Laboratório 7 – *Imitation Learning* com Keras

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 copiar o movimento de um robô humanóide usando a técnica *Imitation Learning*. Para tal feito, foi implementada uma rede neural utilizando o *framework* de *Deep Learning* Keras e esta foi treinada a partir da observação do movimento de um robô que já funciona, isto é, usando *Imitation Learning*.

Metodologia e descrição em alto nível do algoritmo utilizado:

Para a construção da rede neural foi utilizada a *Framework* Keras do Python, que já apresenta funções para a construção da rede neural.

Assim, inicialmente foi testada esta *Framework* em um código já pronto, para analisar seu desempenho na classificação de funções sum_gt_zero() e xor(). Para isso, criou-se uma rede neural com uma camada duas camadas (uma de saída), tal que a primeira tivesse 50 neurônios e a segunda apenas 1, ambas com função de ativação sigmóide. Foi realizado o aprendizado com *Imitation Learning* usando e não usando regularização ($\lambda = 0,002$ ou $\lambda = 0$) para cada caso, o que forneceu um gráfico da convergência do custo nas épocas a partir de um *Dataset* aleatório.

Após isso, foi implementada e treinada a rede neural a ser usada para o problema em questão, para tal, utilizou-se os métodos model.add, model.compile e model.fit do Keras para facilitar a implementação da rede. A rede possui 3 camadas, sendo a primeira de 75 neurônios com função de ativação *Leaky RELU* ($\alpha = 0,01$), a segunda de 50 neurônios com

mesma função de ativação ($\alpha = 0,01$) e a terceira de 20 neurônios com função de ativação linear. Foram usadas 30.000 épocas de treinamento, todo o *dataset* em cada iteração do treinamento, erro quadrático como *loss function* e não foi usada regularização.

Com isso, foi possível realizar o treinamento da rede neural, obtendo gráficos dos ângulos de cada uma das juntas direitas do robô humanóide no tempo.

Resultados do test_keras.py:

Ao final do teste da rede neural, obteve-se os seguintes resultados:

Sem regularização ($\lambda = 0$):

Figura 1: Convergência da função custo utilizando sum_gt_zero sem regularização.

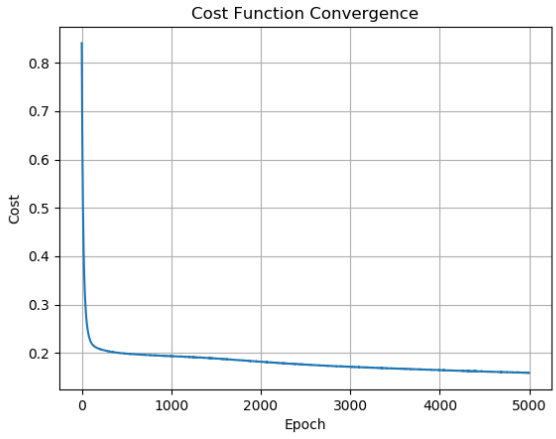


Figura 2: Convergência da função custo utilizando xor. sem regularização

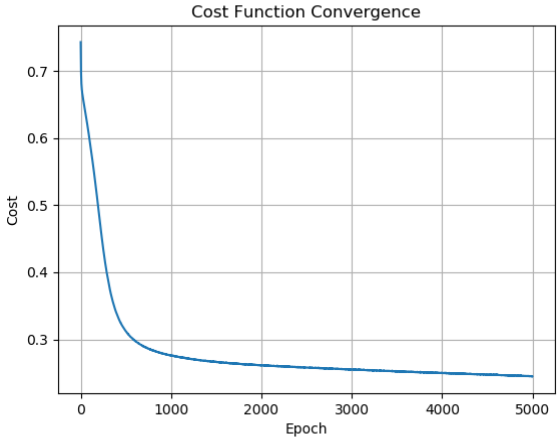


Figura 3: Dataset em sum_gt_zero sem regularização.

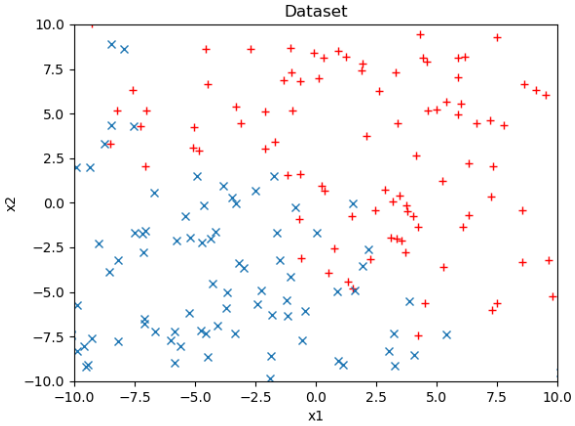


Figura 4: Dataset em xor sem regularização.

