Instituto Tecnológico de Aeronáutica - ITA
Inteligência Artificial para Robótica Móvel - CT213
Aluno: Reynaldo Santos de Lima

Relatório do Laboratório 6 - Redes Neurais

1 Breve Explicação em Alto Nível da Implementação

O método de propagação *forward* foi implementado com lógica semelhante à apresentada em sala, tendo o código a seguir:

```
class NeuralNetwork:
        (...)
        def __init__(self, num_inputs, num_hiddens, num_outputs, alpha):
                (...)
        def forward_propagation(self, inputs):
                (...)
                for l in {1,2}:
                        w = self.weights[l]
                        aux = w@a[l-1]
                        z[l] = w@a[l-1]+self.biases[l]
                        a[l] = sigmoid(z[l])
                return z, a
        def compute_cost(self, inputs, expected_outputs):
                (...)
```

Continuando a lógica, o próximo método implementado foi o de cálculo de gradiente para a propagação *back*, implementado de forma vetorizada de modo a melhorar a performance:

```
        def compute_gradient_back_propagation(self, inputs, expected_outputs):
                (...)
                m = inputs.shape[1]
                w = self.weights[2]
                delta_2 = (y_hat - y) # definição dos deltas, vetorização
                delta_1 = np.transpose(w)@delta_2*sigmoid_derivative(z[1])
                weights_gradient[2] = delta_2 @ np.transpose(a[1]) / m
                weights_gradient[1] = delta_1 @ np.transpose(a[0]) / m
                biases_gradient[2] = delta_2 @ np.ones((m, 1)) / m
                biases_gradient[1] = delta_1 @ np.ones((m, 1)) / m
                return weights_gradient, biases_gradient
```

Por fim, implementou-se o método de propagação *back*, de lógica comum aos algoritmos de descida de gradiente.

2 Figuras Comprovando Funcionamento do Código

2.1 Função de Classificação *sum_ gt_ zeros*

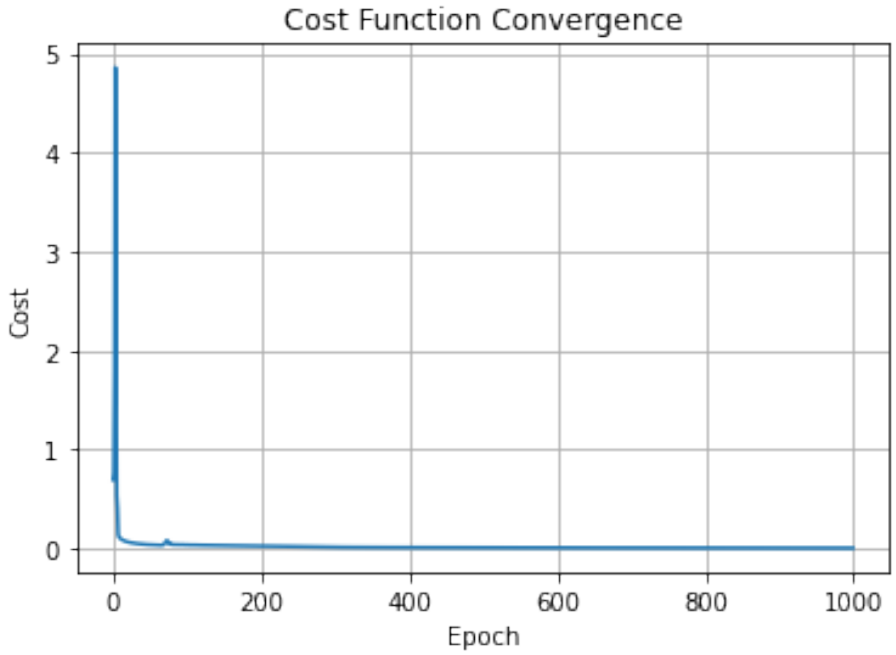

Figura 1: Função custo, *sum_ gt_ zeros*.

