Relatório Cartpole

Jackson Hochscheidt[1], **João Barp**[1]

[1]Universidade Federal da Fronteira Sul (UFFS)
Caixa Postal 89815-899 – Chapecó – SC – Brasil

{jackson94h@gmail.com, barp.joao@gmail.com

Resumo. *Neste relátório será explicado como foi feito o trabalho de Inteligência Artificial para o problema do Cartpole*
Palavras-chave*: Cartpole.*

Abstract. *In this report will be explained how the work of Artificial Intelligence for the Cartpole problem was done*
Keywords*: Cartpole.*

1. Introdução

O problema Cartpole consiste em equilibrar um pino sobre um carrinho, esse carrinho pode se mover para esquerda ou direita, para equilibrar o pino. Para resolver este problema foi utilizado a aprendizagem por reforço, mais especificamente o Q-learning.

2. Aprendizagem por reforço

A aprendizagem por reforço é um problema computacional de aprendizagem em que um agente aprendiz procura maximizar uma medida de desempenho baseada em reforços, que são bonificações ou punições que recebe ao interagir com o ambiente.

O agente atua no ambiente formado por um conjunto de possíveis estados, e escolhe uma ação dentro de um conjunto de ações possíveis. Ele recebe um valor de reforço cada vez que executa uma ação, indicando o valor imediato da transição de estado resultante. Ao longo do tempo isso produz uma sequência de pares estado-ação e seus valores de reforço.

A tarefa do agente é aprender a melhor politica de controle, ou seja, sequência de ações que maximize a soma de reforços, descontando (usualmente de modo exponencial) as recompensas ou punições proporcionalmente ao seu atraso temporal [Monteiro and Ribeiro 2004].

2.1. Q-learning

O algoritmo Q-learning consiste na atualização de valores descontados de recompensas esperadas, Q(s,a). A cada iteração com o ambiente, os valores de Q são atualizados de acordo com a equação [Russell and Norvig 2016]:

Q[s,a] = Q[s,a] + alpha *(reward + gamma*(max(Q[s'])) - Q[s,a])

- alpha: é a taxa de aprendizagem.
- gamma: é o fator de desconto.
- reward: é a recompensa.

Após executar a ação a, o agente sai do estado *s* e vai para um estado *s'*, recebendo por esta ação uma recompensa imediata *reward*. No estado *s'* é feita uma busca, entre as ações disponíveis, para encontrar a ação *a'* que tenha o maior valor de retorno esperado.

[Faria and Romero 1999]

3. Observações

3.1. Dificuldades

Uma dificuldade encontrada durante o treinamento do algoritmo foi quanto a definição de alguns parâmetros, como alpha e gamma.

3.2. Discretização dos intervalos

A discretização do observation foi feita com a função cut, presente na biblioteca pandas. Cada intervalo é discretizado em um conjunto de intervalos.

Exemplo: Velocidade do Carro.

Intervalo: de -6 até 6.
Quantidade de intervalos(domínio): 10.
Resultado: [-4.8, -3.6, -2.4, -1.2, 0. , 1.2, 2.4, 3.6, 4.8, 6.]

3.3. Definição de recompensa

- recompensa: 1 para cada ação, e -10 quando o pino cai ou quando o carrinho passa das bordas laterais.

3.4. Plano de testes treinamento para definição de parâmetros

Será utilizada uma política de testes com três passos:

1. Será utilizado o mesmo valor de domínio(10) para todas as variáveis, exceto para a posição do carrinho que será 1.
   Serão efetuados três treinamentos para cada um dos seguintes valores para episódios[100,200,500,1000].
   - Para alpha ≈ 1 e gamma ≈ 0
   - Para alpha ≈ 1 e gamma ≈ 1
   - Para alpha ≈ 0 e gamma ≈ 0
   - Para alpha ≈ 0 e gamma ≈ 1
2. Comparar os valores para ver qual é melhor: Pegar os melhores valores para alpha e gamma, número de episódios... e treinar (executar o código) por 10 vezes, afim de tentar alcançar o objetivo

3.4.1. Definição dos parâmetros alpha, gamma e número de episódios

Após fazer os testes acima citados, verificou-se que os valores que tiveram melhores resultados foram: alpha = 0.7 e gamma = 0.3. Como pode-se observar nas figuras 1, 3, 2, 4. O que pode-se observar também é que o valor 100 é um bom parâmetro para a quantidade de episódios. Então, optou-se por usar 100 episódios para treinamento.

