1 Introdução

O objetivo desse laboratório foi implementar (em *Python*), algorítmos de aprendizado de máquina com aproximação de modelo, revisando estudos anteriores sobre o uso da biblioteca *Keras* e implementando métodos de tomada de decisão *epsilon-greedy* e *rewrd-engineering*.

2 Código

```
def make_model(self):
    """
    Makes the action-value neural network model using Keras.

    :return: action-value neural network.
    :rtype: Keras' model.
    """
    # Todo: Uncomment the lines below
    model = models.Sequential()

    model.add(layer=layers.Dense(24, activation=activations.relu,
                input_shape=(self.state_size,)))

    model.add(layer=layers.Dense(24, activation=activations.relu))

    model.add(layer=layers.Dense(self.action_size, activation=activations.linear))
    # Todo: implement Keras' model
    model.compile(loss=losses.mse,
                  optimizer=optimizers.Adam(lr=self.learning_rate))
    model.summary()
    return model


def act(self, state):
    """
    Chooses an action using an epsilon-greedy policy.

    :param state: current state.
    :type state: NumPy array with dimension (1, 2).
    :return: chosen action.
    :rtype: int.
    """
    p=np.random.random()

    if p<self.epsilon:
        return np.random.randint(self.action_size)
    else:
        return np.argmax(self.model.predict(state))

def reward_engineering_mountain_car(state, action, reward, next_state, done):
    """
    Makes reward engineering to allow faster training in the Mountain Car environment.

    :param state: state.
    :type state: NumPy array with dimension (1, 2).
    :param action: action.
```

```
        :type action: int.
        :param reward: original reward.
        :type reward: float.
        :param next_state: next state.
        :type next_state: NumPy array with dimension (1, 2).
        :param done: if the simulation is over after this experience.
        :type done: bool.
        :return: modified reward for faster training.
        :rtype: float.
        """
        reward+=(state[0]-START_POSITION_CAR)**2+ state[1]**2
        dr=0
        if next_state[0]>=0.5:
            dr=1
        reward+=50*dr
        return reward
```

3 Resultados e discussão

Após a implementação da rede, foi necessário treiná-la. Os resultados do treino estão apresentados na figura 1.

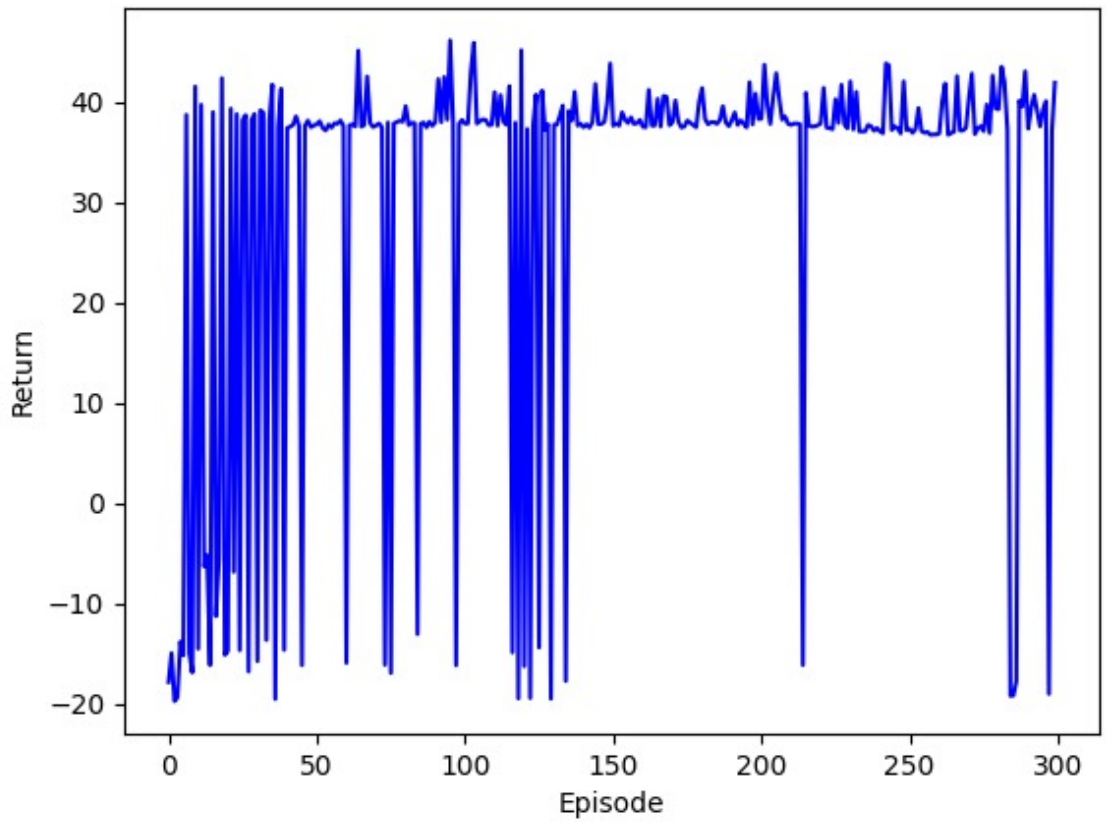

Figura 1: Gráfico de treinamento da rede neural após 300 episódios.

