Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT-213
Aluno: Vinícius José de Menezes Pereira

Relatório do Laboratório 12 - Deep Q-Learning

1. Breve Explicação em Alto Nível da Implementação

Neste laboratório, trabalhamos com Deep Q-Learning. A grande sacada do aprendizado de máquina por reforço profundo é que o aprendizado por reforço tipicamente utiliza uma tabela para guardar a função estado-ação ‘q’, o que pode ser melhor armazenado numa rede neural. A rede neural também tem uma capacidade de generalização maior que a tabela, de forma que estados parecidos tendem a ter uma ação ótima parecida.

Indo para a implementação em si, foram implementadas 3 funções:

def `make_model(self)`: função que retorna a rede neural profunda pedida segundo a Tabela 3 com o framework keras. A rede neural foi sequencial e teve 3 camadas densas. A função custo utilizada para compilar a rede foi *Mean Squared Error* - MSE e o otimizador foi o Adam com taxa de aprendizado fornecida. Os detalhes das camadas podem ser vistos na seguinte tabela:

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

def `act(self,state):` função que, dado um estado, calcula a função valor ‘q’ dada pelas saídas da rede neural implementada e retorna a ação e-greedy. Para isso, há uma probabilidade ‘e’ de ser escolhida uma ação aleatória e ‘1-e’ de escolher a melhor ação analisando as saídas da rede neural.

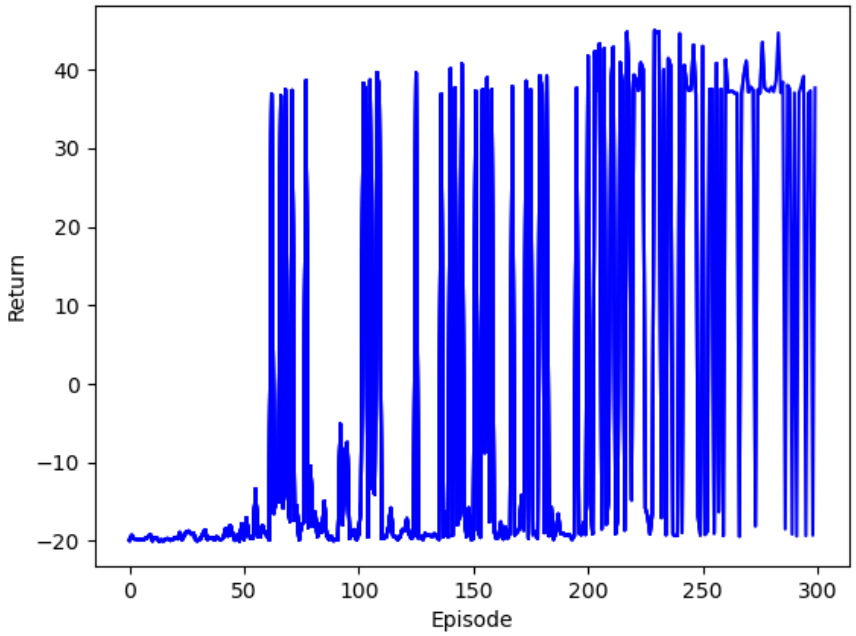
def `reward_engineering_mountain_car(state, action, reward, next_state, done):` função heurística que melhora a recompensa para deixar o treinamento mais rápido. Para isso, dá recompensa extra caso o carro ganhe velocidade, saia da posição inicial ou chegue no objetivo. Essa heurística é necessária, pois caso contrário, o treinamento ia perder muito tempo com velocidade nula e parado sem ganhar recompensa. Essa estado intermediário, sem a heurística, é um tanto nebuloso, pois não tínhamos como recompensar a máquina caso ela não chegasse na posição desejada.