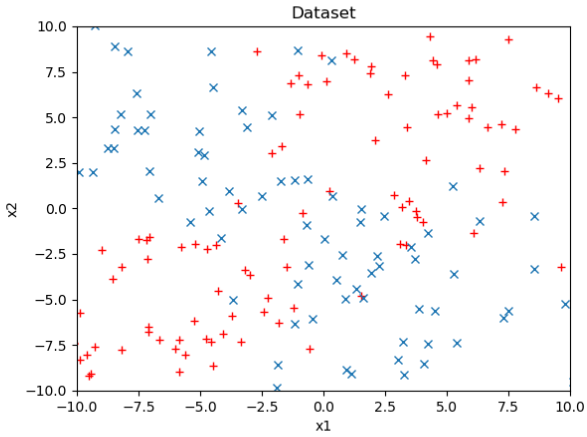
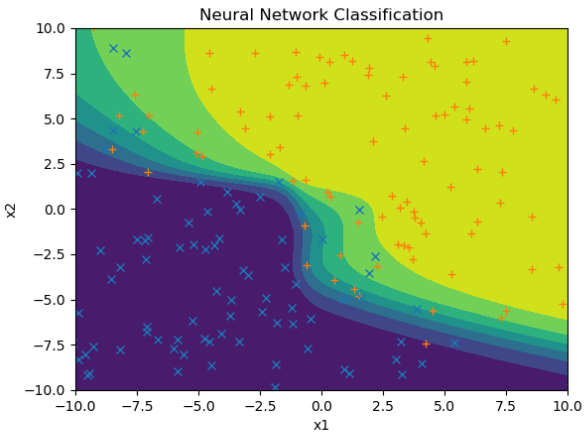
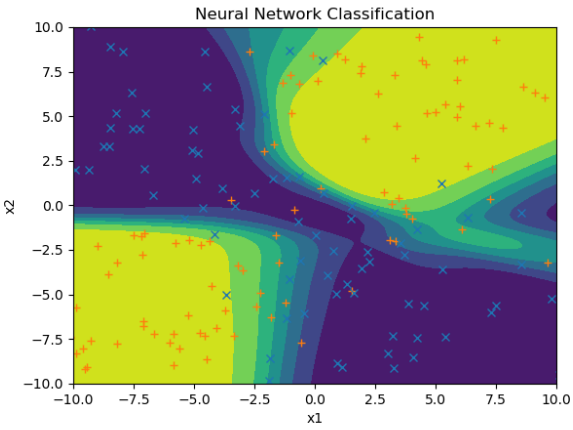


Figura 5: Classificação da função sum_gt_zero sem regularizar.

Figura 6: Classificação da função xor sem regularizar.

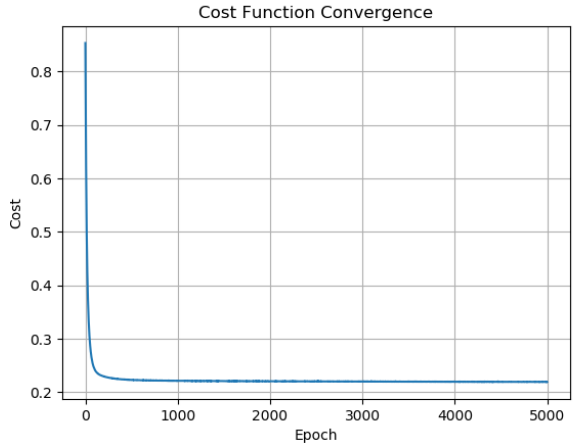




Com regularização ($\lambda = 0,002$):

Figura 7: Convergência da função custo utilizando sum_gt_zero com regularização.

Figura 8: Convergência da função custo utilizando xor. com regularização



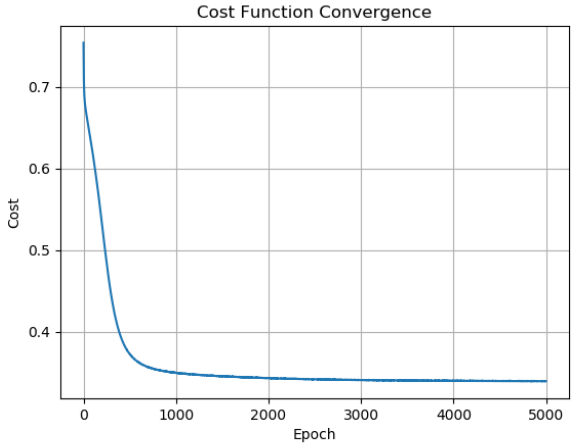
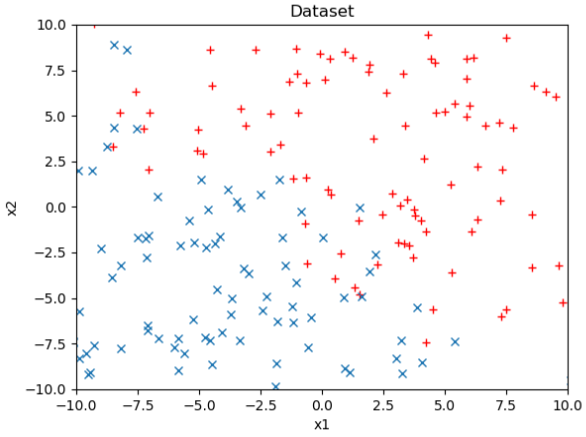


