Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT213
Aluno: Arthur de Sousa Vianna

Relatório do Laboratório 12 - Deep Q-Learning

1 Breve Explicação em Alto Nível da Implementação

O objetivo deste relatório é descrever a implementação da resolução do problema *Mountain Car* – problema clássico de *Reinforcement Learning* – no ambiente da OpenAI Gym em Python utilizando algoritmo de Deep Q-Learning. A descrição do espaço de estados, de ações e das recompensas do problema pode ser consultado em `https://github.com/openai/gym/wiki/MountainCar-v0`. O algoritmo de Deep Q-Networks (DQN) consiste em uma adaptação do algoritmo de Q-Learning utilizando-se uma rede neural como aproximador da função ação-valor em vez de uma abordagem tabular. A cada treinamento, atualiza-se o vetor de parâmetros da rede neural que tem como saída a aproximação da função ação-valor. A DQN tem características como *experience replay* – armazenar transições em um *buffer* e, a cada treinamento, amostrar um *mini-batch* com o objetivo de utilizar amostras descorrelacionadas – e *Fixed Q-Targets* – congelar os pesos da rede neural para estimar o *target* e estabilizar o treinamento. Os hiperparâmetros da rede neural utilizada foram $\gamma = 0,95$ (*temporal difference*), $\varepsilon = 0,5$ (política ε-*greedy*), $\alpha = 0,001$ (*learning rate*) e um *scheduling* de ε com taxa de decaimento 0,98, com valor mínimo de 0,01.

2 Figuras Comprovando Funcionamento do Código

2.1 Sumário do Modelo

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 24)                72

dense_2 (Dense)              (None, 24)                600

dense_3 (Dense)              (None, 3)                 75

=================================================================
Total params: 747 (2.92 KB)
Trainable params: 747 (2.92 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figura 1: Sumário da arquitetura do DQN.

2.2 Retorno ao Longo dos Episódios de Treinamento

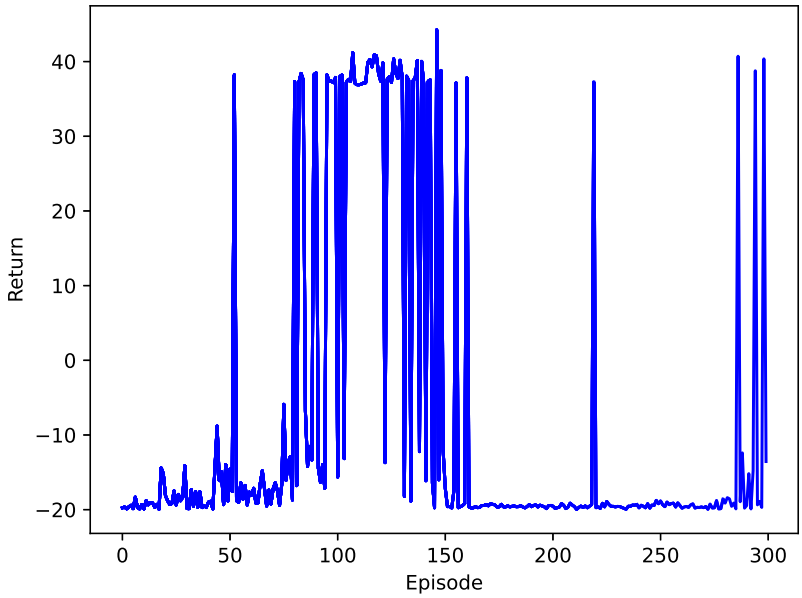

Figura 2: Retorno ao longo dos episódios de treinamento do DQN.

2.3 Política Aprendida pelo DQN

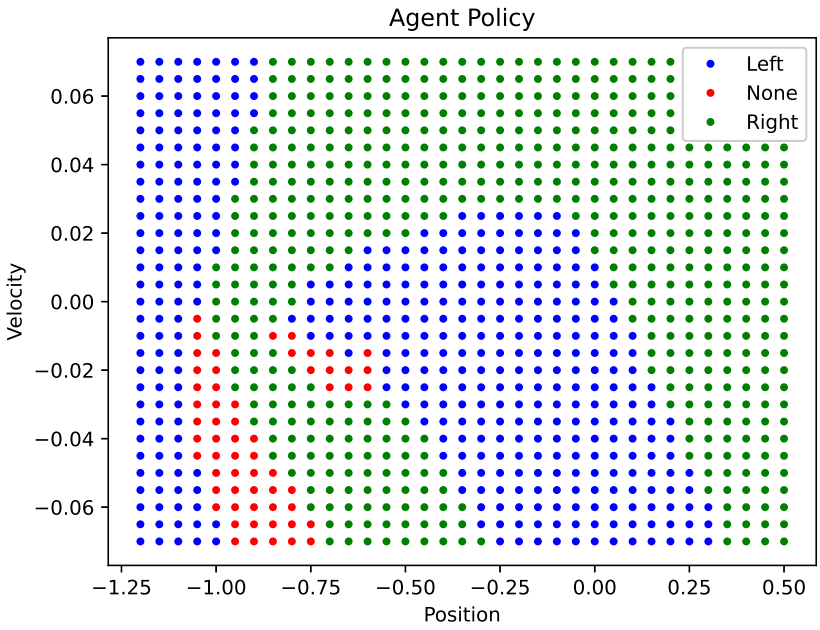

Figura 3: Política ótima aprendida pelo DQN após o treinamento.

