Instituto Tecnológico de Aeronáutica - ITA
CT-213 - Inteligência Artificial aplicada à Robótica Móvel
Aluno: Ulisses Lopes da Silva

Relatório do Laboratório 6 - Redes Neurais

---

1 Breve Explicação em Alto Nível da Implementação

Uma **rede neural artificial** é um modelo computacional levemente inspirado no funcionamento dos neurônios biológicos. Ela é composta por camadas de neurônios, conectadas entre si por pesos ajustáveis. No processo de aprendizado, a rede busca ajustar esses pesos de modo a minimizar a diferença entre suas saídas e os valores esperados. O funcionamento de uma rede neural pode ser dividido em duas etapas principais: a **propagação direta** (*forward propagation*), onde os dados de entrada atravessam a rede até produzir uma predição; e a **retropropagação** do erro (*backpropagation*), onde os pesos e biases da rede são ajustados com base no erro da saída, utilizando o método do gradiente descendente.

No presente laboratório, foi implementada uma rede neural de **três camadas** (entrada, escondida e saída) para resolver problemas de classificação, especificamente a segmentação de cores numa imagem, bem como foi testada com funções tais como `sum_gt_zero()` e `xor()`. A rede recebe entradas normalizadas e, por meio de uma camada escondida com ativação sigmoide, gera uma saída no espaço de cores do sistema RGB, classificando-a, de modo simplificado, entre verde ou branco, e, para cores indefinidas, o preto. O processo de treinamento é realizado com a aplicação do algoritmo de backpropagation, que ajusta os pesos com base nos gradientes da função de custo logístico multiclasse. Esta função mede o quão distantes estão as predições da rede em relação às saídas reais.

A fase de ***forward propagation*** foi implementada utilizando a função de ativação sigmoide em ambas as camadas (escondida e de saída). Os dados de entrada atravessam os pesos e bias da camada escondida, gerando ativações intermediárias, que por sua vez são propagadas até a saída. O resultado final é uma matriz com as ativações da última camada, representando a estimativa da rede.

Na fase de ***backpropagation***, os gradientes da função de custo em relação aos pesos e bias foram calculados de forma vetorizada. A derivada da função sigmoide foi utilizada para retropropagar o erro da saída para a camada escondida. Os gradientes foram normalizados pelo número de amostras e utilizados para atualizar os parâmetros da rede com a taxa de aprendizado definida.

Ao final de cada época de treinamento, foi calculado e armazenado o valor da função de custo, permitindo observar a convergência da rede. O comportamento da rede foi também avaliado visualmente por meio de gráficos de regiões de decisão, mostrando como a rede aprende a separar os dados em regiões associadas a diferentes classes.

