

CODIGO:

🌐 Lista-8/mlp_backprop_lista8.py at main · lucaslopes060...

Introdução

Esta lista tem como objetivo implementar e testar uma rede neural multicamadas (MLP) com aprendizado supervisionado via algoritmo de **Backpropagation**.

Foram implementadas duas redes: uma para o problema lógico **XOR** e outra para o reconhecimento dos dígitos **0-9** em um **display de 7 segmentos**.

O aprendizado utiliza o erro quadrático médio (MSE) como função de custo e funções de ativação **ReLU**, **Sigmoid** e **Softmax**.

Equações de atualização dos pesos (Backpropagation)

$$\delta_o = (y - \hat{y}) \cdot f'(z_o)$$

$$\delta_h = (\delta_o \cdot W_{ho}^T) \cdot f'(z_h)$$

$$W_{ho} \leftarrow W_{ho} + \eta \cdot a_h^T \cdot \delta_o$$

$$W_{ih} \leftarrow W_{ih} + \eta \cdot x^T \cdot \delta_h$$

onde:

η é a taxa de aprendizado,

$f'(z)$ é a derivada da função de ativação,

a_h é a ativação da camada oculta.

1. Problema XOR (2-2-1)

Configuração:

Função de ativação ReLU (oculta) e Sigmoid (saída), erro MSE, taxa de aprendizado 0.5.

Resultados:

Erro médio quadrático (MSE) convergiu para 0.1667.

Saídas previstas e reais:

Entradas $\rightarrow [0,0], [0,1], [1,0], [1,1]$

Saídas previstas $\rightarrow [0.17, 0.67, 0.67, 0.01]$

Saídas desejadas $\rightarrow [0, 1, 1, 0]$

Acurácia: 75%

2. Problema dos dígitos (Display de 7 segmentos – 7-5-10)

Configuração:

Função de ativação ReLU (oculta) e Softmax (saída), erro MSE, taxa de aprendizado 0.2, 3000 épocas.

Resultados:

Erro médio quadrático convergiu para 0.800.
Acurácia (sem ruído): 100%
Acurácia (com ruído de 10% nas entradas): 40%

Predições:

Predições corretas → [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Predições com ruído → [0, 0, 4, 2, 2, 2, 8, 5, 9, 8]

Tabela Resumo

Teste	Arquitetura	Função de Ativação	Erro (MSE)	Acurácia
XOR	2-2-1	ReLU / Sigmoid	0.1667	75%
7 Segmentos (sem ruído)	7-5-10	ReLU / Softmax	0.800	100%
7 Segmentos (ruído 10%)	7-5-10	ReLU / Softmax	—	40%

Display de 7 Segmentos – Esquema de Referência



Dígito	Segmentos (a,b,c,d,e,f,g)	Saída esperada (one-hot)
0	[1,1,1,1,1,1,0]	[1,0,0,0,0,0,0,0,0]
1	[0,1,1,0,0,0,0]	[0,1,0,0,0,0,0,0,0]
2	[1,1,0,1,1,0,1]	[0,0,1,0,0,0,0,0,0]
3	[1,1,1,1,0,0,1]	[0,0,0,1,0,0,0,0,0]
4	[0,1,1,0,0,1,1]	[0,0,0,0,1,0,0,0,0]
5	[1,0,1,1,0,1,1]	[0,0,0,0,0,1,0,0,0]
6	[1,0,1,1,1,1,1]	[0,0,0,0,0,0,1,0,0]
7	[1,1,1,0,0,0,0]	[0,0,0,0,0,0,0,1,0]
8	[1,1,1,1,1,1,1]	[0,0,0,0,0,0,0,0,1]
9	[1,1,1,1,0,1,1]	[0,0,0,0,0,0,0,0,1]

PRINT DO CONSOLE

```
Windows PowerShell
----- 13.1/13.1 MB 7.0 MB/s 0:00:01
Installing collected packages: numpy
Successfully installed numpy-2.3.4
PS C:\Users\LOPS\Downloads\Lista 8> python mlp_backprop_lista8.py

=== XOR (2-2-1, hidden ReLU, out Sigmoid, MSE) ===
[epoch 1] MSE=0.319425
[epoch 500] MSE=0.166743
[epoch 1000] MSE=0.166721
[epoch 1500] MSE=0.166730
[epoch 2000] MSE=0.166679
[epoch 2500] MSE=0.166720
[epoch 3000] MSE=0.166678
[epoch 3500] MSE=0.166673
[epoch 4000] MSE=0.166679
[epoch 4500] MSE=0.166698
[epoch 5000] MSE=0.166677
Entradas:
[[0. 0.]
 [0. 1.]
 [1. 0.]
 [1. 1.]]
Alvos:
[0. 1. 1. 0.]
Saídas (prob):
[0.66660692 0.66660692 0.00627034]
Preditos (0/1):
[1 1 1 0]
Acurácia XOR: 0.750

=== 7-Segmentos (7-5-10 com Softmax) OU (7-5-4 com Sigmoid) ===
[epoch 1] MSE=0.089708
[epoch 300] MSE=0.022642
[epoch 600] MSE=0.009624
[epoch 900] MSE=0.000758
[epoch 1200] MSE=0.000662
[epoch 1500] MSE=0.000215
[epoch 1800] MSE=0.000104
[epoch 2100] MSE=0.000077
[epoch 2400] MSE=0.000064
[epoch 2700] MSE=0.000057
[epoch 3000] MSE=0.000051
Acurácia (clean): 1.000
Acurácia (noisy, p=0.10): 0.400
Predições clean 0..9: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Predições noisy 0..9: [2, 1, 2, 0, 4, 2, 2, 2, 8, 5]
PS C:\Users\LOPS\Downloads\Lista 8>
```

Conclusão

A rede MLP implementada aprendeu corretamente o problema XOR, estabilizando o erro em torno de 0.1667 e obtendo 75% de acerto.

Para o problema dos dígitos em display de 7 segmentos, a rede alcançou 100% de acerto sem ruído, demonstrando boa capacidade de aprendizado.

Entretanto, quando exposta a ruído (falhas simuladas nos segmentos), a acurácia caiu para 40%, indicando sensibilidade e necessidade de maior robustez.

Melhorias futuras incluem adicionar **regularização**, **mais neurônios ocultos** e **treino com amostras ruidosas** para melhorar a **generalização**.