Instituto Tecnológico de Aeronáutica

CT-213: Inteligência Artificial para Robótica Móvel

Lab 7 - *Imitation Learning* com Keras

Bruno Benjamim Bertucci - Turma 23.2

1 Introdução

Com a popularização das redes neurais, junto com a evolução do poder de processamento dos computadores, surgiu a necessidade de criar técnicas para permitirem o treinamento de redes neurais profundas, ou seja, com muitas camadas e muitos neurônios. Essas técnicas vieram a constituir a área de *Deep Learning*. Uma delas, denominada regularização, tem o intuito de penalizar a criação de pesos muito grandes na rede. A implementação mais utilizada dessa técnica, denominada regularização L_2, inclui na função de custo o termo descrito na Equaçao 1, onde λ é um parâmetro.

$$J_{L_2}(\theta) = \frac{\lambda}{2m} \sum_j \theta_j^2 \tag{1}$$

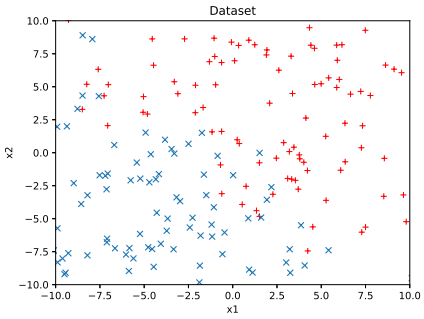
2 Teste do uso de Regularização

Para avaliar o efeito da regularização na rede neural, foi utilizada uma rede neural de duas camadas utilizando o *framework* Keras. A primeira camada é constituída de 50 neurônios, e a camada de output, de um neurônio. Ambas usaram o sigmoide como função de ativação.

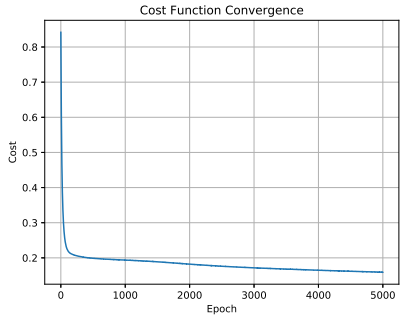
Essa rede foi usada para aproximar uma determinada função de classificação. O *dataset* foi criado pela geração de um *input* com 200 valores de entrada. Em seguida, um vetor de *outputs* esperados foi criado, armazenando os resultados da função de classificação para cada elemento do *input*. Finalmente, foi adicionado ruído ao *input*.

Ao todo, foram testadas duas funções de classificação. Uma delas é a *sum_gt_zero*, que recebe duas coordenadas e retorna 1 se a soma delas for maior que zero, e 0 se a soma for menor ou igual a zero. Ou seja, as regiões de cada classificação são divididas pela equação $y = -x$. A outra função testada é a *xor*, que recebe duas coordenadas de um ponto e classifica-o como 1 se ambas tiverem o mesmo sinal, e como 0 se tiverem sinais distintos (nesse caso, incluiu-se o zero junto com os números positivos). Ou seja, no plano cartesiano, a função *xor* classifica como 1 o primeiro e o terceiro quadrante, e como 0 o segundo e quarto quadrante.

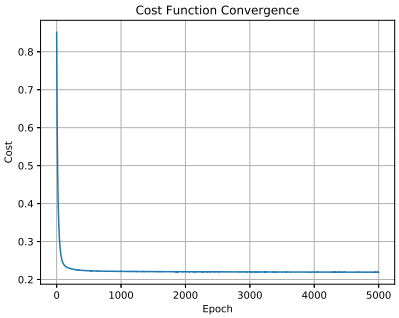
Cada função foi aproximada usando a rede neural duas vezes, uma sem usar regularização L_2, e outra usando regularização L_2 com $\lambda = 0,002$. Os resultados desses testes encontram-se na Figura 1 usando a função *sum_gt_zero*, e na Figura 2 usando a função *xor*. Para cada teste foram realizadas 5000 iterações de treinamento.



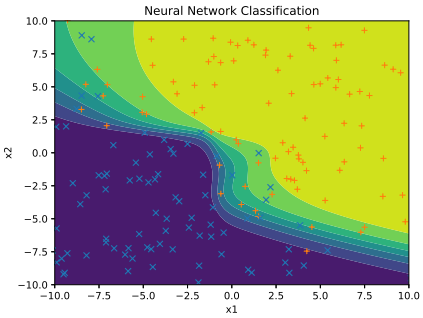
(a) Dataset gerado com ruído adicionado.



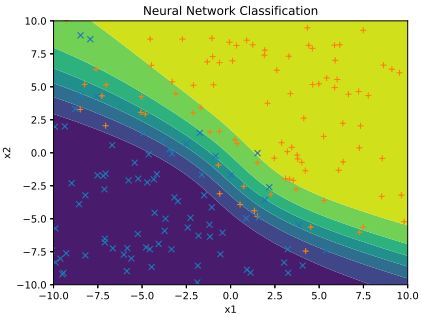
(b) Convergência sem regularização.