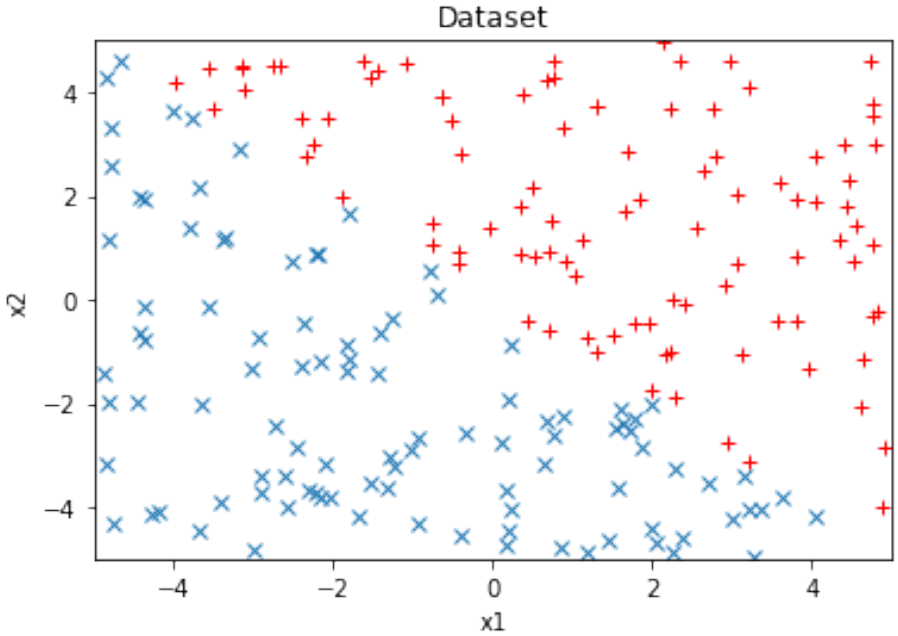

Figura 2: Conjunto de dados, *sum_ gt_ zeros*.

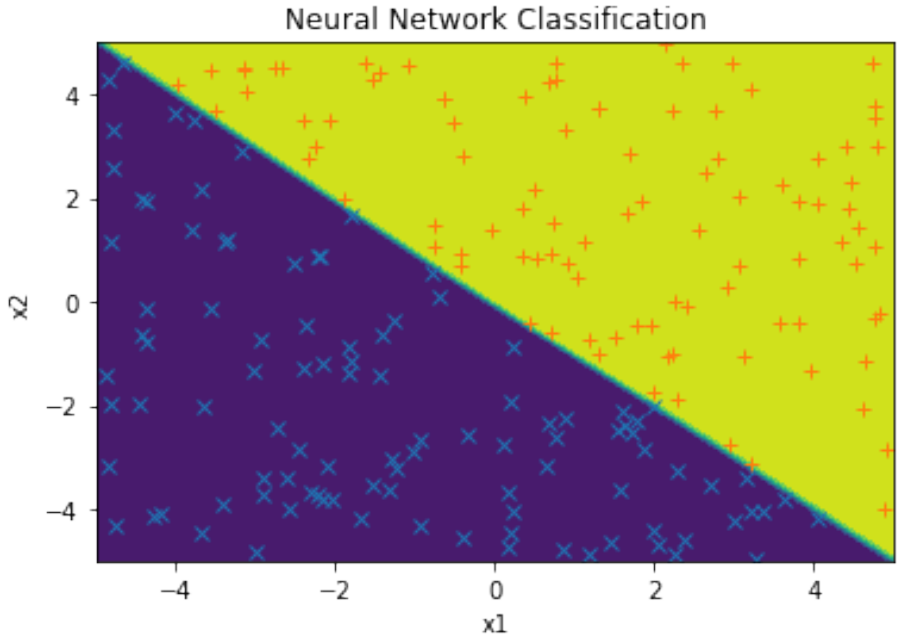

Figura 3: Resultado da classificação da rede neural, *sum_gt_zeros*.

