Laboratório 7 - Imitation Learning com Keras

Marcelo Buga Martins da Silva

CT-213 - Professor Marcos Ricardo Omena de Albuquerque Máximo

17/05/2021

1 Introdução

O propósito desse laboratório foi a familiarização com a *framework* de Deep Learning Keras, que utiliza do *TensorFlow* para a implementação dos algoritmos. Após o estudo de um caso de exemplo, foi possível utilizar a biblioteca para avaliar o efeito de Regularização em funções básicas de teste, assim como utilizar uma rede neural para copiar o comportamento de caminhada de um robô humanoide.

Para avaliar o sucesso das redes neurais analisadas, foram traçados gráficos para cada uma das situações de teste. Esses gráficos, assim como a discussão da implementação de cada seção são apresentados a seguir.

1.1 Análise do Efeito de Regularização

Para essa análise, o *script* base de teste do Keras fornecido foi utilizado, tendo sido testadas duas funções, *sum_gt_zero* que retorna 1 se a soma das coordenas de um ponto x é maior que 0 e 0 caso contrário e *xor* que retorna 1 se o sinal das coordenadas forem iguais e 0 se forem distintos. Testou-se, então, o efeito da regularização L_2, testando-se, para cada função, o resultado com e sem regularização. Sem regularização, $\lambda = 0$ e com regularização foi utilizado $\lambda = 0,002$. Em todos os casos foi inserido ruído para corromper o *dataset*. As

Figuras 1 e 2 mostram as regiões de classificação para a função *sum_gt_zero*, enquanto as Figuras 3 e 4 mostram as regiões de classificação da função *xor*.

