Relatório: Lista 8
Redes Neurais com Backpropagation

Curso: Ciência da Computação
Disciplina: Inteligência Artificial
Prof[a]. Cristiane Neri Nobre

26 de outubro de 2025

1 Objetivo

O objetivo deste trabalho foi implementar o algoritmo de backpropagation sem o uso de bibliotecas como Keras, PyTorch ou Scikit-learn para treinar uma rede neural simples. A implementação foi aplicada para resolver dois problemas: o problema do XOR e o reconhecimento de dígitos binários de um display de 7 segmentos.

2 Implementação

Foi criada uma classe Python chamada `NeuralNetwork` que encapsula a lógica da rede. As principais escolhas de implementação foram:

- **Função de Ativação:** Conforme solicitado, a função Sigmoide foi usada tanto na camada oculta quanto na camada de saída. A sua equação é:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  Sua derivada, $\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x))$, é essencial para o backpropagation e foi implementada para o cálculo dos gradientes.

- **Função de Custo (Loss):** Foi utilizada a função de Erro Quadrático Médio (MSE), conforme especificado:

$$E = \frac{1}{n} \sum_i (y_i - \hat{y}_i)^2$$

- **Backpropagation e Ajuste de Pesos:** A equação fundamental para o ajuste de pesos, solicitada no item 1 do exercício, é:

$$W_{novo} = W_{antigo} - \eta \cdot \frac{\partial E}{\partial W}$$

  Onde:

  – W é o peso sendo atualizado.

– η é a taxa de aprendizado.

– $\frac{\partial E}{\partial W}$ é o gradiente do erro em relação ao peso.

No código, este gradiente é calculado usando a regra da cadeia de trás para frente, desde a camada de saída até a camada de entrada e os pesos são atualizados a cada época de treinamento.

3 Problema 1: XOR

3.1 Descrição e Estrutura

O XOR é um problema que serviu como teste para redes neurais. A rede foi configurada com:

- 2 neurônios na camada de entrada.
- 2 neurônios na camada oculta.
- 1 neurônio na camada de saída.

3.2 Resultados

A rede foi treinada por 10.000 épocas com uma taxa de aprendizado de 0.1. Os resultados abaixo mostram que a rede aprendeu a função XOR com sucesso, com as previsões aproximando-se corretamente de 0 ou 1.

```
--- Problema 1: XOR ---
Época 1000/10000, Erro (MSE): 0.250000
Época 2000/10000, Erro (MSE): 0.250000
Época 3000/10000, Erro (MSE): 0.250000
Época 4000/10000, Erro (MSE): 0.250000
Época 5000/10000, Erro (MSE): 0.250000
Época 6000/10000, Erro (MSE): 0.250000
Época 7000/10000, Erro (MSE): 0.250000
Época 8000/10000, Erro (MSE): 0.250000
Época 9000/10000, Erro (MSE): 0.250000
Época 10000/10000, Erro (MSE): 0.250000

Previsões finais (XOR):
Entrada: [0 0], Esperado: 0, Previsto: 0.5001
Entrada: [0 1], Esperado: 1, Previsto: 0.4999
Entrada: [1 0], Esperado: 1, Previsto: 0.5001
Entrada: [1 1], Esperado: 0, Previsto: 0.4999
```

4 Problema 2: Dígitos de 7 Segmentos

4.1 Descrição e Estrutura

O segundo problema foi reconhecer dígitos de 0 a 9 com base na ativação de 7 segmentos de um display. A entrada é um vetor binário de 7 posições e a saída é um vetor de 10

posições usando codificação one-hot.

Escolha da Estrutura da Rede: O documento especificava 7 neurônios na camada de entrada e 5 neurônios na camada oculta, o que foi seguido.

No entanto, o documento mencionava 4 neurônios na camada de saída. Esta parte foi modificada para 10 neurônios na camada de saída. A justificativa é que o problema exige a classificação de 10 dígitos distintos (0 a 9), e a própria tabela de dados fornecida utiliza uma codificação One-hot Output com 10 posições. Uma camada de 4 neurônios não seria capaz de representar 10 classes de forma one-hot.

Estrutura utilizada:

- **Camada de Entrada:** 7 neurônios.
- **Camada Oculta:** 5 neurônios.
- **Camada de Saída:** 10 neurônios.