(c) Convergência com regularização.



(d) Predição da rede neural sem regularização.



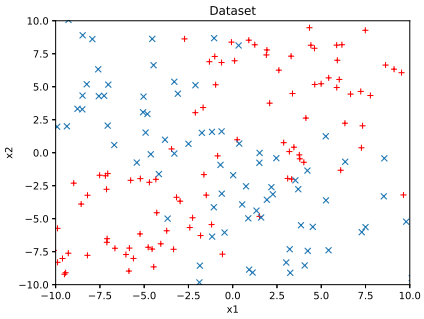
(e) Predição da rede neural com regularização.

Figura 1: Comparação do efeito da regularização usando a função sum_gt_zero.

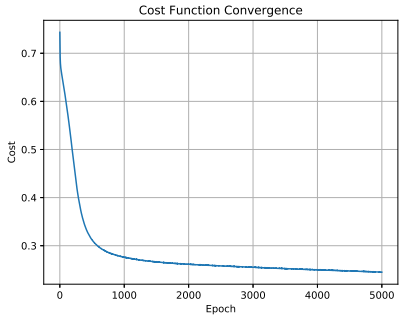
Comparando-se o treinamento com e sem regularização, observa-se que, sem usar a regularização, obteve-se um custo menor, porém com um *overfitting* visível. Em outras palavras, o treinamento sem regularização desviou significativamente do resultado ideal, que seria duas regiões delimitadas pela reta $y = -x$, devido ao ruído introduzido no *dataset*. Usando regularização, o treinamento obteve um resultado melhor.

A mesma tendência se repete no treinamento para a função *xor*. O treinamento feito sem regularização teve um valor menor para a função de custo, porém com sinais perceptíveis de *overfitting*. Dado que a aproximação perfeita da função resultaria numa predição que dividiria o plano cartesiano em duas regiões, uma contendo o primeiro e terceiro quadrante, e a outra contendo o segundo e quarto quadrante, observa-se que o uso de regularização promoveu uma aproximação melhor.

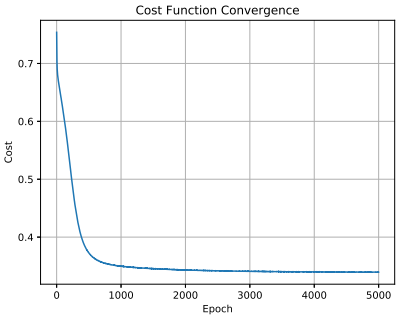
Levando em conta os experimentos realizados, pode-se concluir que o uso da regularização auxilia significativamente na redução de *overfitting* no treinamento de redes neurais.



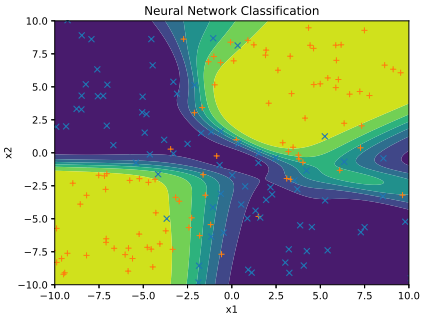
(a) Dataset gerado com ruído adicionado.



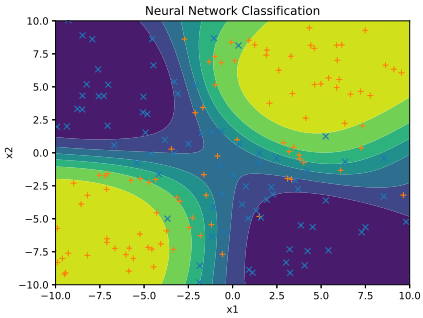
(b) Convergência sem regularização.



(c) Convergência com regularização.



(d) Predição da rede neural sem regularização.



(e) Predição da rede neural com regularização.

Figura 2: Comparação do efeito da regularização usando a função xor.

3 *Imitation Learning*

Um segundo teste foi realizado para verificar a capacidade de uma rede neural de executar aprendizado por imitação (*imitation learning*), ou seja, a tentativa de imitar uma determinada função. Nesse caso, a rede neural será aplicada para tentar imitar a evolução das juntas da perna direita de um robô humanoide durante um ciclo de caminhada.

Para esse teste, foi criado, novamente usando Keras, uma rede neural com as primeiras duas camadas com função de ativação *Leaky ReLU*, com $\alpha = 0,01$, sendo a primeira com 75 neurônios, e a segunda com 50. Usando o Keras, para criar cada camada com funçao de ativação *Leaky ReLU*, foi necessário primeiro criar uma camada com funçao de ativação linear, e uma camada posterior específica para a função *Leaky ReLU*. Por fim, a camada de *output* tem 20 neurônios e função de ativação linear.