Observe na saída do programa, o carrinho consegue completar o percurso em 21 das 30 tentativas (70% de aproveitamento). Além disso, as figuras 2 e 3 mostram, respectivamente, as ações aprendidas pelo agente e a avaliação de cada episódio do carrinho.

```
episode: 1/30, time: 197, score: 37.2833, epsilon: 0.0
episode: 2/30, time: 200, score: -19.9915, epsilon: 0.0
episode: 3/30, time: 200, score: -19.9925, epsilon: 0.0
```

```
episode: 4/30, time: 200, score: -19.9788, epsilon: 0.0
episode: 5/30, time: 100, score: 42.4584, epsilon: 0.0
episode: 6/30, time: 153, score: 36.8084, epsilon: 0.0
episode: 7/30, time: 200, score: -14.4899, epsilon: 0.0
episode: 8/30, time: 155, score: 37.0417, epsilon: 0.0
episode: 9/30, time: 188, score: 36.9366, epsilon: 0.0
episode: 10/30, time: 153, score: 36.8113, epsilon: 0.0
episode: 11/30, time: 161, score: 45.4709, epsilon: 0.0
episode: 12/30, time: 192, score: 37.3783, epsilon: 0.0
episode: 13/30, time: 188, score: 36.9517, epsilon: 0.0
episode: 14/30, time: 200, score: -19.9886, epsilon: 0.0
episode: 15/30, time: 200, score: -19.9914, epsilon: 0.0
episode: 16/30, time: 155, score: 37.0435, epsilon: 0.0
episode: 17/30, time: 157, score: 37.2238, epsilon: 0.0
episode: 18/30, time: 152, score: 36.9774, epsilon: 0.0
episode: 19/30, time: 200, score: -19.9923, epsilon: 0.0
episode: 20/30, time: 200, score: -19.992, epsilon: 0.0
episode: 21/30, time: 158, score: 46.9528, epsilon: 0.0
episode: 22/30, time: 99, score: 42.396, epsilon: 0.0
episode: 23/30, time: 200, score: -19.99, epsilon: 0.0
episode: 24/30, time: 93, score: 41.3533, epsilon: 0.0
episode: 25/30, time: 197, score: 37.1065, epsilon: 0.0
episode: 26/30, time: 151, score: 37.0461, epsilon: 0.0
episode: 27/30, time: 190, score: 37.426, epsilon: 0.0
episode: 28/30, time: 108, score: 43.8607, epsilon: 0.0
episode: 29/30, time: 127, score: 45.6815, epsilon: 0.0
episode: 30/30, time: 161, score: 37.2804, epsilon: 0.0
Mean return:  21.769382124614154
```

