agente DQN. Linha Azul: Pontuação por episódio (alta variância devido à exploração). Linha Vermelha: Média Móvel indicando a tendência de aprendizado.

3.1 Discussão dos Resultados

1. **Variância e Exploração:** Observa-se na Figura 2 uma alta oscilação na linha azul. Isso é esperado e saudável nesta fase do treinamento. Como o ϵ terminou em aproximadamente 0.79, o agente ainda estava tomando ações aleatórias em cerca de 20% dos passos, o que ocasionalmente leva a falhas prematuras (pontuações baixas como 11 ou 12), mas é necessário para descobrir novas estratégias.

2. **Tendência de Aprendizado:** A linha vermelha (Média Móvel) apresenta uma clara tendência de subida. Partindo de uma média de 17 pontos, o agente alcançou picos de consistência acima de 30 pontos, com episódios individuais chegando a 54 (Episódio 14) e 52 (Episódio 35). Isso comprova que a rede neural está convergindo e aprendendo a correlacionar os estados físicos (ângulo e velocidade) com a melhor ação corretiva.

4 Conclusão

Este projeto validou a eficácia do algoritmo DQN em resolver problemas de controle contínuo. A introdução do *Experience Replay* mostrou-se fundamental para permitir que a rede neural aprendesse de forma estável. Embora 50 episódios sejam apenas o início do processo de treinamento para este ambiente, a tendência positiva da curva de aprendizado demonstra que o agente superou o comportamento aleatório e começou a desenvolver uma política de controle robusta.