2 Figuras Comprovando Funcionamento do Código

2.1 Função de Classificação *sum_ gt_ zeros*

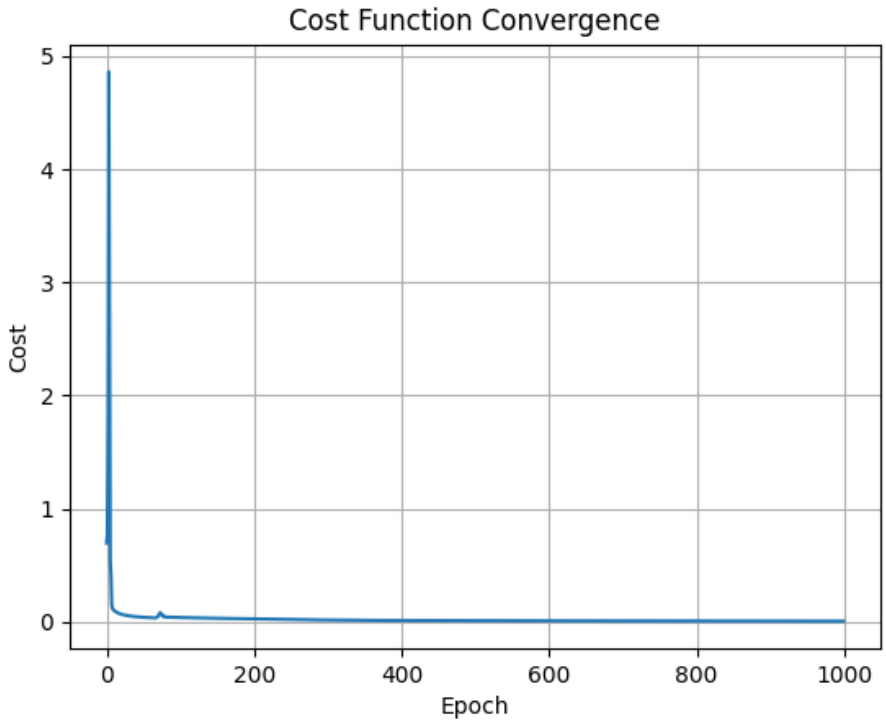

Fig. 1: Convergência da função de custo de `sum_gt_zero()`

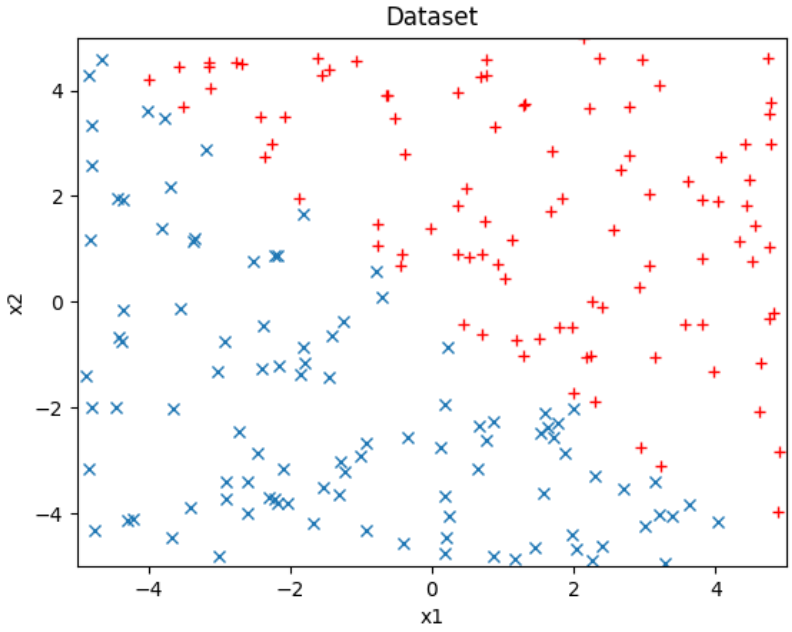

Fig. 2: *Dataset* utilizado para `sum_gt_zero()`

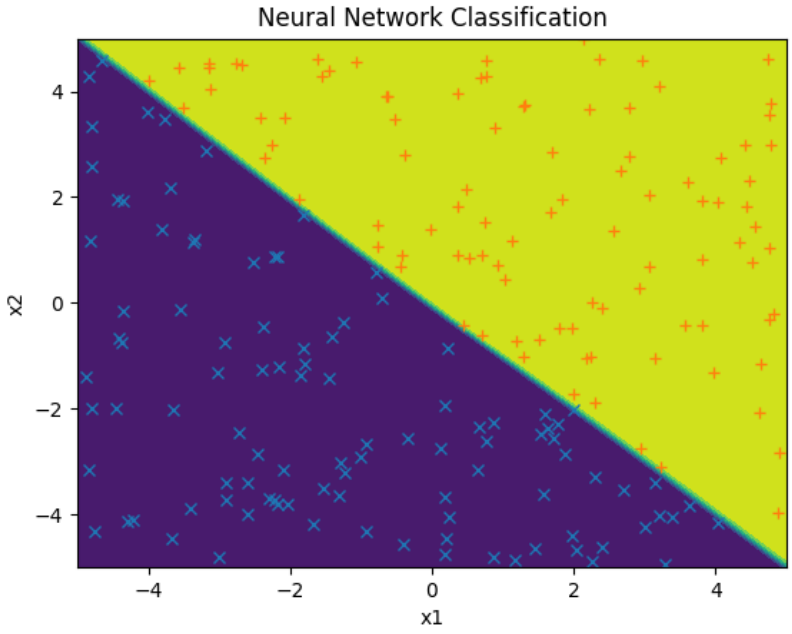

Fig. 3: Classificação da Rede Neural para `sum_gt_zero()`

2.2 Função de Classificação *XOR*

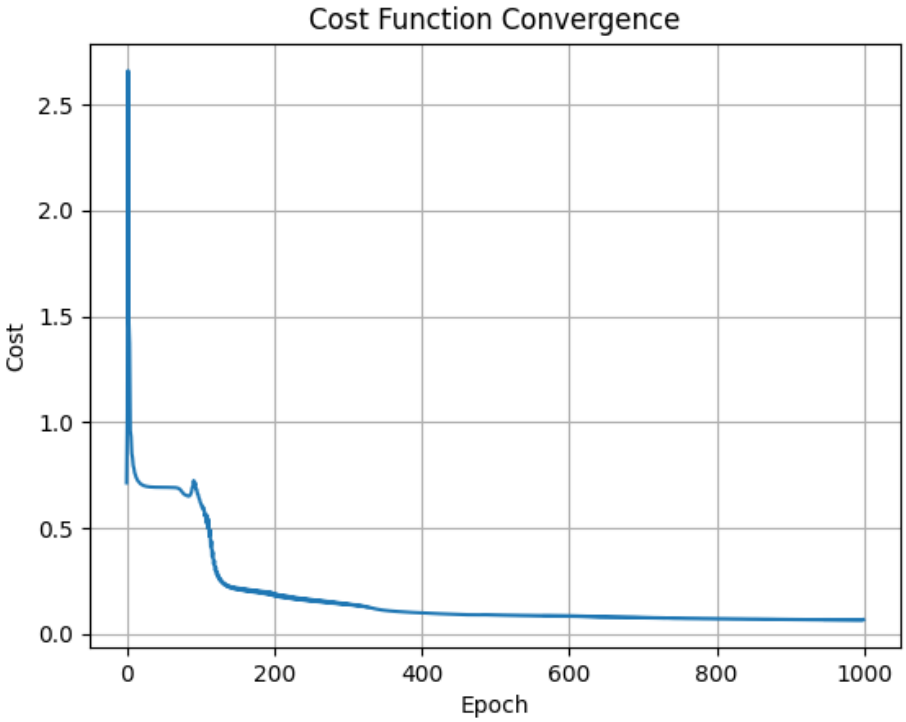

Fig. 4: Convergência da função de custo de `xor()`

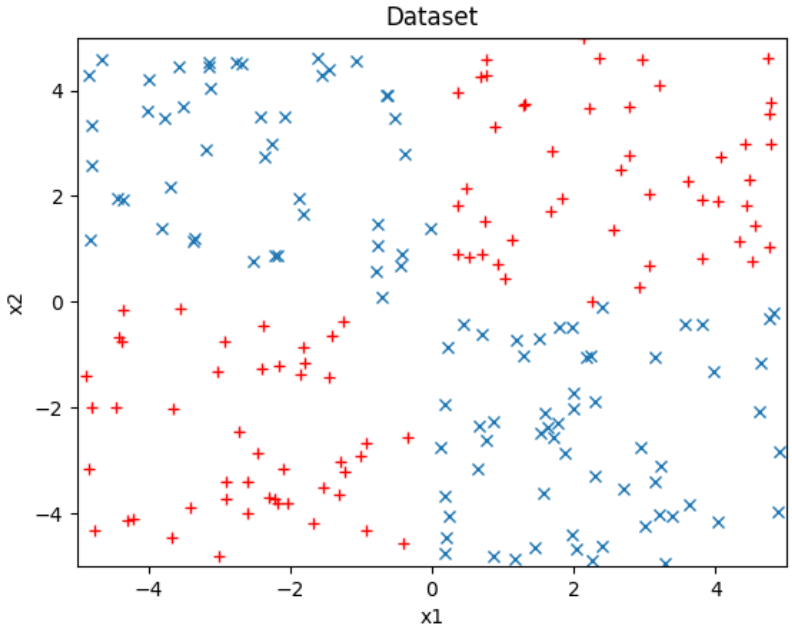

Fig. 5: *Dataset* utilizado para `xor()`

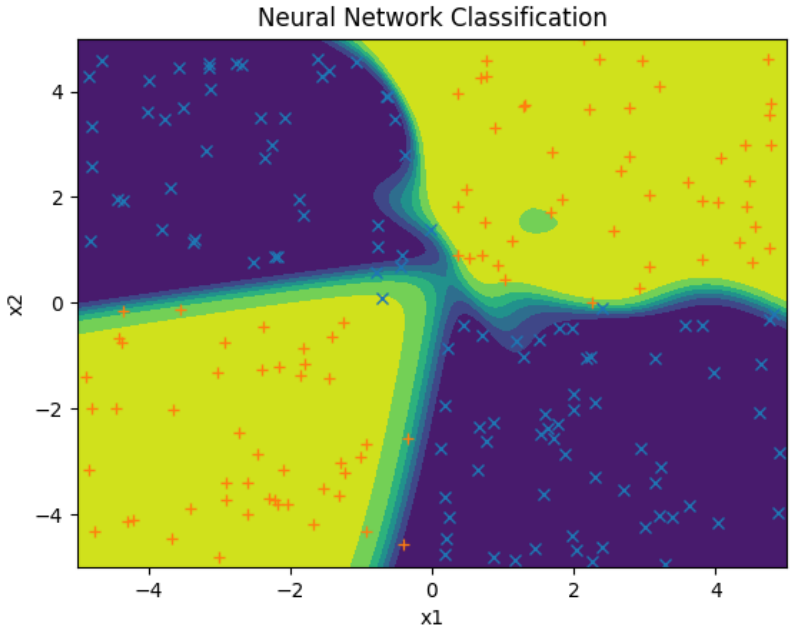

Fig. 6: Classificação da Rede Neural para `xor()`

2.3 Segmentação de Cores

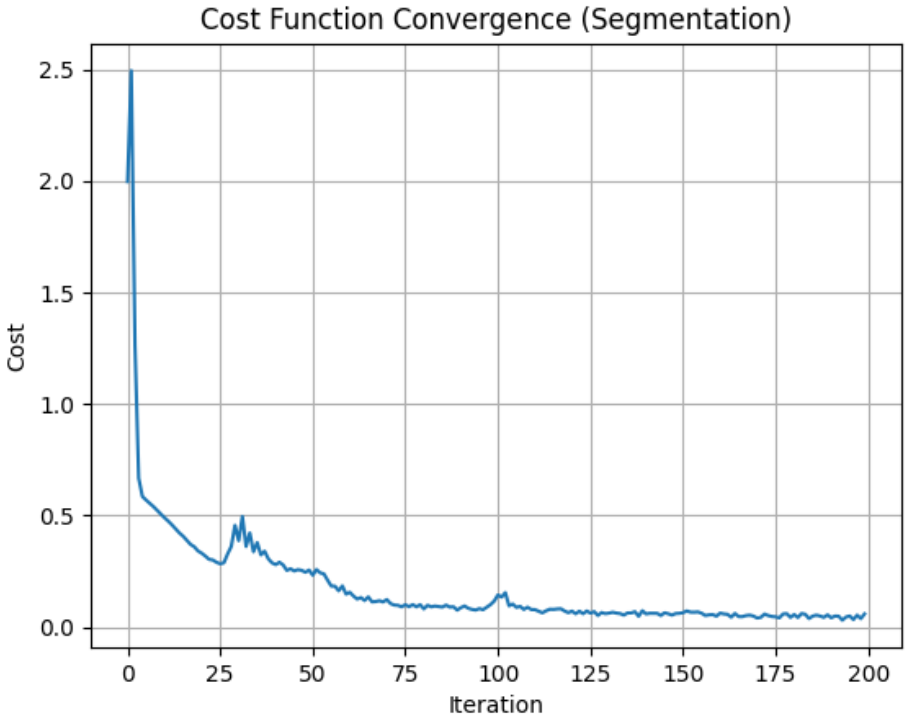

Fig. 7: Convergência da função de custo da segmentação de cores

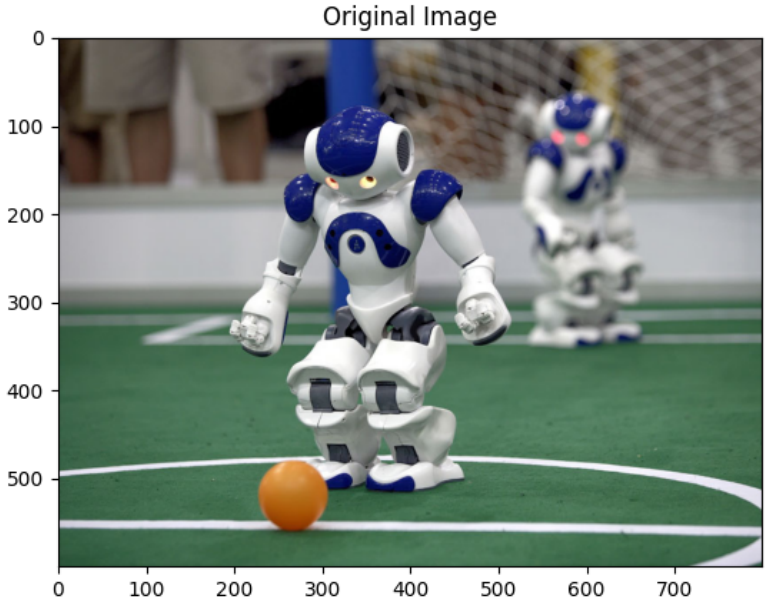

Fig. 8: Imagem original utilizada para a segmentação de cores

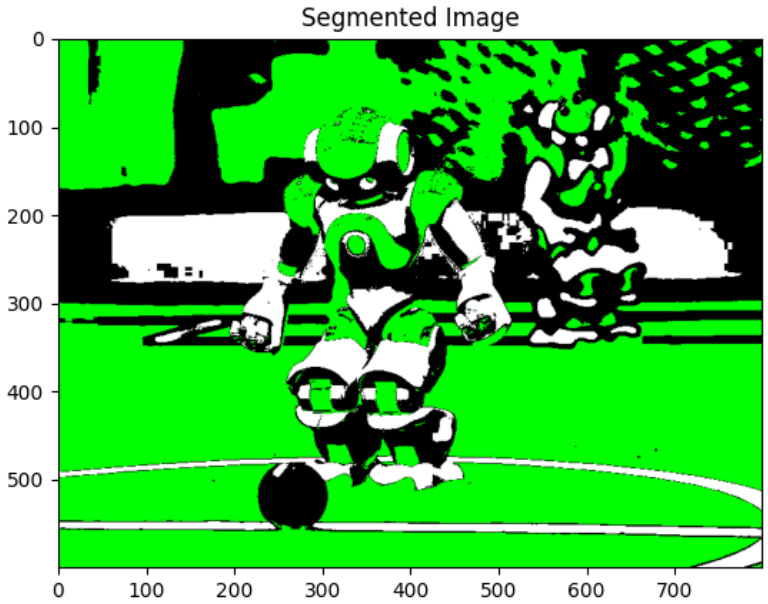

Fig. 9: Classificação da Rede Neural para a segmentação de cores, a partir da imagem original