Em seguida, a rede foi treinada, usando o *input* de ângulos das juntas da perna direita, ao longo de 30.000 iterações de treinamento. Após o treinamento, a rede foi usada para realizar a predição dos ângulos das juntas em um ciclo de caminhada, porém com resolução maior que os dados do *input*. Ou seja, enquanto este forneceu os ângulos das juntas com resolução de $0,008ms$, a predição foi feita para fornecer os ângulos a cada $0.001ms$ do ciclo de caminhada. Os gráficos relativos ao treinamento do *imitation learning* encontram-se na Figura 3.

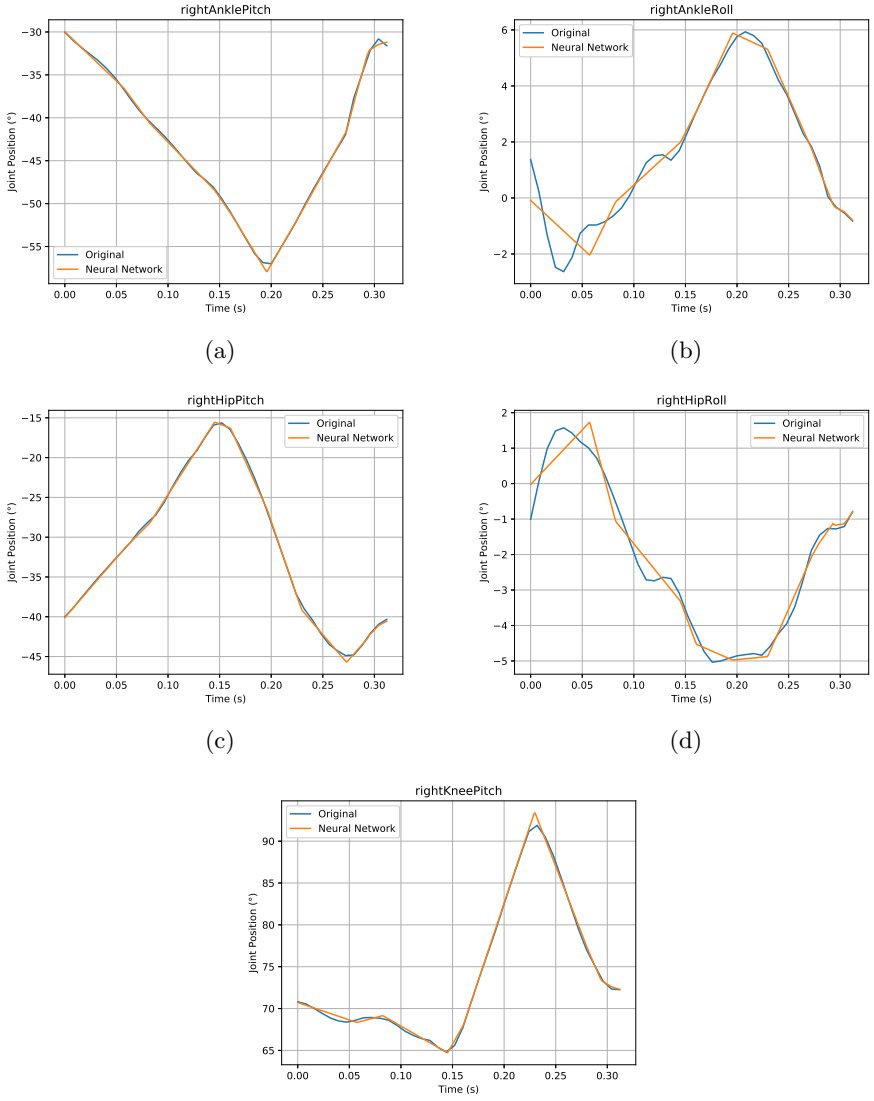


Figura 3: Resultados do *imitation learning*.

Observando os gráficos, conclui-se que a rede neural foi capaz de aproximar bem a evolução dos ângulos de todas as 5 juntas consideradas nos momentos em que elas não variavam bruscamente. Nos momentos em que as juntas variavam rapidamente, porém, a predição se distanciou significativamente dos valores esperados, como pode ser observado na figura correspondente às juntas *rightAnkleRoll* e *rightHipRoll*. Em outras palavras, essa implementação de rede neural só foi capaz de realizar o *imitation learning* de forma satisfatória quando as velocidades dos movimentos das juntas variavam pouco com o tempo. Ou seja, para que seja possível realizar um *imitation learning* satisfatório para todo o movimento, é necessário utilizar uma rede neural mais profunda, e com mais tempo de treinamento.