2.2 Função de Classificação *XOR*

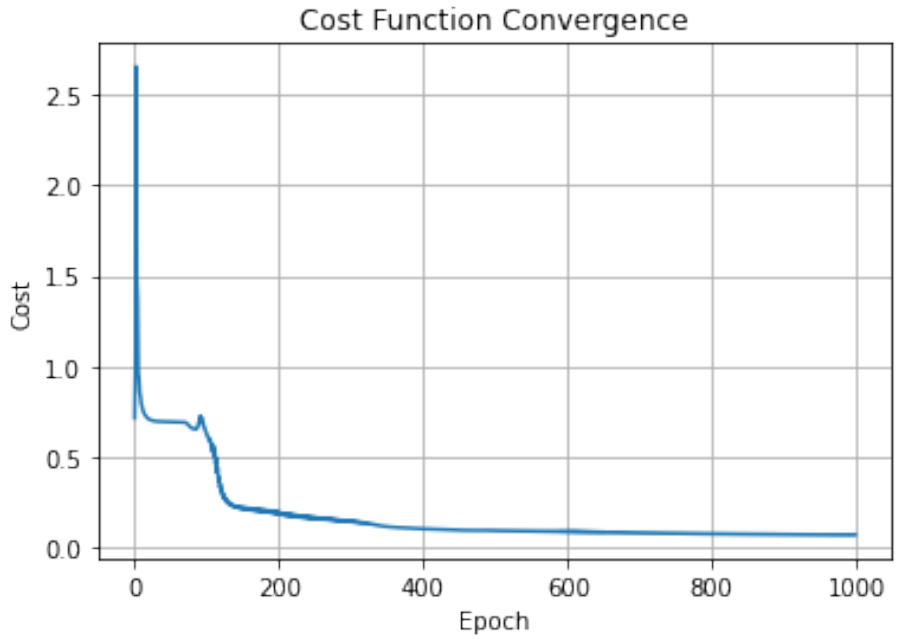

Figura 4: Função custo, *XOR*.

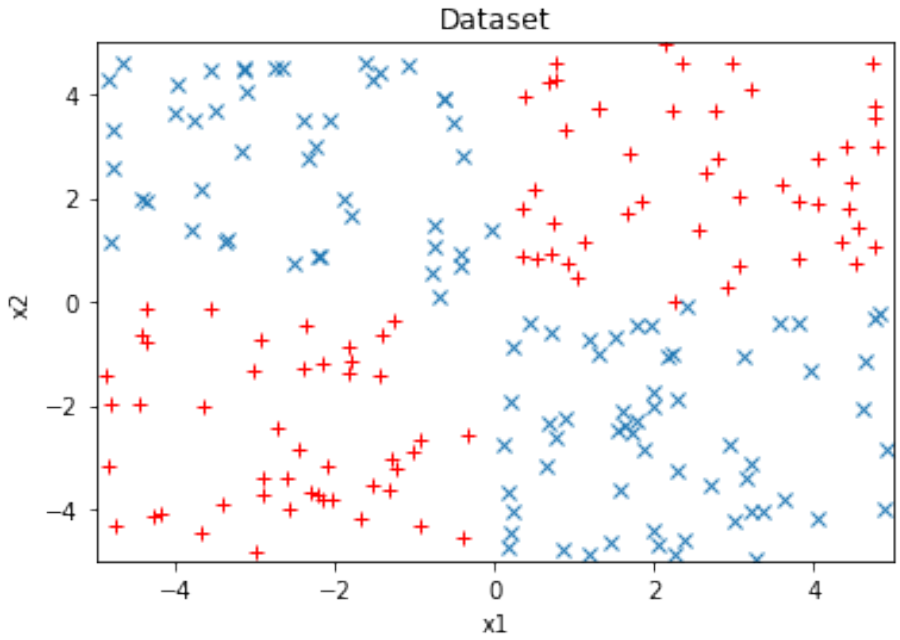

Figura 5: Conjunto de dados, *XOR*.