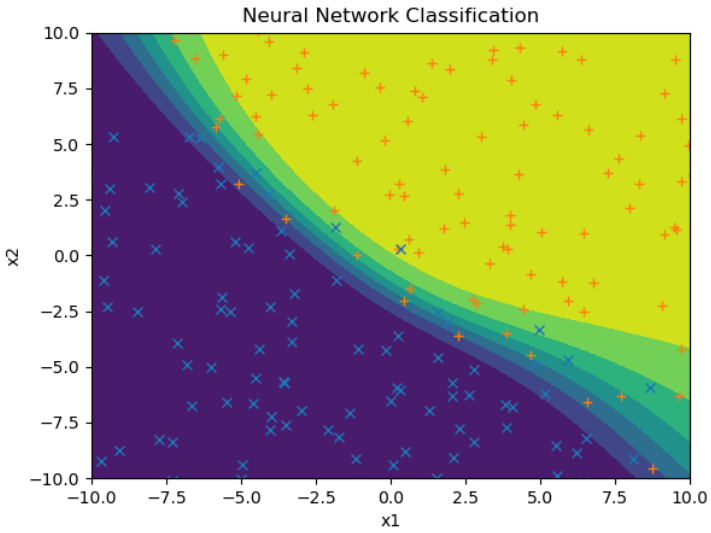

Figura 1: Regiões de classificação da função *sum_gt_zero* sem regularização

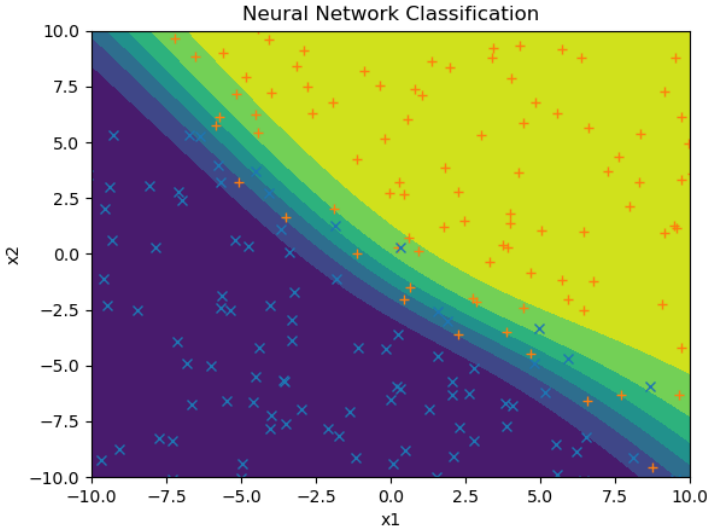

Figura 2: Regiões de classificação da função *sum_gt_zero* com regularização

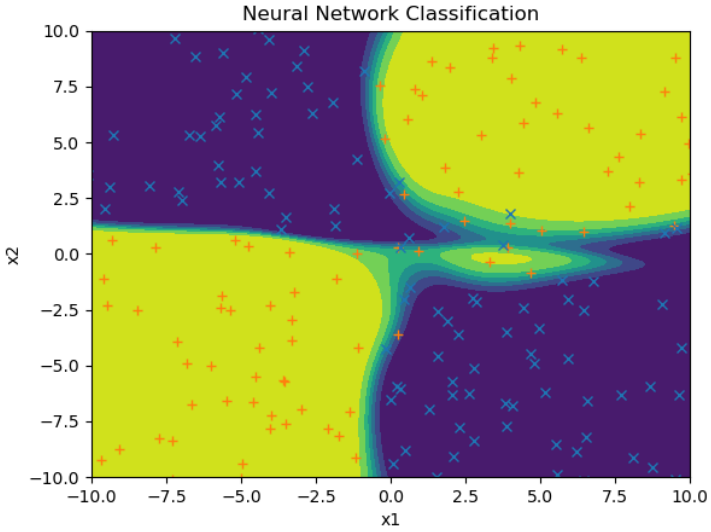

Figura 3: Regiões de classificação da função *xor* sem regularização

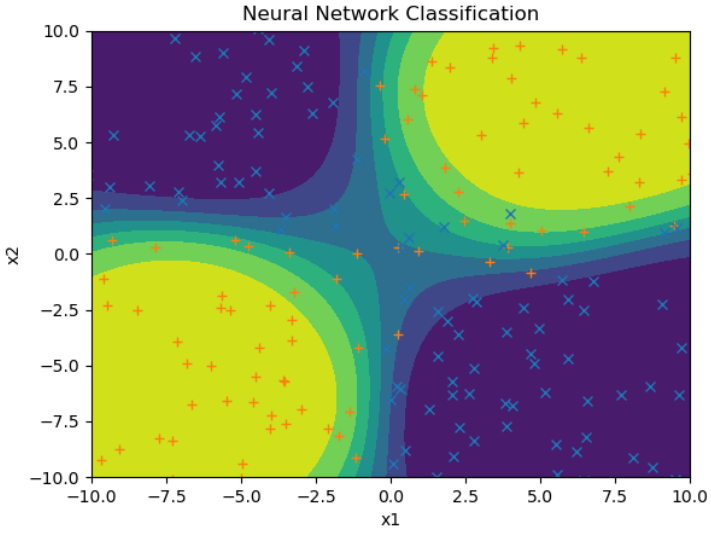

Figura 4: Regiões de classificação da função *xor* com regularização

Sendo o propósito da regularização evitar *overfitting*, os resultados, em geral, se provaram satisfatórios. Pode-se perceber que, no caso da função *sum_gt_zero*, a regularização teve efeito de tornar mais reta a curva que divide os espaços, como era de se esperar, dado que idealmente essa curva é a bissetriz dos quadrantes pares. Por sua vez, o efeito de diminuir o *overfitting* é mais claro para a segunda função, em que uma pequena região separada, provavelmente gerada pelo ruído, que aparecia na Figura 3 não apareceu mais após a regularização, sendo a Figura 4 mais próxima do esperado, em que os quadrantes são idealmente coloridos alternadamente.