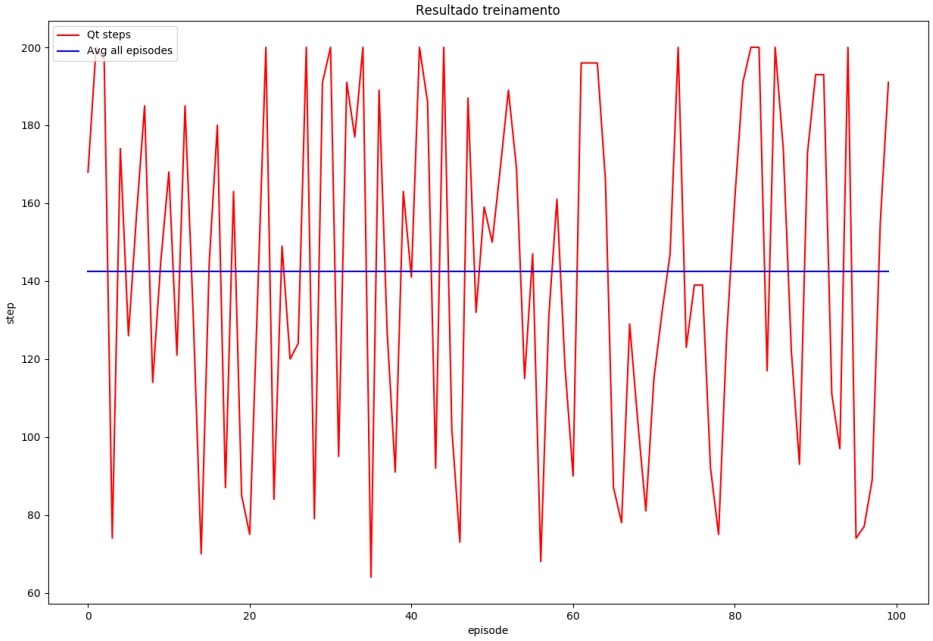

Figura 1. Treinamento com alpha = 0.2 e gamma=0.8

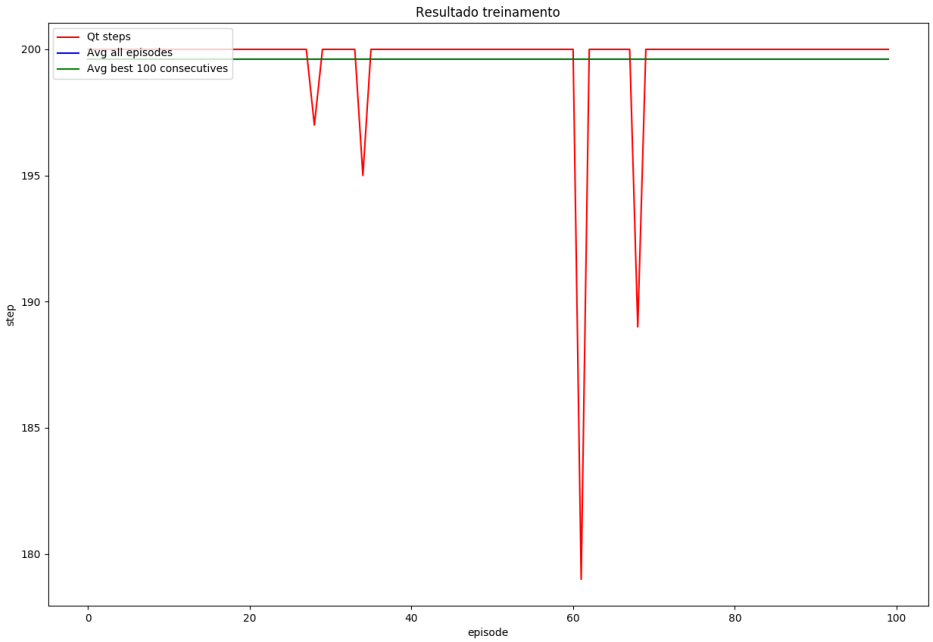

Figura 2. Treinamento com alpha = 0.7 e gamma=0.3

4. Resultados

Após realizar os testes para definição dos parâmetros alpha, gamma e quantidade de episódios para o treinamento, obteu-se o seguinte resultado, conforme as figuras: 5 que