2.4 Retorno de 30 Episódios Usando a Rede Neural Treinada

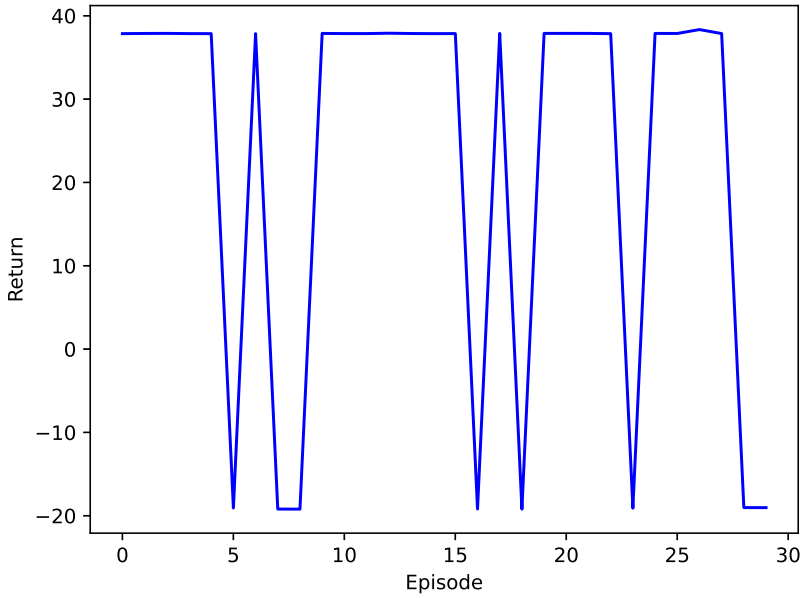

Figura 4: Retorno de 30 episódios utilizando a rede neural treinada.

3 Discussão dos Resultados

A implementação do algoritmo Deep Q-Networks (DQN) para a resolução do problema *Mountain Car* demonstrou tanto o potencial quanto os desafios inerentes a métodos de aprendizado por reforço com aproximadores de função. A análise dos resultados indica que o agente foi capaz de aprender uma política de controle eficaz, embora o processo de treinamento tenha exibido instabilidades notáveis.

O gráfico de retorno ao longo dos episódios de treinamento (Figura 2) ilustra um comportamento característico de implementações de DQN. Inicialmente, o agente explora o ambiente com um desempenho baixo. Subsequentemente, observa-se uma fase de aprendizado rápido, em que o agente atinge retornos elevados com frequência, indicando que descobriu uma estratégia para alcançar o objetivo. No entanto, essa fase é seguida por um longo período de colapso de desempenho, um fenômeno conhecido como "esquecimento catastrófico". Isso ocorre quando as atualizações na rede neural, baseadas em novas experiências, acabam por corromper a política bem-sucedida que havia sido aprendida. Essa instabilidade é um desafio comum em DQN e sugere que os hiperparâmetros, como a taxa de aprendizado ($\alpha = 0,001$) e o decaimento de ε, poderiam ser ajustados para obter uma convergência mais estável.

Apesar da instabilidade no treinamento, a política final aprendida pelo agente (Figura 3) é notavelmente lógica e eficaz. O gráfico da política do agente demonstra que o DQN aprendeu a estratégia fundamental para o *Mountain Car*: aplicar força na mesma direção da velocidade atual para acumular momento. O agente empurra para a direita (pontos verdes) quando a velocidade

é positiva e para a esquerda (pontos azuis) quando a velocidade é negativa. A emergência dessa estratégia de controle complexa a partir do zero comprova que o algoritmo estava, de fato, aprendendo a estrutura do problema.

A avaliação final do agente treinado (Figura 4) reforça essa dualidade. O gráfico mostra que, mesmo após o treinamento, a política não é infalível, apresentando sucesso em 22 dos 30 casos de testes, com acurácia de cerca de 73%. Isso indica que a função ação-valor aproximada pela rede neural ainda possui imperfeições, tornando a política frágil para certas condições. Conclui-se que o experimento foi bem-sucedido em demonstrar a capacidade do DQN em resolver o problema, mas também expôs de forma prática os desafios de estabilidade que motivaram o desenvolvimento de melhorias no algoritmo, como o uso de *experience replay* e *fixed Q-targets*.