Figura 9: Dataset em sum_gt_zero com regularização.

Figura 10: Dataset em xor com regularização.



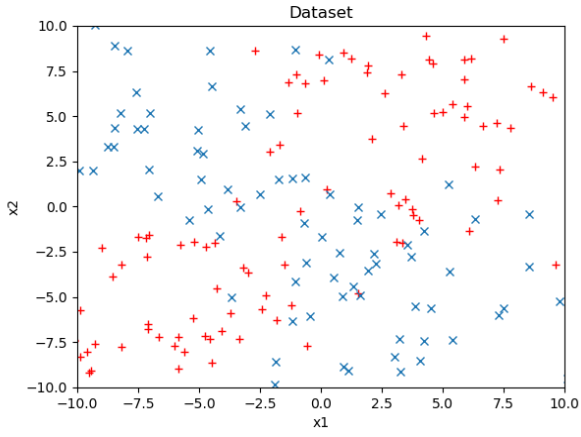
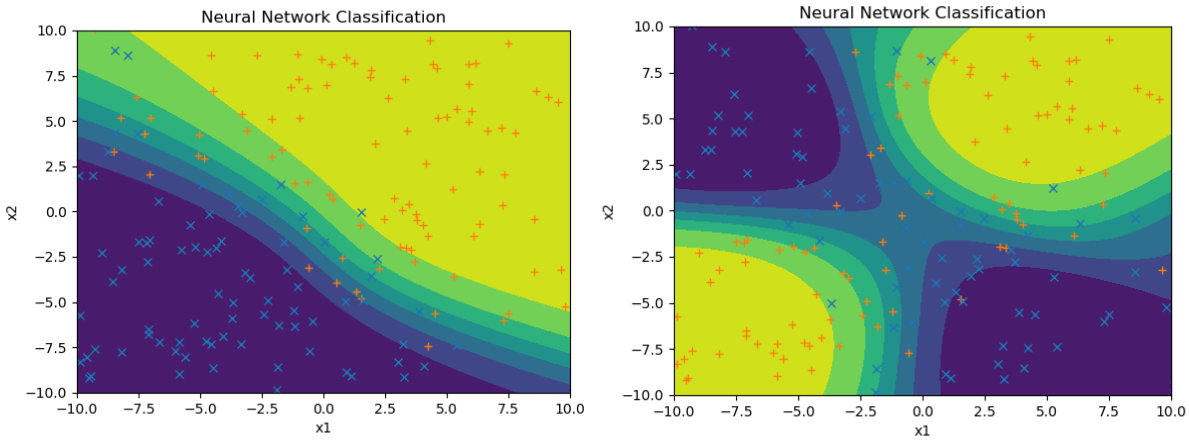


Figura 11: Classificação da função sum_gt_zero regularizando.

Figura 12: Classificação da função xor regularizando.



Analisando os resultados, é notório que, com regularização, os resultados foram muito mais satisfatórios, pois ela penaliza pesos muito grandes, dando assim, menos relevância aos pontos do *dataset* afastados do lugar esperado.

Resultados do imitation_learning.py:

Figura 13: Gráfico da angulação do rolo do quadril direito em função do tempo.

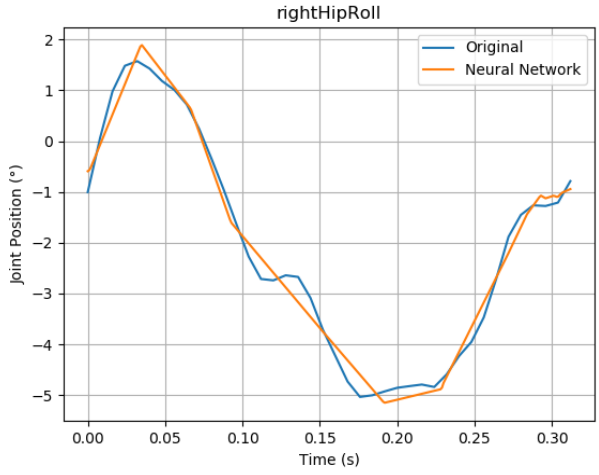


Figura 14: Gráfico da angulação do passo do quadril direito em função do tempo.

