Laboratório 13 – *Deep Q-Learning*

Inteligência Artificial para Robótica Móvel – CT-213

Aluno: Caio Graça Gomes

Professor: Marcos Ricardo Omena de Albuquerque Maximo

INTRODUÇÃO:

Nesse laboratório, teve-se por objetivo resolver o problema do *Mountain Car* usando o algoritmo *Deep Q-Networks* (DQN). Nesse problema, um carro começa numa posição aleatória em uma curva montanhosa, próximo a um vale. O objetivo do carro é alcançar o ponto mais alto à direita.

METODOLOGIA E DESCRIÇÃO EM ALTO NÍVEL DO ALGORITMO UTILIZADO:

Para realização do problema, foi considerado que o carro tem espaço de estados e de ações conforme apresentados nas tabelas a seguir:

Número do estado	**Estado**	**Mínimo**	**Máximo**
0	posição	-1,2	0,6
1	velocidade	-0,07	0,07

Tabela 1: espaço de estados do *Mountain Car*.

Número da ação	**Ação**
0	Empurrar para a esquerda (*push left*)
1	Sem empurrar (*no push*)
2	Empurrar para a direita (*push right*)

Tabela 2: espaço de ações do *Mountain Car*.

Além disso, considerou-se que a recompensa do aprendizado fosse -1 por passo de tempo, até que o objetivo (0,5 na direita) fosse atingido. O limite esquerdo funcionou como uma parede (colisão elástica), o estado inicial foi uma posição aleatória com probabilidade uniforme entre -0,6 e -0,4, com velocidade nula. Por fim, o episódio terminava quando o carro atingia a posição objetivo ou eram efetuados 200 passos de tempo na execução.

Para calcular a função ação-valor, usou-se uma aproximação com rede neural, ideia principal do DQN. Utilizou-se o *experience replay* e o *Fixed Q-targets* para o DQN. A rede neural consistiu de três camadas e teve arquitetura conforme a tabela a seguir:

Layer	**Neurons**	**Activation Function**
Dense	24	ReLU
Dense	24	ReLU
Dense	`action_size`	Linear

Tabela 3: arquitetura da rede neural usada para aproximar a função ação-valor $\hat{q}(s,a)$.

O laboratório consistiu em implementar três métodos/funções, foram eles:

1) `make_model(self)` da classe `class DQNAgent`: Constrói a rede neural conforme a arquitetura da tabela 3;
2) `act(self, state)` da classe `class DQNAgent`: Escolhe uma ação a ser tomada com base em uma política *epsilon greedy*;
3) `reward_engineering_mountain_car(state, action, reward, next_state, done)`: computa a recompensa para permitir um treinamento mais rápido.

O DQN foi inicialmente treinado em `train_dqn.py` e posteriormente executado para problema do *Mountain Car* em `evaluate_dqn.py`.

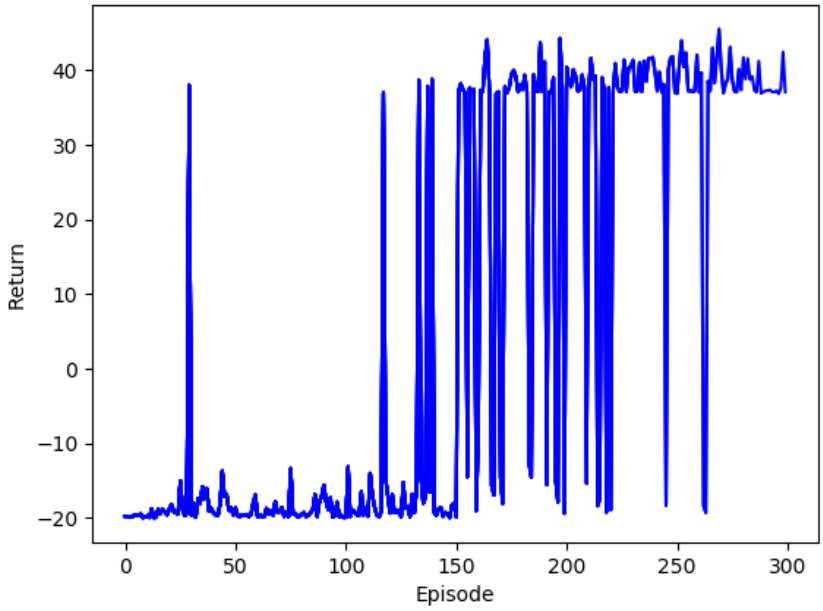
RESULTADOS:

Resultados do train_dqn.py:

Figura 1: Sumário da rede neural construída.

```
_________________________________________________________________
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
Trainable params: 747
Non-trainable params: 0
_________________________________________________________________
No weights found from previous learning session.
```

Figura 2: Gráfico do retorno do carrinho em cada episódio do treinamento.