3 Discussões

A implementação da rede neural (RN) se mostrou especialmente eficiente nos testes com as funções `sum_gt_zero()` e `xor()`. Na primeira função, conforme se verifica na imagem final de classificação do *dataset*, a rede teve um desempenho praticamente "perfeito"na generalização. Isso, provavelmente, se deve ao fato de que o conjunto de dados estava bem distribuído, favorecendo a correta generalização, bem como o ajuste final poderia ser traduzido em uma função linear.

Para a função `xor()`, a classificação também foi boa, mas já se percebem algumas "imperfeições"na imagem, apesar de estar muito bem ajustado ao que o conjunto de dados apresentou. Neste caso, a função é não-linear e classicamente utilizada para testar a capacidade de redes neurais com camadas ocultas. É interesante notar que a imagem da região de decisão mostra claramente que a rede aprendeu fronteiras curvas complexas, já que, em certas áreas do *dataset* `xor()`, há regiões esparsas de pontos, o que levou a RN a "aprender"a delimitar os pontos de forma curva. Há áreas bem delimitadas, mas também pequenos erros residuais, especialmente em regiões de transição. Isso indica que, embora a rede não tenha alcançado uma separação perfeita, ela generalizou bem sem se tornar excessivamente sensível ao ruído do treinamento.

Para a segmentação de cores da imagem, a RN se mostrou menos eficiente, apesar de ainda ter separado bem as cores principais nas regiões de interesse, que eram o campo de futebol e as cores do robô. Todavia, devido ao caráter distribuído das cores na realidade (o *range* de cores é bastante variável), resolução, contornos indefinidos etc., bem como a limitação de camadas da rede e a simplicidade da classificação (apenas verde e branco como cores principais, e preto para cores "indefinidas"), ela acabou classificando erroneamente muitos pontos da imagem, "englobando-os"no verde. Para o branco, todavia, a RN classificou razoavelmente bem.

A presença de classificações imperfeitas, especialmente visível nos contornos das regiões de decisão, não é um problema — pelo contrário, ela é uma consequência de uma rede que aprendeu a generalizar e não apenas memorizar. Quando a rede acerta a maior parte das amostras, mas erra suavemente em regiões fronteiriças, ela está demonstrando capacidade de adaptação a dados não vistos. Isso é preferível a uma classificação perfeita (em treinamento) que leva a *overfitting*.

Os gráficos de convergência da função de custo mostram um comportamento típico de treinamento eficaz. Em todos os casos, observa-se uma queda acentuada no custo nas primeiras iterações, indicando que a rede rapidamente encontra uma direção promissora no espaço de parâmetros. Após essa fase inicial, o custo vai diminuindo de forma mais gradual.

Contudo, nota-se a presença de pequenos picos ao longo do treinamento — especialmente visíveis na segmentação de cores e no problema xor. Esses picos não indicam falha no treinamento, mas refletem o comportamento oscilatório do gradiente estocástico, já que o treinamento ocorre com mini-batches. Essa variabilidade é esperada e até desejável, pois evita que a rede fique presa em mínimos locais rasos, favorecendo uma melhor generalização.