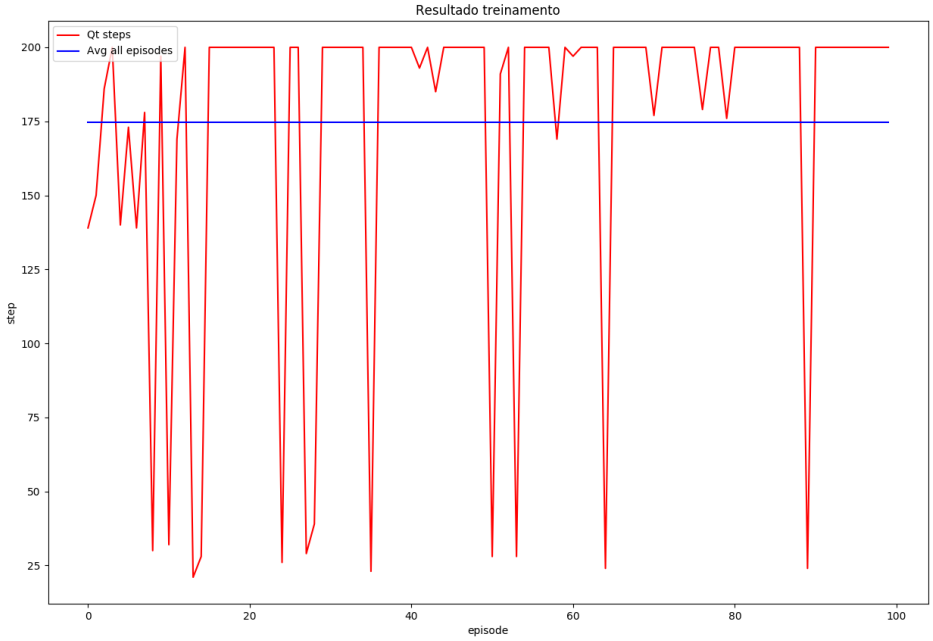

Figura 3. Treinamento com alpha = 0.4 e gamma=0.6

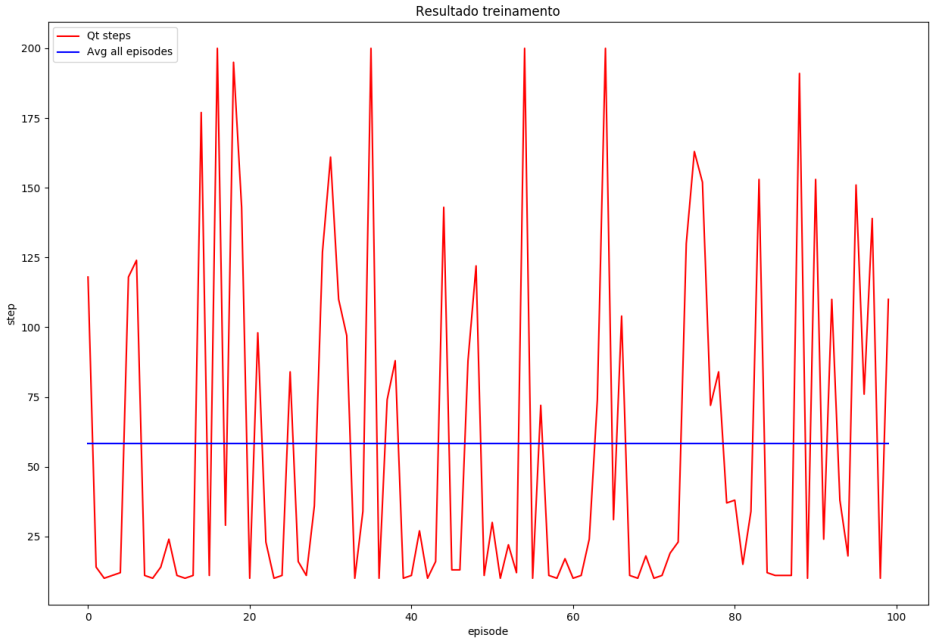

Figura 4. Treinamento com alpha = 0.8 e gamma=0.2

mostra o gráfico do treinamento com 100 episódios para alpha=0.7 e gamma=0.3, e 6 que

mostra o término da execução do treinamento, onde é os episódios e seus respectivos passos, bem como os valores de alpha e gamma, e também a média entre todos os episódios, o valor do melhor episódio e o desvio padrão.