Além do resultado classificatório, pode-se também avaliar a convergência da função de custo, em que foi utilizada a função *binary cross-entropy*, também conhecida como *log loss*. Os gráficos de convergência traçados para ambas as funções tiveram características similares. As Figuras 5 e 6 exemplificam esse comportamento:

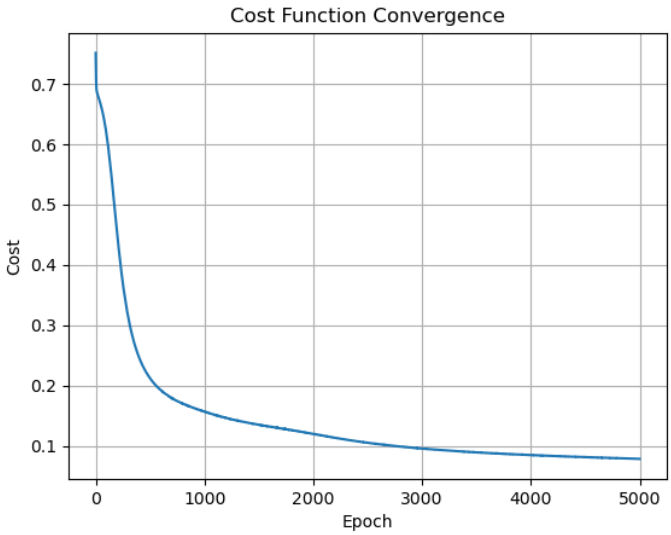

Figura 5: Convergência da função *xor* para $\lambda = 0$

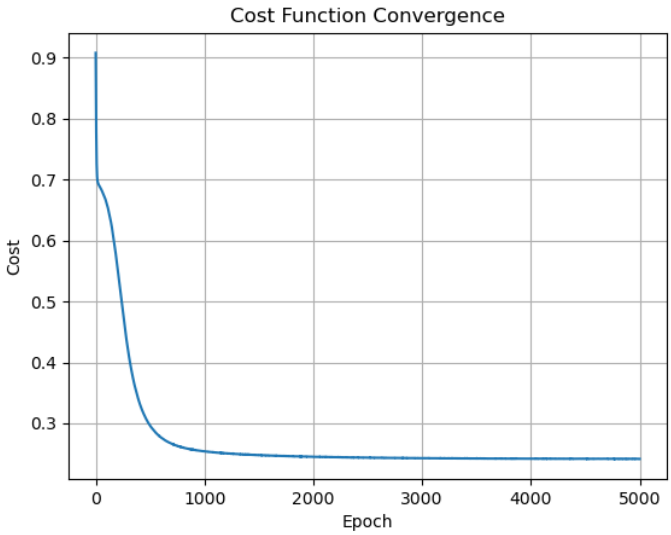

Figura 6: Convergência da função *xor* para $\lambda = 0,002$

O efeito mais claro da regularização para esses gráficos é o aumento do custo, sendo que ele não converge mais para zero. Isso é um efeito esperado, dado que foi somada à função de custo um termo que envolve o somatório dos valores dos parâmetros ao quadrado, o que faz o custo mínimo ser maior que zero.

1.2 Imitation Learning

Por fim, utilizou-se o Keras para montar uma rede neural com o propósito de aprender o movimento de caminhada de um robô humanoide. Para essa rede neural, foram utilizados os dados das posições de cada uma das vinte juntas do robô humanoide em cada instante de tempo para o treinamento da rede. A montagem da rede se deu com uma camada de entrada, duas *hidden layers*, sendo a primeira de 75 neurônios e a segunda de 50 neurônios, ambas com função de ativação *Leaky ReLU* ($\alpha = 0,01$), e uma camada de saída com 20 neurônios e função de ativação linear. O otimizador utilizado foi o Adam (1) e a função de custo escolhida foi a perda quadrática. Não foi utilizada regularização.