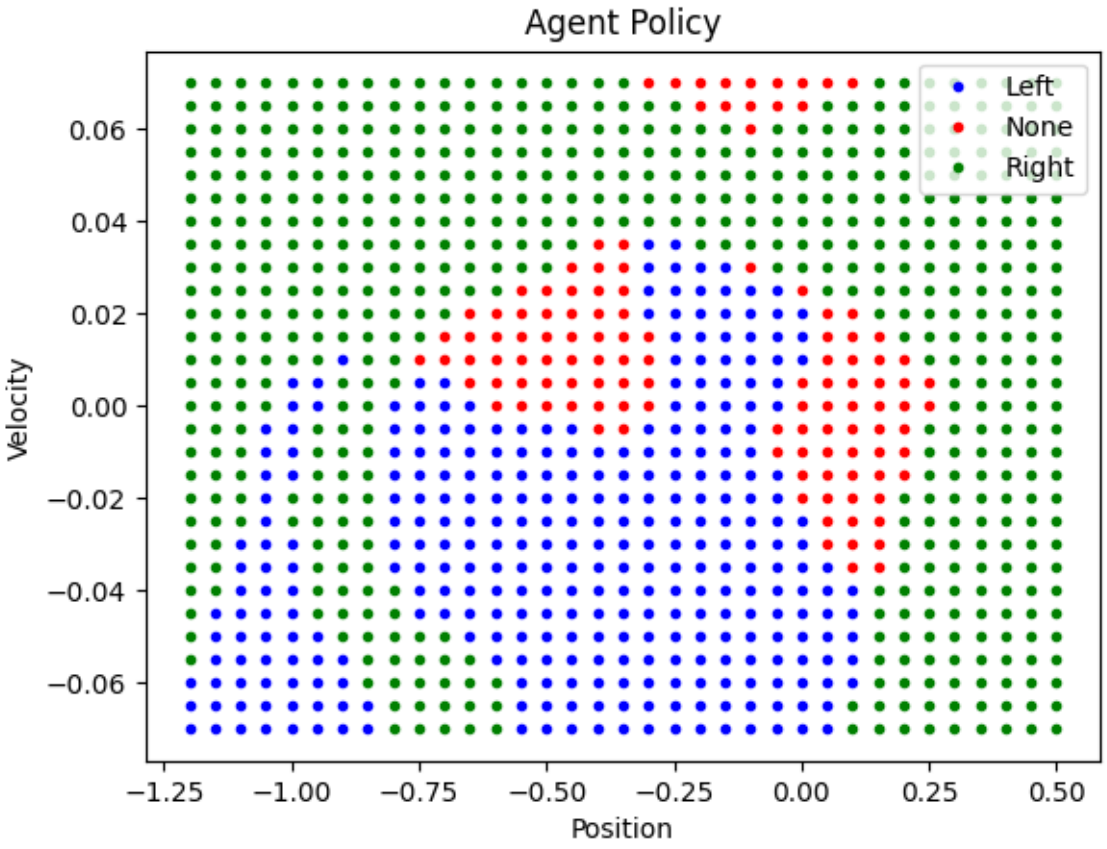

Figura 2: Decisão aprendida pelo agente em função da velocidade e posição.

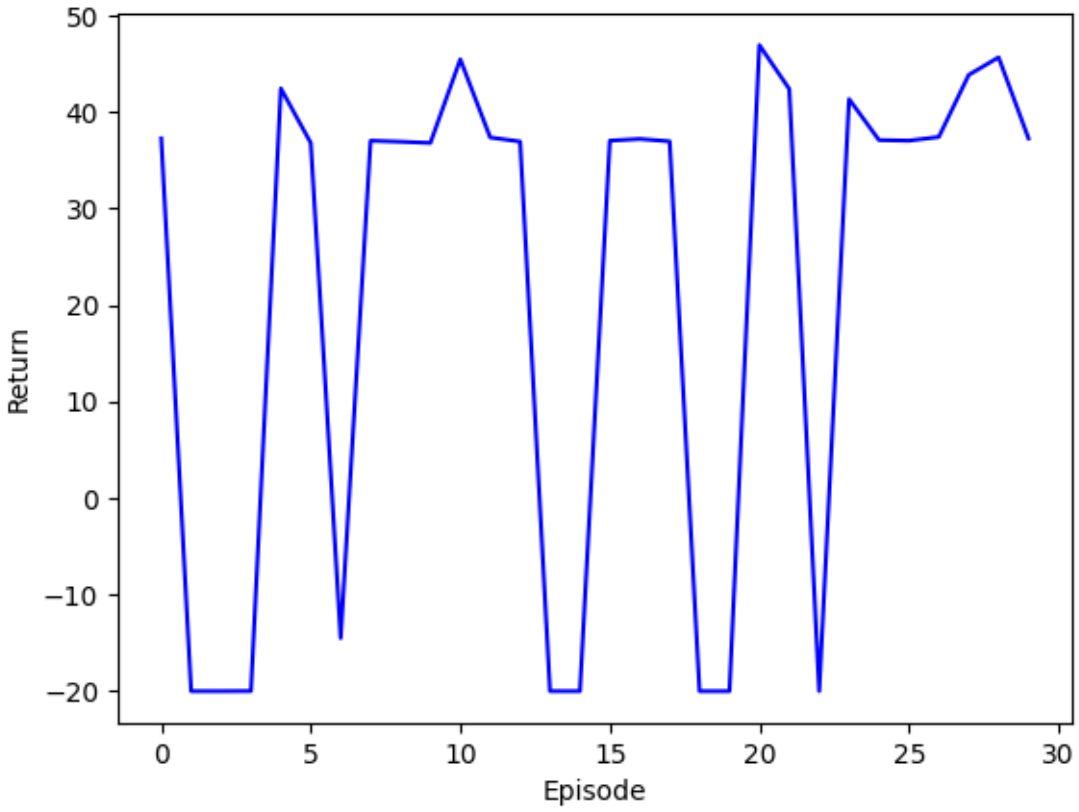

Figura 3: Resultado da implementação do *mountain_car*. Os picos positivos representam as situações em que o carrinho conseguiu completar o percurso, enquanto os picos negativos representam as situações em que o carrinho não conseguiu completar o percurso.