



Instituto Infnet

Matemática para Data Science e Machine Learning

PROFESSOR | Adalberto Igor de Souza

TESTE DE PERFORMANCE

PABLO FELIPE



1. Explicar o que é análise de dados e como funciona o Machine Learning.

O *script* disponível no link [Classificador](#) apresenta o processo de aprendizado supervisionado de uma rede neural capaz de identificar números manuscritos. A partir desse código, responda às seguintes questões:

1.1. Explique a função de cada uma das camadas utilizadas no modelo apresentado.

Criar conjunto de dados → Parte onde carregamos um conjunto de dados para treinamento, no entanto, separamos uma parte da rede para avaliar o quão bom está esse treinamento.

Visualizando um elemento do dataset → Apenas plotando um valor para saber se temos algo no dataset.

Normalizar os dados → Neste formato temos os valores de 0 a 255, então nessa parte dividimos os dados por 255 no intuito de deixar o valor sempre entre 0 e 1.

Criar modelo de dados → Essa é a parte onde criamos o modelo, então, nessa parte criaremos as camadas. Entretanto, esse modelo é apresentado por três camadas a Flatten com entrada de uma imagem de 28X28(784), uma camada densamente conectada(128) com a função de ativação 'relu' e uma camada densamente conectada de saída (10).

Treinar o modelo de dados → Parte onde o modelo é treinado por 10 épocas.

Fazer o “evaluate” (accuracy) do Modelo → Essa parte avalia, através de cálculo, o quão bom foi o treinamento, medindo a acurácia, nele existe a função de perda/erro que avalia o que eu queria ter e o que o treino entregou.

Realizar inferências → Através das saídas, ele avalia qual a melhor saída de acordo com o dado passado.



Definindo a probabilidade de saída do modelo → Cria uma camada Softmax a fins de converter em probabilidade os valores de saída

Carregar a nova rede com modelo treinado

Salva o modelo

Criando um novo classificador a partir do modelo salvo

Realizando inferência com o modelo salvo

1.2. Explique a etapa de tratamento inicial dos dados utilizados durante o processo de treinamento.


Resposta já explicado no item 1.1

1.3. O que seria necessário para que esse modelo fosse capaz de reconhecer caracteres alfabéticos ao invés de números?

1.4. A partir do código apresentado, faça uma modificação que permita que sejam utilizadas imagens externas de números manuscritos. Mostre que seu código é capaz de realizar a tarefa, incluindo, além do trecho de código, uma imagem do predição. Dica: utilize a biblioteca python **cv2**.

▼ Importando as bibliotecas necessárias

```
✓ [18] import tensorflow as tf  
0s      import numpy as np  
      from tensorflow import keras  
      from keras.datasets import mnist as mn  
      import matplotlib.pyplot as plt  
      import cv2
```



▼ Carregando uma imagem de fora

```

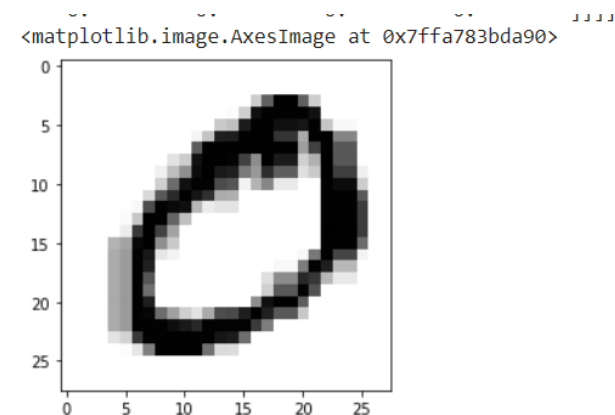
✓ 0s ▶ imgExt = cv2.imread('imagem.png',0)
imgExtResize = cv2.resize(imgExt,(28,28))
img_input = np.array(imgExtResize,np.float32)
img_input /= 255
img_input= np.expand_dims(img,0)
print(img_input)
# show image
plt.imshow(imgExtResize, cmap='gray')

```

```

[0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.14901961 0.44313725
 0.89803922 0.98823529 0.98823529 0.98823529 0.93333333 0.49019608
 0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.]
[0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.57647059 0.99215686
 0.99607843 0.99215686 0.95686275 0.93333333 0.99607843 0.99215686
 0.50196078 0.01568627 0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.]
[0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.77254902 0.98823529
 0.99215686 0.83921569 0.31764706 0.14901961 0.6  0.98823529
 0.98823529 0.50196078 0.  0.  0.  0.
 0.  0.  0.  0.  0.  0.]
[0.  0.  0.  0.  0.  0.
 0.  0.  0.  0.  0.31764706 0.65882353
 0.41568627 0.0745098 0.  0.  0.02745098 0.58431373
 0.98823529 0.98823529 0.14901961 0.  0.  0.
 0.  0.  0.  0.  0.  0.]