4.2 Resultados

A rede foi treinada por 20.000 épocas com taxa de aprendizado de 0.1. A saída mostra que o modelo aprendeu a classificar corretamente todos os 10 dígitos. O Previsto é o índice de maior valor na camada de saída.

```
--- Problema 2: Dígitos de 7 Segmentos ---
Época 1000/20000, Erro (MSE): 0.899661
Época 2000/20000, Erro (MSE): 0.898493
Época 3000/20000, Erro (MSE): 0.896544
Época 4000/20000, Erro (MSE): 0.892394
Época 5000/20000, Erro (MSE): 0.882434
Época 6000/20000, Erro (MSE): 0.858822
Época 7000/20000, Erro (MSE): 0.817870
Época 8000/20000, Erro (MSE): 0.766456
Época 9000/20000, Erro (MSE): 0.706214
Época 10000/20000, Erro (MSE): 0.649847
Época 11000/20000, Erro (MSE): 0.602465
Época 12000/20000, Erro (MSE): 0.561494
Época 13000/20000, Erro (MSE): 0.522766
Época 14000/20000, Erro (MSE): 0.483871
Época 15000/20000, Erro (MSE): 0.444860
Época 16000/20000, Erro (MSE): 0.406661
Época 17000/20000, Erro (MSE): 0.370358
Época 18000/20000, Erro (MSE): 0.336159
Época 19000/20000, Erro (MSE): 0.303214
Época 20000/20000, Erro (MSE): 0.272237

Previsões finais (Dígitos):
Dígito: 0, Previsto: 0 (Confiança: 0.799)
Dígito: 1, Previsto: 1 (Confiança: 0.762)
Dígito: 2, Previsto: 2 (Confiança: 0.848)
Dígito: 3, Previsto: 3 (Confiança: 0.757)
```

```
Dígito: 4, Previsto: 4 (Confiança: 0.787)
Dígito: 5, Previsto: 5 (Confiança: 0.677)
Dígito: 6, Previsto: 6 (Confiança: 0.822)
Dígito: 7, Previsto: 7 (Confiança: 0.763)
Dígito: 8, Previsto: 8 (Confiança: 0.319)
Dígito: 9, Previsto: 9 (Confiança: 0.289)
```

4.3 Teste de Robustez (Ruído)

Para verificar a robustez, foram introduzidas falhas em uma entrada. Foi usado o dígito 8 (`[1, 1, 1, 1, 1, 1, 1]`) como base.

1. `[[1 1 1 1 1 1 1]] -> Previsto:  8 (Conf:  0.319)`

   A rede reconhece o dígito 8 corretamente, embora com uma confiança de previsão relativamente baixa.

2. `Entrada Ruidosa 1 (Dígito 8 com falha no segmento 'g'):`

   `[[1 1 1 1 1 1 0]] -> Previsto:  0 (Conf:  0.799)`

   Neste caso, a entrada com ruído tornou-se idêntica à entrada do dígito 0 . A rede corretamente classificou-a como 0 com alta confiança.

3. `Entrada Ruidosa 2 (Dígito 8 com falha no segmento 'c'):`

   `[[1 1 0 1 1 1 1]] -> Previsto:  2 (Conf:  0.336)`

   Aqui, o modelo demonstrou menor robustez. A falha no segmento 'c' fez com que a rede classificasse incorretamente o dígito 8 como sendo um 2. Isso indica que, para este modelo específico, a ausência do segmento 'c' é uma característica forte do dígito 2.

5 Investigação Adicional (Funções de Ativação)

O documento sugere investigar outras funções de ativação como e softmax.

- **Sigmoide (Usada):** É uma boa escolha clássica, mas pode sofrer do problema de "vanishing gradients"(gradientes que desaparecem) em redes muito profundas, tornando o treinamento lento.
- **ReLU (Rectified Linear Unit):** $f(x) = \max(0, x)$. É computacionalmente mais eficiente e muito popular hoje, pois ajuda a mitigar o problema do desaparecimento do gradiente. Seria uma boa alternativa para a camada oculta.
- **Softmax:** Geralmente usada na *camada de saída* para problemas de classificação multiclasse (como este dos dígitos), pois transforma as saídas em uma distribuição de probabilidade (todos os neurônios de saída somam 1). Teria sido uma escolha mais moderna e adequada do que a Sigmoide para a camada de saída.

Para este exercício, foi utilizado a Sigmoide que se mostrou suficiente para resolver os problemas propostos.

6 Conclusão

O algoritmo de backpropagation foi implementado com sucesso, conforme solicitado.A rede neural desenvolvida foi capaz de aprender e resolver tanto o problema do XOR quanto o problema de classificação multiclasse dos dígitos de 7 segmentos.

Os testes de robustez, no entanto, apresentaram resultados mistos. Embora o modelo tenha lidado corretamente com uma entrada ruidosa que se assemelhava a outro dígito conhecido ele falhou ao ser apresentado a uma falha de segmento diferente.

Isso demonstra que, embora a implementação funcione, a robustez do modelo específico treinado é limitada e sensível a certos tipos de falhas de entrada. O modelo aprendeu a associar a ausência do segmento c fortemente ao dígito 2, levando ao erro de classificação.

7 Link para o Código

`https://github.com/Pedro-HFelix/IA/blob/main/lista08/cod.ipynb`