Montada a rede neural, ele foi então treinada com *batches* do tamanho do *dataset*, no caso, 40 instantes de tempo. Foram feitas 30000 iterações para o treinamento da rede. Os *outputs* esperados utilizados no treinamento foram as posições angulares de cada junta em cada um dos 40 *inputs*.

Após o treinamento, a rede foi utilizada para determinar a posição angular de cada instante de um *array* que ia de 0 a 0,312s, espaçado de 0,001s. As Figuras 7, 8, 9, 10 e 11 apresentam os resultados da comparação do gráfico original com o obtido pela rede neural para as juntas da perna direita do robô.

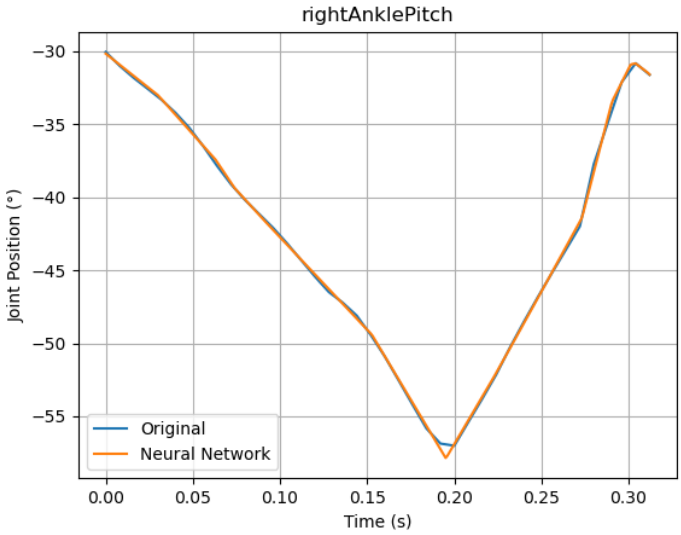

Figura 7: Junta *Right Ankle Pitch*

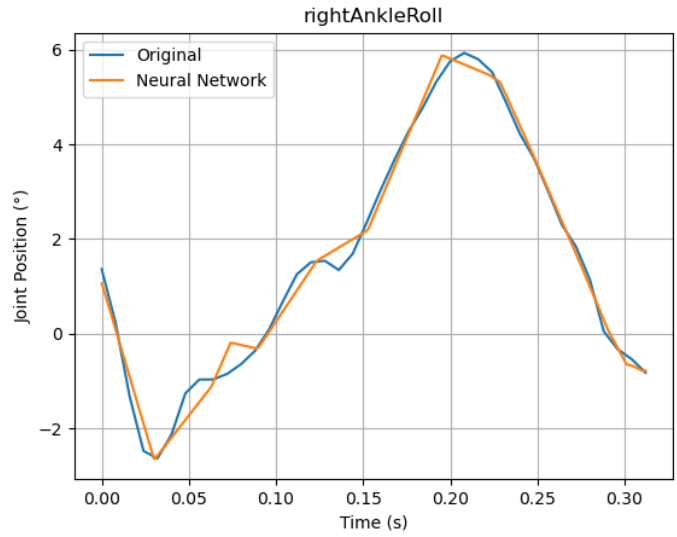

Figura 8: Junta *Right Ankle Roll*

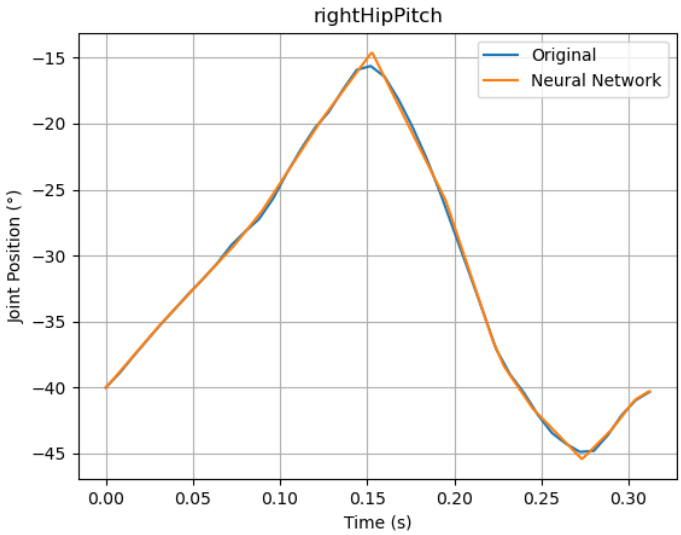