```



▼ Realizando inferência com o modelo salvo

2. Aplicar fundamentos de programação em Data Science.

Considere as seguintes matrizes:

$$A = \begin{bmatrix} -7 & 1 & -6 & -4 \\ -9 & -9 & 3 & -8 \\ 0 & 7 & 2 & 4 \\ 5 & -1 & 5 & 9 \end{bmatrix} \quad B = \begin{bmatrix} 7 & -4 & 2 & -10 \\ 6 & -2 & -7 & 8 \\ 3 & 0 & -2 & 0 \\ -4 & 9 & -5 & -2 \end{bmatrix} \quad C = \begin{bmatrix} 1 & -10 & 1 & -6 \\ -9 & -6 & -9 & 6 \\ 0 & -10 & -7 & -7 \\ 3 & -9 & -2 & 2 \end{bmatrix} \quad D = \begin{bmatrix} 7 & -9 & -6 & 9 \\ -5 & -4 & 3 & -7 \\ -5 & 7 & 2 & -1 \\ 2 & 1 & 9 & -5 \end{bmatrix}$$

Realize as seguintes operações:

2.1. A + B

$$\begin{bmatrix} -7+7=0 & 1+(-4)=-3 & -6+2=-4 & -4+(-10)=-14 \\ -9+6=-3 & -9+(-2)=-11 & 3+(-7)=-4 & -8+8=0 \\ 0+3=3 & 7+0=7 & 2+(-2)=0 & 4+0=4 \\ 5+(-4)=1 & -1+9=8 & 5+(-5)=0 & 9+(-2)=7 \end{bmatrix}$$

2.2. B * D

$$\begin{bmatrix} (7*7)+(-4*-5)+(2*-5)+(-10*2)=39 & (7*-9)+(-4*-4)+(2*7)+(-10*1)=-43 & (7*-6)+(-4*3)+(2*2)+(-10*9)=-140 & (7*9)+(-4*-7)+(2*-1)+(-10*-5)=139 \\ (6*7)+(-2*-5)+(-7*-5)+(8*2)=103 & (6*-9)+(-2*-4)+(-7*7)+(8*1)=-87 & (6*-6)+(-2*3)+(-7*2)+(8*9)=16 & (6*9)+(-2*-7)+(-7*-1)+(8*-5)=35 \\ (3*7)+(0*-5)+(-2*-5)+(0*2)=31 & (3*-9)+(0*-4)+(-2*7)+(0*1)=-41 & (3*-6)+(0*3)+(-2*2)+(0*9)=-22 & (3*9)+(0*-7)+(-2*-1)+(0*-5)=29 \\ (-4*7)+(9*-5)+(-5*-5)+(-2*2)=-52 & (-4*-9)+(9*-4)+(-5*7)+(-2*1)=-37 & (-4*-6)+(9*3)+(-5*2)+(-2*9)=23 & (-4*9)+(9*-7)+(-5*-1)+(-2*-5)=-84 \end{bmatrix}$$

2.3. C * A

$$\begin{bmatrix} (1 * -7) + (-10 * -9) + (1 * 0) + (-6 * 5) = \mathbf{53} & (1 * 1) + (-10 * -9) + (1 * 7) + (-6 * -1) = \mathbf{104} & (1 * -6) + (-10 * 3) + (1 * 2) + (-6 * 5) = \mathbf{-64} & (1 * -4) + (-10 * -8) + (1 * 4) + (-6 * 9) = \mathbf{26} \\ (-9 * -7) + (-6 * -9) + (-9 * 0) + (6 * 5) = \mathbf{147} & (-9 * 1) + (-6 * -9) + (-9 * 7) + (6 * -1) = \mathbf{-24} & (-9 * -6) + (-6 * 3) + (-9 * 2) + (6 * 5) = \mathbf{48} & (-9 * -4) + (-6 * -8) + (-9 * 4) + (6 * 9) = \mathbf{102} \\ (0 * -7) + (-10 * -9) + (-7 * 0) + (-7 * 5) = \mathbf{55} & (0 * 1) + (-10 * -9) + (-7 * 7) + (-7 * -1) = \mathbf{48} & (0 * -6) + (-10 * 3) + (-7 * 2) + (-7 * 5) = \mathbf{-79} & (0 * -4) + (-10 * -8) + (-7 * 4) + (-7 * 9) = \mathbf{-11} \\ (3 * -7) + (-9 * -9) + (-2 * 0) + (-2 * 5) = \mathbf{50} & (3 * 1) + (-9 * -9) + (-2 * 7) + (-2 * -1) = \mathbf{72} & (3 * -6) + (-9 * 3) + (-2 * 2) + (-2 * 5) = \mathbf{-59} & (3 * -4) + (-9 * -8) + (-2 * 4) + (-2 * 9) = \mathbf{34} \end{bmatrix}$$

2.4. Considere o sistema linear apresentado a seguir:

$$\begin{cases} -10x - 7y - 5z = -8 \\ 7y + 5z = 8 \\ 4x + 7y - 9z = -1 \end{cases}$$