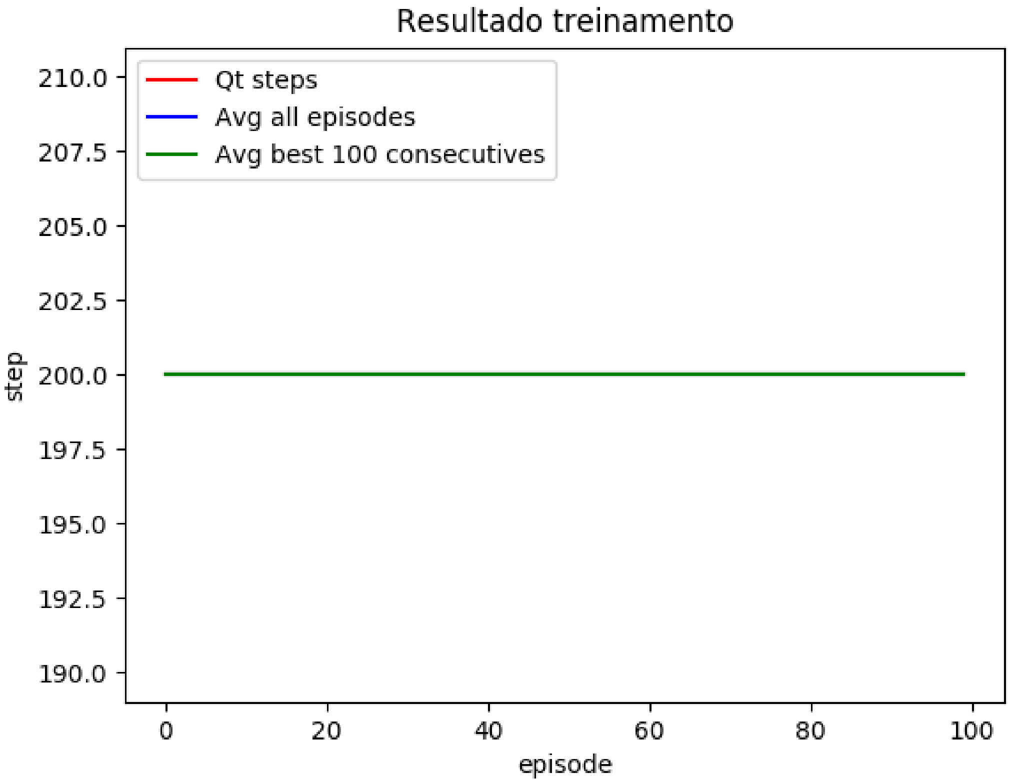

Figura 5. Após 10 treinamentos com alpha = 0.7 e gamma=0.3

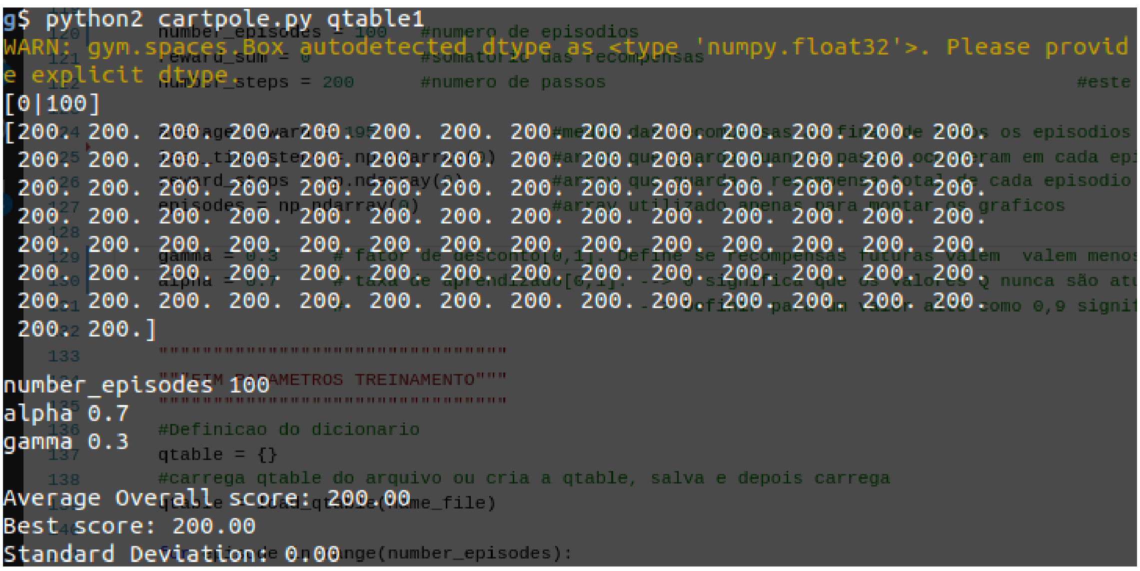

Figura 6. Print execução, após 10 treinamentos com alpha = 0.7 e gamma=0.3

5. Execução do trabalho

5.1. Compilação

Para compilar/executar o código use o comando:

- **python cartpole.py ”nomeArquivoQtable”** sem aspas.

5.2. Bibliotecas necessárias

As bibliotecas necessárias para a execução do trabalho são:

- numpy
- gym
- pandas
- json
- matplotlib

Para instalar alguma delas use o programa ”pip”, usando o comando: **pip install ”nome da biblioteca”** sem aspas.

Referências

Faria, G. and Romero, R. F. (1999). Explorando o potencial de algoritmos de aprendizado com reforço em robôs m^{TM}oveis. In *Anais do 4 Congresso Brasileiro de Redes Neurais*, pages 237–242, São Jos^{TM}e dos Campos, SP. CNRN.

Monteiro, S. T. and Ribeiro, C. H. (2004). Desempenho de algoritmos de aprendizagem por reforço sob condições de ambiguidade sensorial em robótica móvel. *Sba: Controle & Automação Sociedade Brasileira de Automatica*, 15(3):320–338.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.