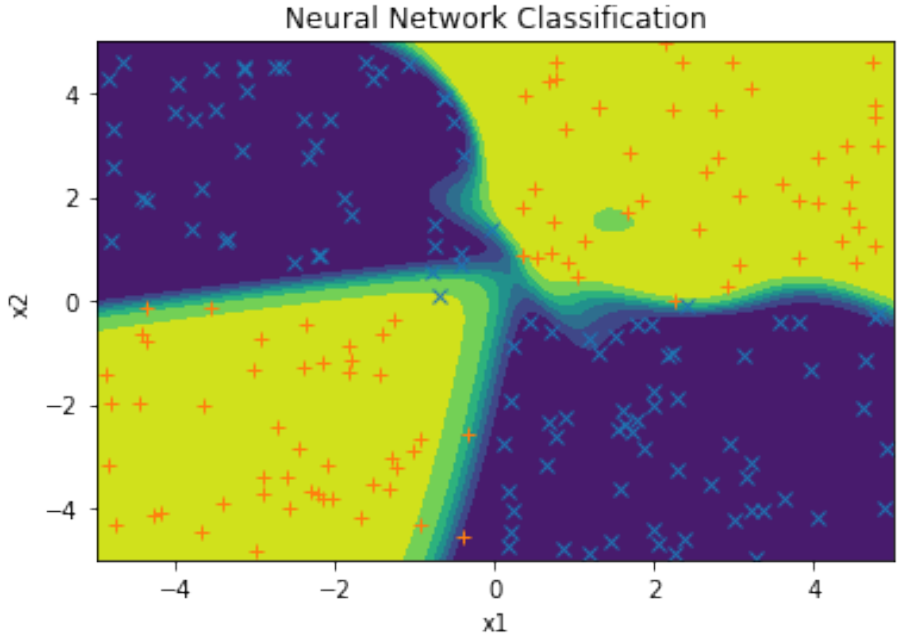

Figura 6: Resultado da classificação da rede neural, *XOR*.

2.3 Segmentação de Cores

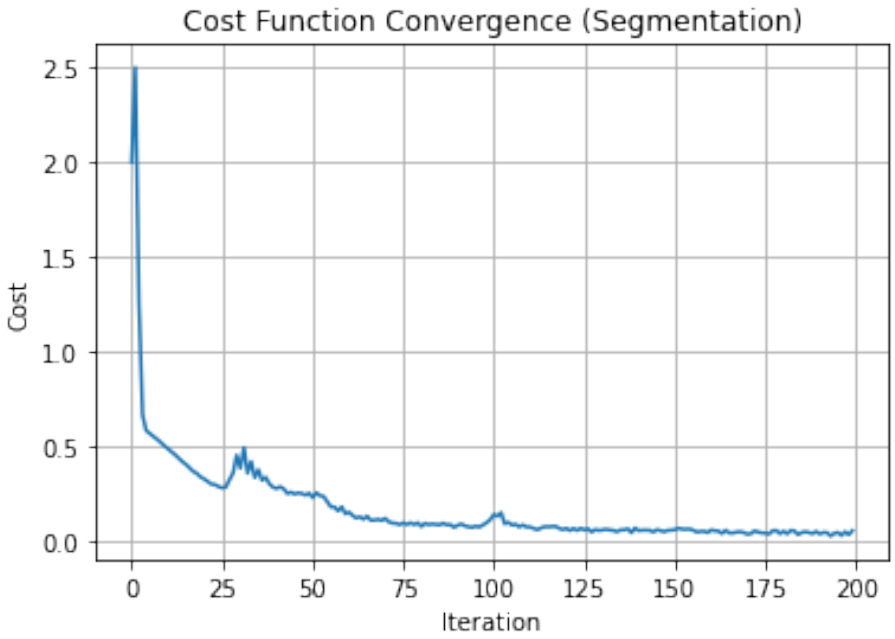

Figura 7: Função custo, aplicação em segmentação de cores.