Encontre os valores de **x**, **y** e **z**, utilizando a técnica de sistemas lineares apresentada em aula. Todos os cálculos devem ser apresentados. Caso julgue necessário, escreva um *script* utilizando a biblioteca *numpy* para conferir o resultado dos seus cálculos.

$$\begin{matrix} \mathbf{A} & & \mathbf{X} & & \mathbf{B} \\ \begin{bmatrix} -10 & -7 & -5 \\ 0 & 7 & 5 \\ 4 & 7 & -9 \end{bmatrix} & \times & \begin{bmatrix} x \\ y \\ z \end{bmatrix} & = & \begin{bmatrix} -8 \\ 8 \\ -1 \end{bmatrix} \end{matrix} \rightarrow \mathbf{X = A^{-1}B}$$

$$\begin{bmatrix} -10 & -7 & -5 \\ 0 & 7 & 5 \\ 4 & 7 & -9 \end{bmatrix} \times \begin{bmatrix} a & d & g \\ b & e & h \\ c & f & i \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} -10a + (-7b) + (-5c) = \mathbf{1} & -10d + (-7e) + (-5f) = \mathbf{0} & -10g + (-7h) + (-5i) = \mathbf{0} \\ 0a + (7b) + (5c) = \mathbf{0} & 0d + (7e) + (5f) = \mathbf{1} & 0g + (7h) + (5i) = \mathbf{0} \\ 4a + (7b) + (-9c) = \mathbf{0} & 4d + (7e) + (-9f) = \mathbf{0} & 4g + (7h) + (-9i) = \mathbf{1} \end{bmatrix}$$

$$a = \frac{7b+5c+1}{-10} \quad b = \frac{10a+5c+1}{7} \quad c = \frac{10a+7b+1}{5} \quad d = \frac{-7e+9f}{-4} \quad e = \frac{10d+5f}{7} \quad f = \frac{10d+7e}{5} \quad g = \frac{7h+5i}{-10} \quad h = \frac{-5i}{-7} \quad i = \frac{-7h}{-5}$$

3. Descrever dados quantitativamente, aplicando estatística.

3.1. O algoritmo de Aprendizado por Reforço conhecido por *Q-Learning* tem como base o processo de atualização da **Tabela Q** através da aproximação da Equação de Bellman, descrita por:

$$Q(s, a) = r_t + \gamma \max_a Q(s_{t+1}, a)$$

Explique cada um dos elementos desta equação.

$Q(s, a)$ → Estado e ação no tempo t

r_t → Recompensa no tempo t

γ → Fator de desconto

$\max_a Q(s_{t+1}, a)$ → Recompensa máxima do próximo estado, dada a ação máxima

3.2. Considere o problema *Cliff Walk*, mostrado na Figura 1. Apresente um conjunto de recompensas para os estados desse ambiente, sabendo que ele deve sair da posição inicial (*Start*) e chegar ao final (*Stop*) sem cair no penhasco (*Cliff*) e percorrendo o melhor caminho possível.

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1
Start -1	Cliff (-100)							Stop 100

3.3. Tendo como base o processo de treinamento apresentado em aula e a Equação de Bellman, apresentada anteriormente, construa a Tabela-Q para uma rodada de treinamento para o ambiente da Figura 1, apresentando os cálculos.

$$\gamma = 0,9$$

$$\text{usaremos a função abaixo: } Q(s_t, a_t) = r_t + \gamma \max_a Q(s_{t+1}, a)$$

Rodada 1:

$$Q(s_{[1,1]}, Up_0) = -1_{[1,1]} + 0,9 * \max_a Q(s_{[1,2]}, 0) = -1$$

$$Q(s_{[1,2]}, Up_0) = -1_{[1,2]} + 0,9 * \max_a Q(s_{[1,3]}, 0) = -1$$

$$Q(s_{[1,3]}, Up_0) = -1_{[1,3]} + 0,9 * \max_a Q(s_{[1,4]}, 0) = -1$$

$$Q(s_{[1,4]}, Right_0) = -1_{[1,4]} + 0,9 * \max_a Q(s_{[2,4]}, 0) = -1$$

$$Q(s_{[2,4]}, Right_0) = -1_{[2,4]} + 0,9 * \max_a Q(s_{[3,4]}, 0) = -1$$

$$Q(s_{[3,4]}, Right_0) = -1_{[3,4]} + 0,9 * \max_a Q(s_{[4,4]}, 0) = -1$$

$$Q(s_{[4,4]}, Right_0) = -1_{[4,4]} + 0,9 * \max_a Q(s_{[5,4]}, 0) = -1$$

$$Q(s_{[5,4]}, Right_0) = -1_{[5,4]} + 0,9 * \max_a Q(s_{[6,4]}, 0) = -1$$

$$Q(s_{[6,4]}, Right_0) = -1_{[6,4]} + 0,9 * \max_a Q(s_{[7,4]}, 0) = -1$$

$$Q(s_{[7,4]}, Right_0) = -1_{[7,4]} + 0,9 * \max_a Q(s_{[8,4]}, 0) = -1$$

$$Q(s_{[8,4]}, Right_0) = -1_{[8,4]} + 0,9 * \max_a Q(s_{[9,4]}, 0) = -1$$

$$Q(s_{[9,4]}, Down_0) = -1_{[9,4]} + 0,9 * \max_a Q(s_{[9,3]}, 0) = -1$$

$$Q(s_{[9,3]}, Down_0) = -1_{[9,3]} + 0,9 * \max_a Q(s_{[9,2]}, 0) = -1$$

$$Q(s_{[9,2]}, Down_0) = -1_{[9,2]} + 0,9 * \max_a Q(s_{[9,1]}, 0) = -1$$

$$Q(s_{[9,1]}, Terminal_0) = 100_{[9,1]} + 0,9 * \max_a Q(0,0) = 100$$



Tabela-Q

Estado	a=0 (Up)	a=1 (Right)	a=2 (Down)	a=3 (Left)
[1,1]	-1	0	0	0
[1,2]	-1	0	0	0
[1,3]	-1	0	0	0
[1,4]	0	-1	0	0
[2,1]	0	0	0	0
[2,2]	0	0	0	0
[2,3]	0	0	0	0
[2,4]	0	-1	0	0
[3,1]	0	0	0	0
[3,2]	0	0	0	0
[3,3]	0	0	0	0
[3,4]	0	-1	0	0
[4,1]	0	0	0	0
[4,2]	0	0	0	0
[4,3]	0	0	0	0
[4,4]	0	-1	0	0
[5,1]	0	0	0	0
[5,2]	0	0	0	0
[5,3]	0	0	0	0
[5,4]	0	-1	0	0
[6,1]	0	0	0	0
[6,2]	0	0	0	0
[6,3]	0	0	0	0
[6,4]	0	-1	0	0
[7,1]	0	0	0	0
[7,2]	0	0	0	0
[7,3]	0	0	0	0
[7,4]	0	-1	0	0
[8,1]	0	0	0	0
[8,2]	0	0	0	0
[8,3]	0	0	0	0
[8,4]	0	-1	0	0
[9,1]		100		
[9,2]	0	0	-1	0
[9,3]	0	0	-1	0
[9,4]	0	0	-1	0

3.4. A partir do exemplo disponível no link [Wearehouse](#) faça as devidas modificações para que seja possível que um agente aprenda a solucionar o problema *Cliff Walker*, mostrado na Figura 1, explicando as modificações propostas.

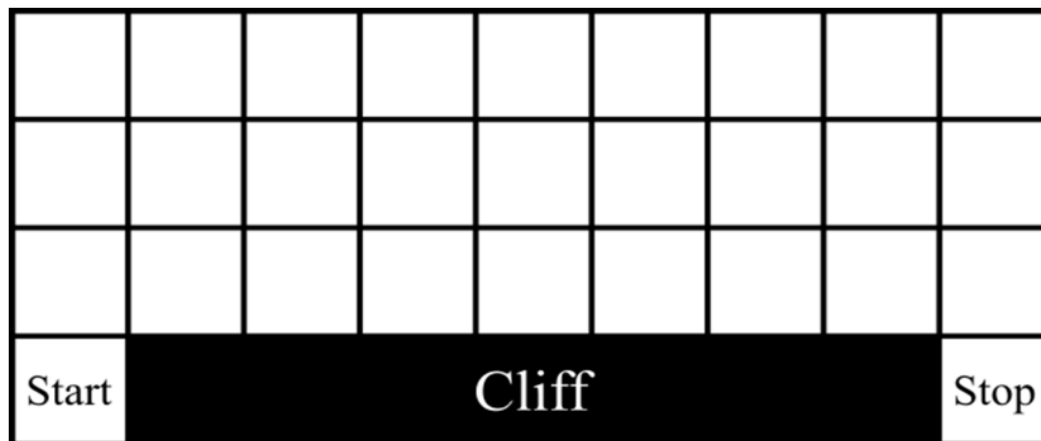


Figura 1 - Ambiente *Cliff Walk* para Aprendizado por Reforço.

Campos alterados:

```
# Definindo o ambiente (estados)
environment_rows = 4
environment_columns = 9
num_actions = 4
```

```
# Definindo as ações numericamente: 0 = up, 1 = right, 2 = down, 3 = left  
#actions = ['up', 'right', 'down', 'left']  
actions = ['right','down','up','left']
```

Minhas recompensas

```
# Criando a matriz 2D para armazenar o valor das recompensas de cada estado  
rewards = np.full((environment_rows, environment_columns), -1.)  
  
rewards[0,8] = 100. #set the reward for the packaging area (i.e., the goal) to 100  
  
rewards[0,1] = -100.  
rewards[0,2] = -100.  
rewards[0,3] = -100.  
rewards[0,4] = -100.  
rewards[0,5] = -100.  
rewards[0,6] = -100.  
rewards[0,7] = -100.
```

```
182 path = get_shortest_path(0,0)
183 print('Caminho até a área de entrega:\n',path)
```