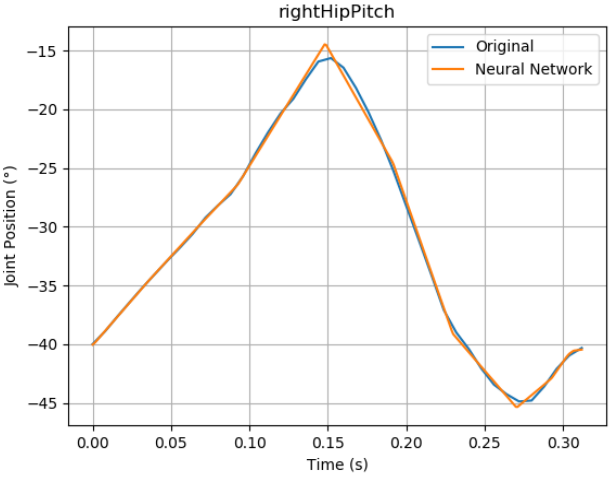


Figura 15: Gráfico da angulação do passo do joelho direito em função do tempo.

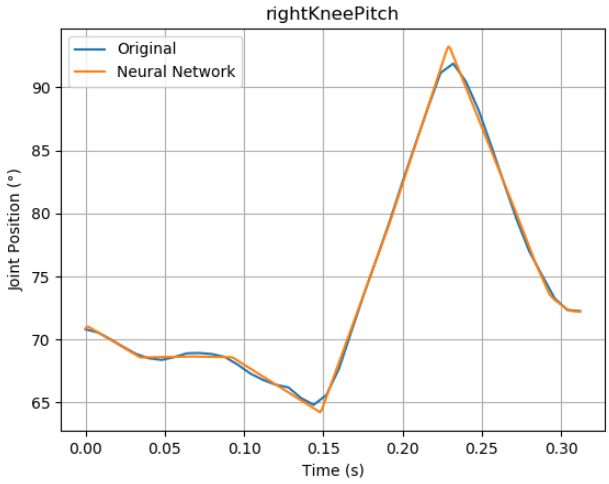


Figura 16: Gráfico da angulação do passo do tornozelo direito em função do tempo.

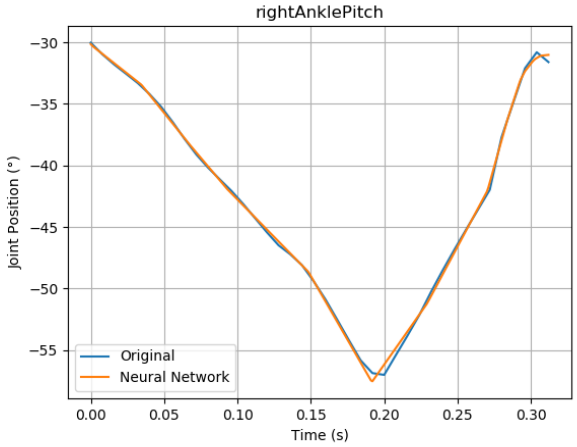
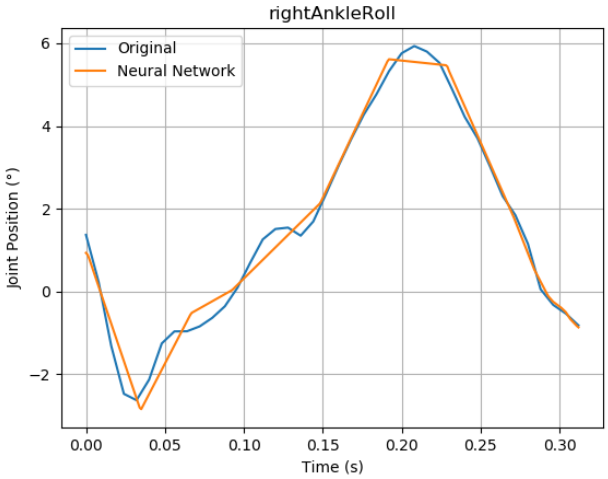


Figura 17: Gráfico da angulação do rolo do tornozelo direito em função do tempo.



Assim, com os resultados obtidos, é possível concluir que o *Imitation Learning* implementado obteve um desempenho satisfatório, dado que os desvios em relação aos pontos do *dataset* não foram tão significantes. Seria possível, ainda, melhorar ainda mais o algoritmo ao usar mais *"walking cycles"* do robô para o treinamento, ao invés de usar apenas o primeiro.