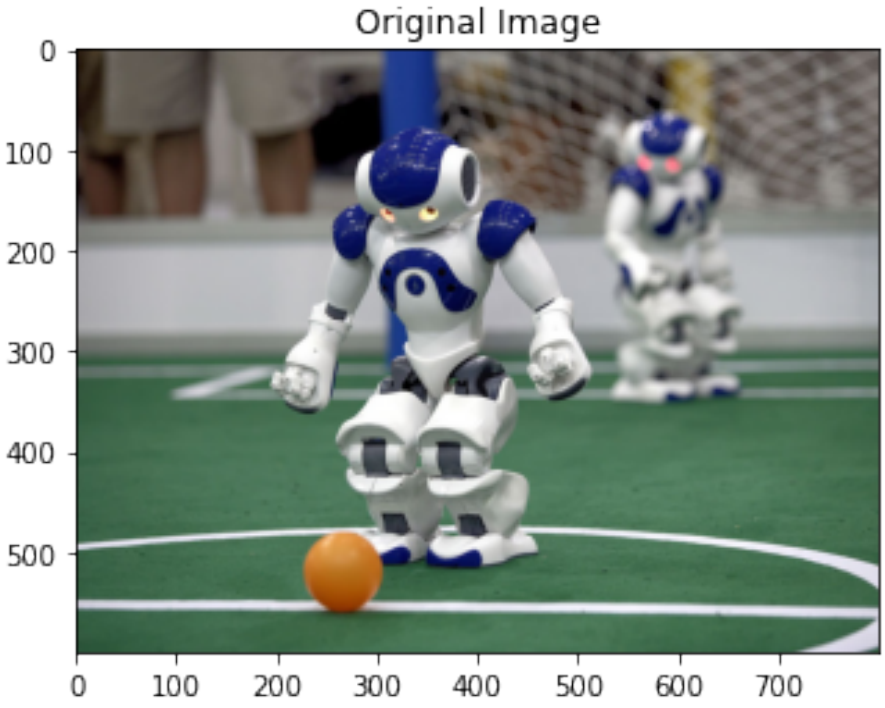

Figura 8: Imagem original.

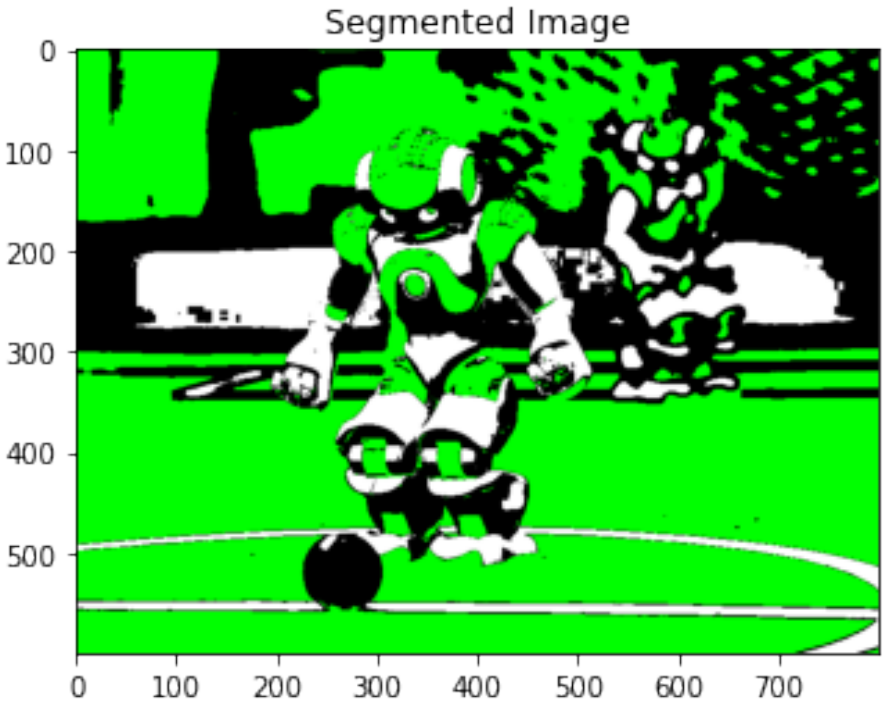

Figura 9: Resultado da segmentação da rede neural.

3 Discussões

Observou-se como as redes neurais, ainda que simples, são funcionais para tratar de problemas de classificação. Destaca-se inicialmente o resultado principalmente do teste com a função "XOR", dado que uma solução de uma reta, por exemplo, não resolveria o problema.

Na segmentação de cores, por sua vez, nota-se a aplicabilidade da solução, ainda que a implementação tenha sido feita para fins ilutrativos do que redes neurais otimizadas podem vir a fazer. Destaca-se também a vantagem em implementar o código de forma vetorizada dado que foi utilizada uma linguagem interpretativa na implementação.