2. Figuras Comprovando Funcionamento do Código

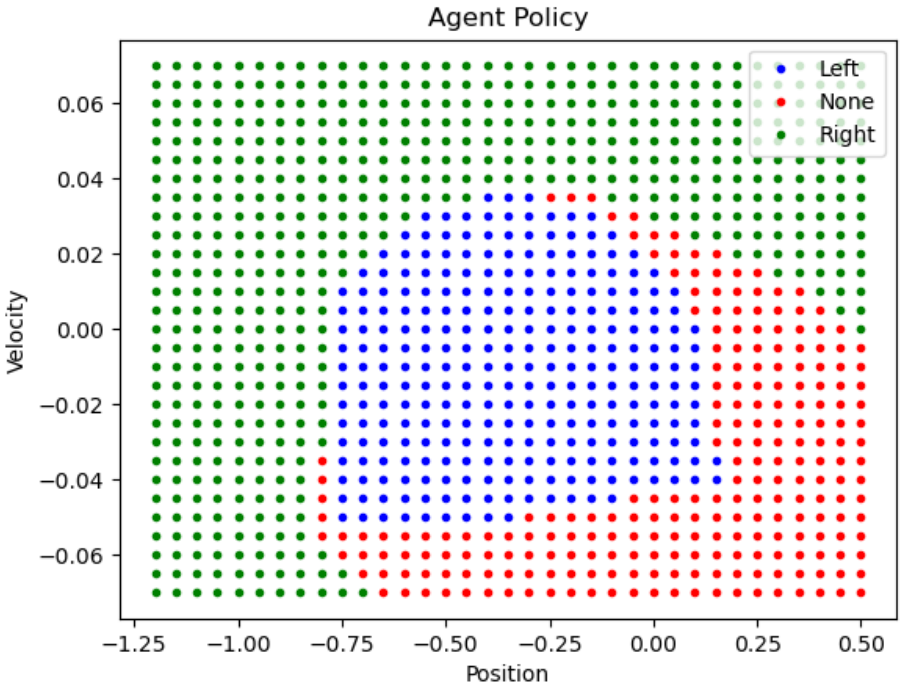
2.1. Sumário do Modelo

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 24)                72
_________________________________________________________________
dense_1 (Dense)              (None, 24)                600
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 75
=================================================================
Total params: 747
Trainable params: 747
```

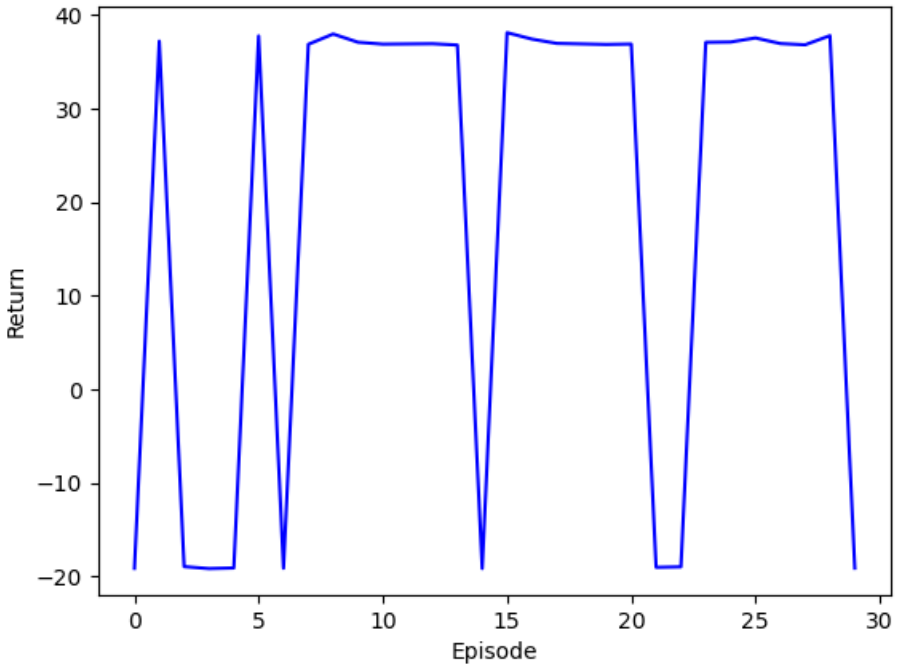
2.2. Retorno ao Longo dos Episódios de Treinamento



2.3. Política Aprendida pelo DQN



2.4. Retorno de 30 Episódios Usando a Rede Neural Treinada



3. Discussão dos Resultados

Vemos um ótimo resultado do aprendizado, que conseguiu chegar ao objetivo 22/30 vezes. Vemos que errar muito no começo faz com que ele aprenda como errar é ruim. Isso faz com que, tanto no treino da rede neural, quanto no Mountain Car utilizado para avaliar o treino haja aumento da reward ao longo do tempo. Percebemos que a política ótima é empurrar para traz no início, para dar ‘embalo’ no carro, empurrar para a direita quando ele

estiver voltando da ladeira até o carro atingir o objetivo. Vemos que se a velocidade dele estiver baixa ou negativa, ou seja, voltando, na subida, vale a pena não fazer nada e voltar para dar um empurrão para a esquerda e o carro pegar embalo.