Figura 9: Junta *Right Hip Pitch*

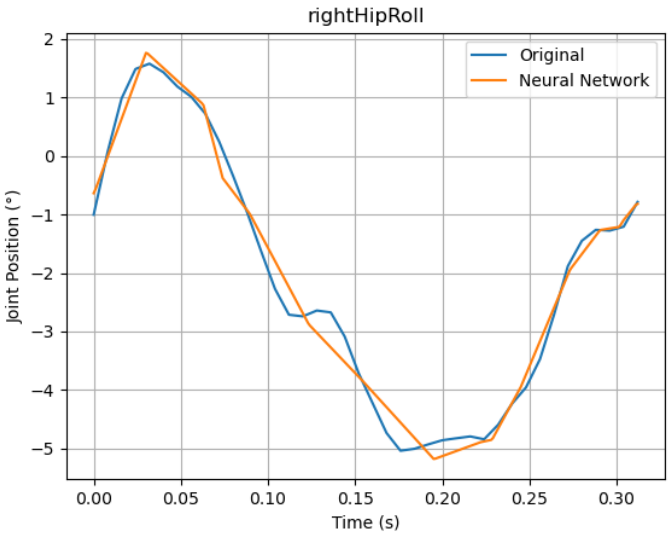

Figura 10: Junta *Right Hip Roll*

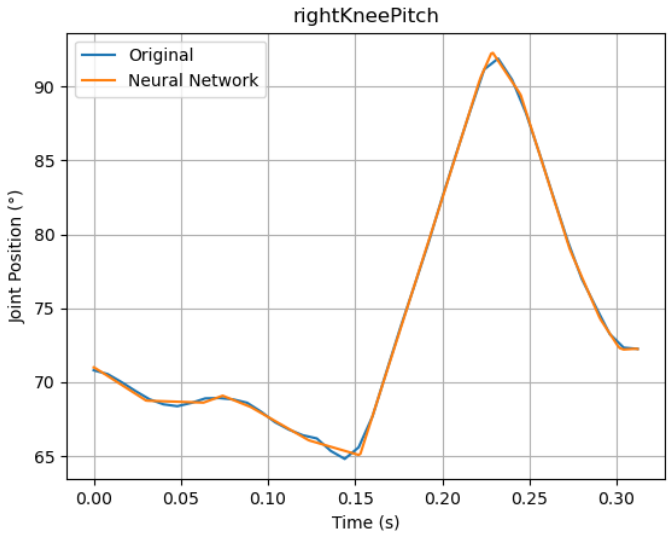

Figura 11: Junta *Right Knee Pitch*

Pode-se considerar que a rede neural conseguiu um excelente *fit* com os dados, tendo sido mais evidentes discrepâncias apenas nos gráficos de menores escalas. Para esses casos, normalmente as discrepâncias se deram pela rede neural ter assumido um comportamento mais linear do movimento. Além disso, os gráficos da rede neural tem mais "bicos", enquanto os originais são mais suaves. De qualquer forma, a imitação foi muito precisa e a rede neural se provou uma boa forma de conseguir imitar a caminhada de um robô humanoide.

Referências

1 KINGMA, Diederik P; BA, Jimmy. Adam: A method for stochastic optimization, 2014.