Veja que até o centésimo episódio ele conseguiu concluir o objetivo apenas uma vez (em torno do trigésimo), a partir do 150º muitos episódios tiveram sucesso e a partir do 230º quase todos conseguiram concluir. Essa descontinuidade brusca do gráfico mostra que a função recompensa adotada foi muito resultadista, mas foi o melhor encontrado para o dado problema.

Resultados do evaluate_dqn.py:

Figura 3: Gráfico do retorno obtido pelo carrinho nos episódios.

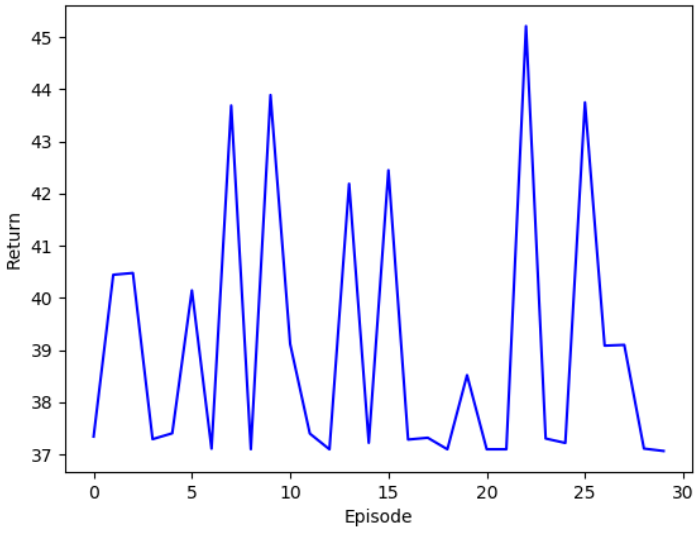
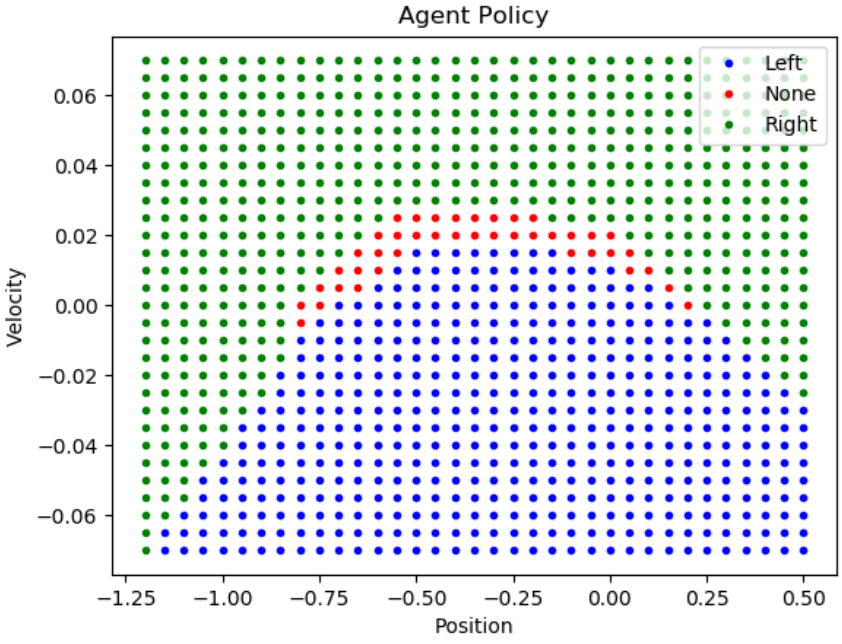


Figura 4: Gráfico da posição velocidade e política adotada pelo carrinho.



Analisando os resultados, nota-se que o carrinho conseguiu concluir o percurso em todos os 30 episódios. Apesar do resultado fantástico, tamanho desempenho foi atingido com um pouco de sorte, pois existe ruído para a posição inicial do carro. Além disso, é muito interessante a análise da figura 4. Quando o carro está à esquerda, quase sempre a política ideal é ir a direita, o que faz sentido. Conforme a posição aumenta no sentido da direita, passa a fazer sentido ir para a esquerda para velocidades baixas(considerando o sentido) para o posterior ganho de velocidade. A partir de uma certa posição à direita, faz sentido se esforçar indo mais à direita para alcançar o objetivo, mesmo que chegue com baixa velocidade no topo. Ao final, foi atingida uma média de 39,02 para a recompensa, o que sugere a eficácia do DQN.