Comentei essa parte

```
"""## Selecionando o menor caminho
Agora que o agente já foi completamente treinado, podemos verificar a qualidade do treinamento, verificando se o caminho su

# Testado para a posição sugerida anteriormente
print('Origem:\nLinha: 9 / Coluna: 10\n',get_shortest_path(9,10)) # origem: linha 9, coluna 10
print('Origem:\nLinha: 5 / Coluna: 0\n',get_shortest_path(5, 0)) # origem: linha 5, coluna 0
print('Origem:\nLinha: 9 / Coluna: 5\n',get_shortest_path(9, 5)) # origem: linha 9, coluna 5
print('Origem:\nLinha: 9 / Coluna: 4\n',get_shortest_path(9, 4)) # origem: linha 9, coluna 4

### Caminho reverso
Para que o robô possa fazer o caminho reverso, baste inserir a coordenada final e fazer a reversão do caminho encontrado
"""
```

```
### Caminho reverso
Para que o robô possa fazer o caminho reverso, baste inserir a coordenada final e fazer a reversão do caminho encontrado
"""

# Inserir a coordenada final (neste caso, coordenada [9,10])
path = get_shortest_path(0,0)

# Realiza a reversão do caminho recebido
#path.reverse()

print('Origem: área de entrega([0,0])\nDestino: [0,8]:\n',path)
```

4. Construir análises de dados simples.

Considere o exemplo [Jogo Catch the Ball](#), em que um agente é treinado para pegar uma bola.

4.1. Explique como os estados desse ambiente são representados.

Os estados desse ambiente são representados por uma pilha/sequência de 4 quadros(imagens).

4.2. Ao invés do método utilizado no algoritmo, como os estados desse ambiente poderiam ser representados e quais seriam eles?

Poderiam ser representados pela posição([x,y]) e a velocidade da bola e do paddle, no ambiente.

4.3. Utilizando o código original, realize o treinamento modificando um dos seguintes parâmetros de treinamento: tamanho do *batch*, fator de exploração *epsilon*, fator de desconto *gamma*. Compare com o resultado obtido com os valores originais do código, explicando o motivo para a variação no resultado, caso exista. **Dica: realize uma rodada de treinamento com os valores originais para que se tenha uma base de comparação com os novos valores.**

```
✓ 0s # Defindo os parâmetros do treinamento
NUM_ACTS = 3 # 0: Left; 1: Stay; 2: Right
BATCH_SIZE = 32 # Número de amostras para geração do dataset
NUM_EPOCHS = 1000
EPSILON = 0.3
GAMMA = 0.98 # Fator de desconto
MEMORY_SIZE = 5000000
MODEL_NAME = 'modelo_teste_8622.h5'
EPSILON_INIT = EPSILON
EPSILON_FINAL = 0.001
DEC_FACTOR = (EPSILON_INIT - EPSILON_FINAL) / NUM_EPOCHS
```

Recompensa acumulada = 22

```
✓ 0s ▶ # Defindo os parâmetros do treinamento
NUM_ACTS = 3 # 0: Left; 1: Stay; 2: Right
BATCH_SIZE = 32 # Número de amostras para geração do dataset
NUM_EPOCHS = 1000
EPSILON = 0.87
GAMMA = 0.98 # Fator de desconto
MEMORY_SIZE = 5000000
MODEL_NAME = 'modelo_teste_8622.h5'
EPSILON_INIT = EPSILON
EPSILON_FINAL = 0.001
DEC_FACTOR = (EPSILON_INIT - EPSILON_FINAL) / NUM_EPOCHS
```

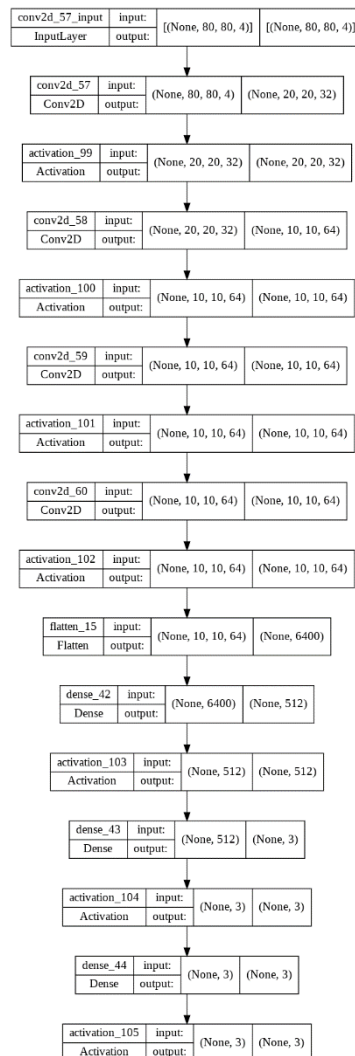
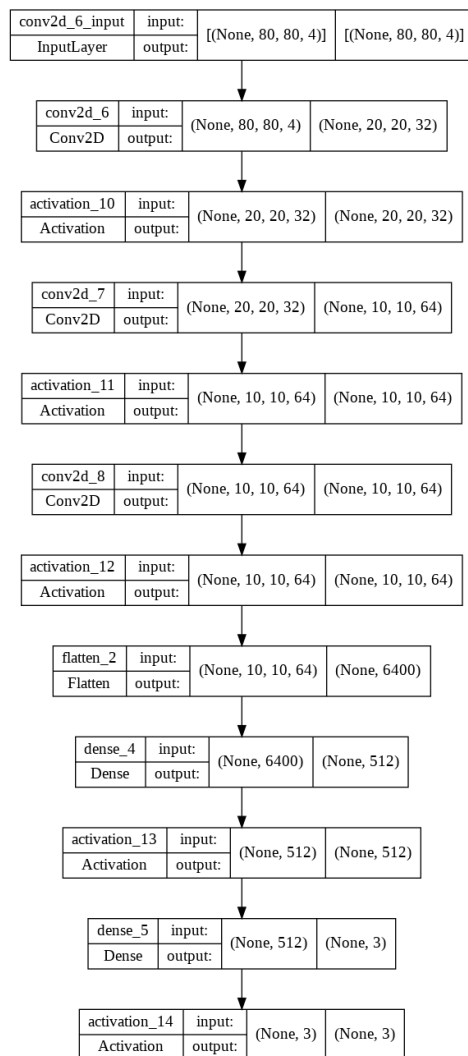
Recompensa acumulada = 23

O parâmetro alterado foi o Epsilon na intensão de uma exploração maior. Portanto, não houve impacto na mudança, com o epsilon 0.3 tivemos uma recompensa acumulada igual a 22, contra 23 com o épsilon a 0.87 (Os arquivos *.h5 estarão em anexo no ZIP).



4.4. Modifique o modelo neural, adicionando mais uma camada convolutiva (*Conv2D*) e uma camada densamente conectada (*Dense*) e apresente a imagem do seu novo modelo. Após essa modificação, realize o treinamento do agente e compare com o resultado gerados da base de comparação.

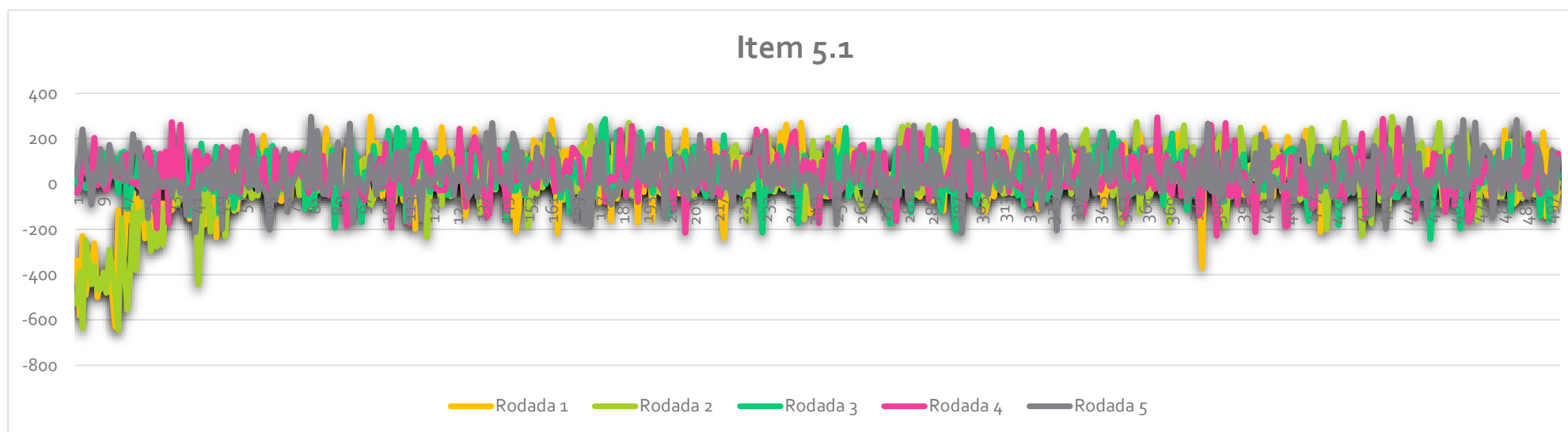
Os valores de recompensa encontrados foram igual 20, ou seja, bem próximo dos números citados no item 4.3, onde não apresentou muita diferença mesmo com mais uma camada convolutiva e mais uma cama densamente conectada, imagem abaixo.



5. Implementar análises de dados.

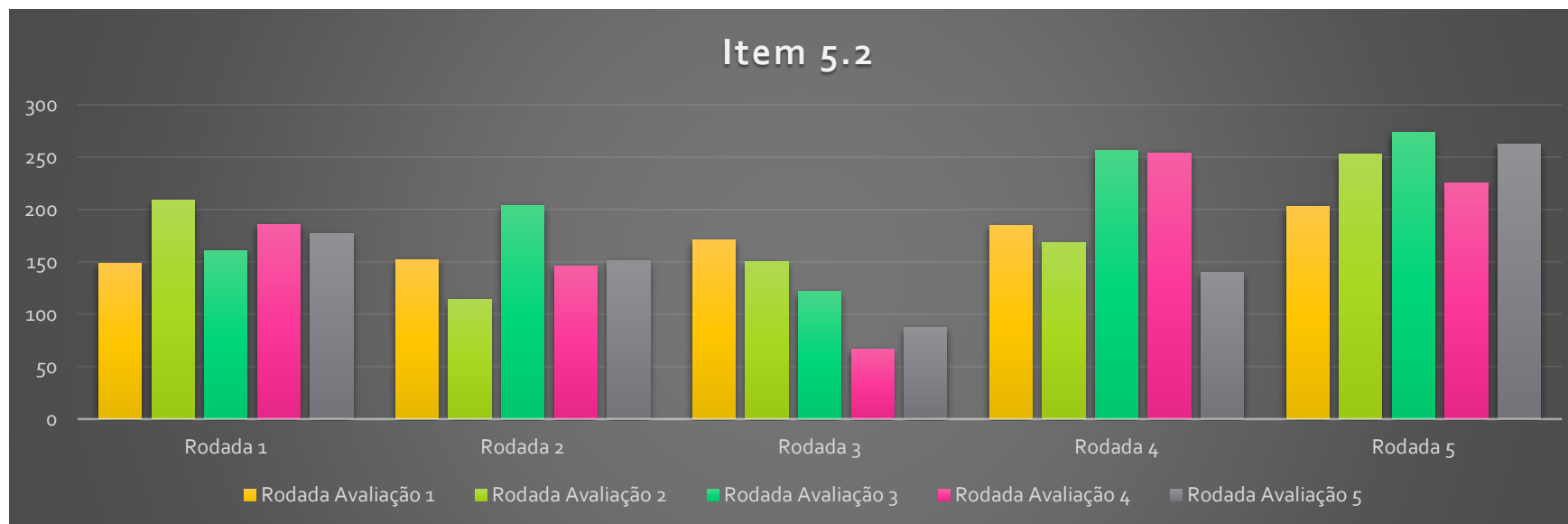
Considere o exemplo [Lunar Lander](#), em que um agente é treinado para pousar um rover lunar. A partir dele, realize as atividades a seguir.

5.1. Mantendo os parâmetros originais, realize cinco rodadas de treinamento, colete os dados gerados e apresente um gráfico comparando o resultado das cinco rodadas. Em seguida, analise o comportamento do aprendizado do agente.



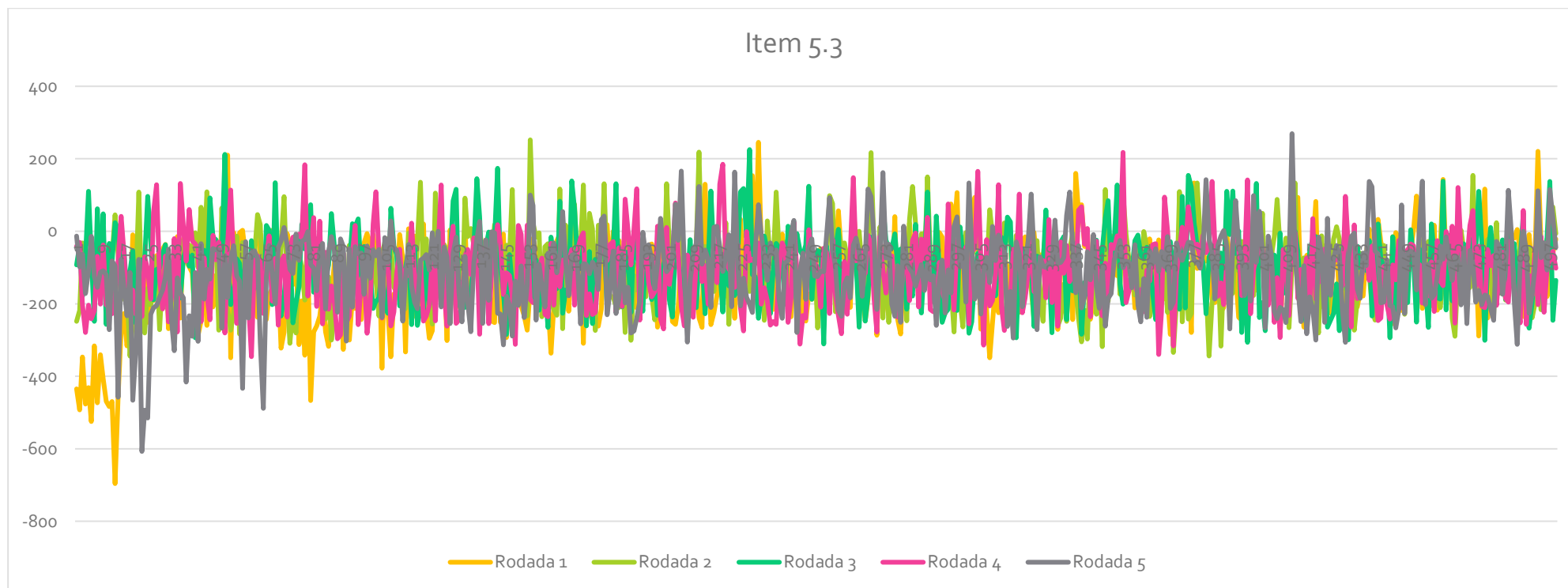
Como se pode ver no gráfico, as Rodadas 1 e 2 iniciam muito mal e depois mantem o treinamento tendo boas recompensas sem oscilar muito, já as Rodadas 3, 4 e 5 já iniciam tendo boas recompensas e assim vão até o final na mesma pegada.

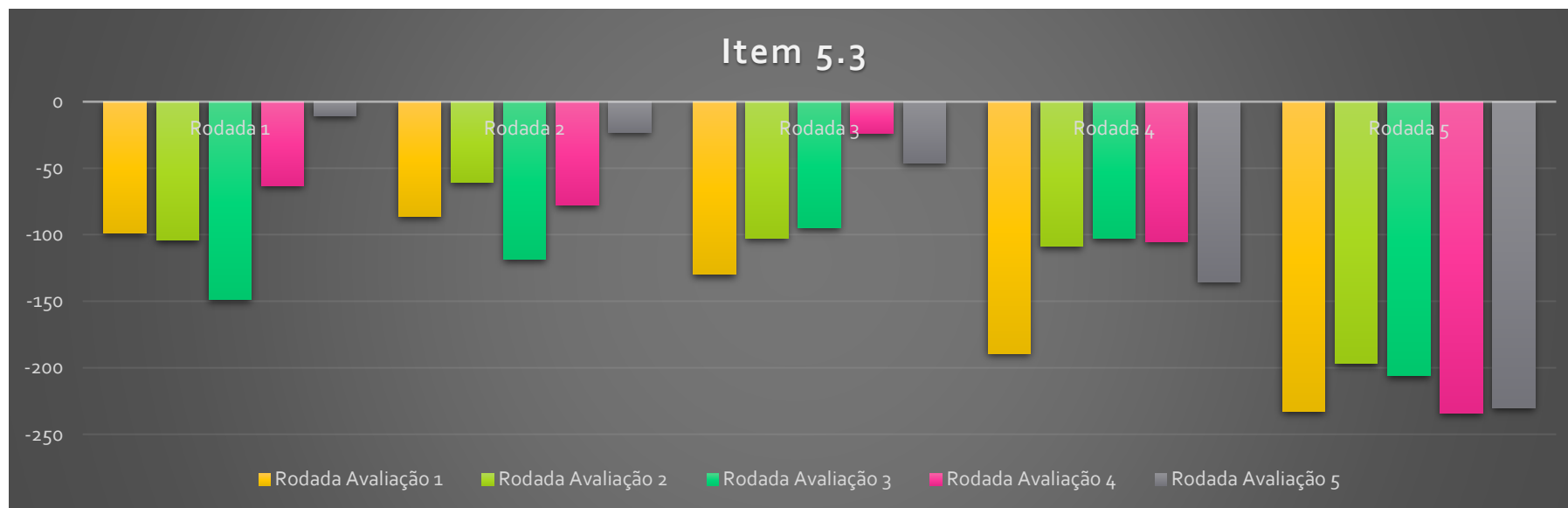
5.2. Para cada agente treinado, realize cinco rodadas de avaliação, mostrando a média dos resultados para cada um dos estados do ambiente (posição em x, y e orientação, velocidade em x, y e velocidade angular, posição da perna direita e esquerda). Avalie o resultado do treinamento, sabendo que o esperado é que o rover pouse com todos os valores de posição e velocidade zerados (ou muito próximos de zero) e com as duas pernas tocando o solo (valor 1).



Como se pode ver no gráfico acima, tivemos uma avaliação bem aquém, onde apenas na Rodada 3 > Avaliação 4 tivemos a média mais próxima de zero.

5.3. Modifique um dos parâmetros de treinamento (número de episódio de treinamento, fator de exploração *epsilon*, fator de desconto *gamma*, estrutura da rede neural) e realize os mesmos testes propostos em 5.1 e 5.2, fazendo uma comparação entre os resultados.





O parâmetro alterado foi o GAMMA de “0.98” para “0.2”. Nas rodadas de treinamento, ele se assimilou muito a questão 5.1, no entanto, as rodadas de avaliação foram todas negativas, onde o resultado mais próximo do zero foi na Rodada 1 > Avaliação 5.

GIT >> Arquivos complementares

<https://github.com/pablofelipecampelo/MatematicaDataScienceMachineLearning>