Deep Learning

Backpropagation

Lucas Silveira Kupssinskü

Machine Learning Theory and Applications Laboratory
PUCRS

agosto de 2026

Overview

1. Introdução

2. Treinamento de um MLP

2.1 Forward Pass
2.2 Função de Custo
2.3 Backward Pass
2.4 Descida de Gradiente

3. Vetorização do Treinamento

Visão Geral

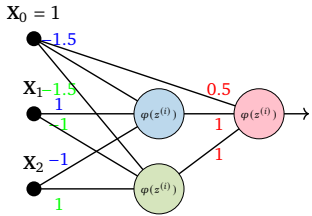
1. Introdução

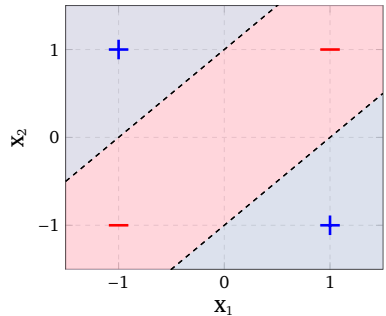
2. Treinamento de um MLP
   - 2.1 Forward Pass
   - 2.2 Função de Custo
   - 2.3 Backward Pass
   - 2.4 Descida de Gradiente

3. Vetorização do Treinamento

MLP

- Um MLP é um aproximador universal de funções[1,2]
- Contudo, o PLA (*Perceptron Learning Algorithm*) não funciona para um MLP





[1] Kurt Hornik, Maxwell Stinchcombe e Halbert White. “Multilayer feedforward networks are universal approximators”. Em: Neural networks 2.5 (1989), pp. 359–366

[2] George Cybenko. “Approximation by superpositions of a sigmoidal function”. Em: Mathematics of control, signals and systems 2.4 (1989), pp. 303–314

Inverno das Redes Neurais

- Com a publicação de Minsky e Papert (1969) enfatizando as limitações do Perceptron[3] o interesse em redes neurais diminuiu.
  - Pouca verba de pesquisa
  - Por que pesquisar um modelo linear enquanto temos modelos lineares melhores (Regressão Linear) e diversos modelos não-lineares?
  - MLP resolve problemas não-lineares, mas ninguém sabe como treinar um MLP
- Foi apenas 20 anos depois que começaram a surgir as primeiras publicações com um método de treinamento para MLPs e redes neurais em geral[3]
  - Hinton não alega ter inventado o *backpropagation*, mas ele foi um dos principais responsáveis pela sua popularização

[3] David E Rumelhart, Geoffrey E Hinton e Ronald J Williams. "Learning Internal Representations by Error Propagation, Parallel Distributed Processing, Explorations in the Microstructure of Cognition, ed. DE Rumelhart and J. McClelland. Vol. 1. 1986". Em: Biometrika 71 (1986), pp. 599–607.

Visão Geral

1. Introdução

2. Treinamento de um MLP
   2.1 Forward Pass
   2.2 Função de Custo
   2.3 Backward Pass
   2.4 Descida de Gradiente

3. Vetorização do Treinamento

Treinando um MLP

- Vamos calcular a mão uma iteração de treino em um MLP
- **Ideia:** Aplicar a descida de gradiente
- **Função de custo:** *Half Mean Squared Error*[4]

$$J(\boldsymbol{\theta}) = \frac{1}{2N} \sum_i \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \right)^2$$

- **Regra de atualização:**

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla J$$

Premissas para simplificar o problema:

- Apenas 2 camadas
- 1 neurônio em cada camada
- Desconsideramos o bias
- Temos apenas um exemplo $\mathbf{y} = 1, \mathbf{X}_1 = 1$

[4] O fator $1/2$ é convenção: simplifica a derivada do quadrático.

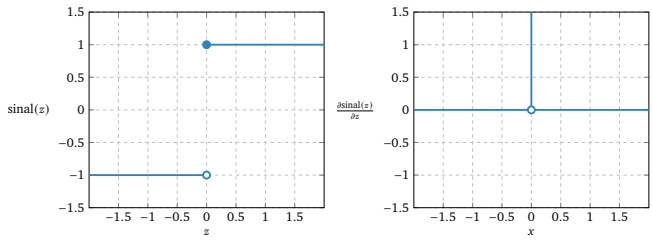
Treinando um MLP
Notação e Definições

- $l \in [0, 1, 2, ..., L]$ é o índice da camada do MLP (L é o número de camadas)
- $\mathbf{X}$ é a entrada do MLP (também pode ser chamado de $a^{(0)}$)
- $\mathbf{y}$ é a variável alvo
- $\boldsymbol{\theta}_{ij}^{(l)}$ é o peso da camada l que conecta a entrada j a saída i
- $z^{(l)}$ é a **pré-ativação** da unidade, $z^{(l)} = a^{(l-1)}\boldsymbol{\theta}^{(l)^\top}$
- $a^{(l)}$ é a **ativação** (saída) da unidade $a^{(l)} = \varphi(z^{(l)})$

Treinando um MLP
A função de Ativação

Função Sinal:

$$\varphi(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$



Treinando um MLP
A função de Ativação

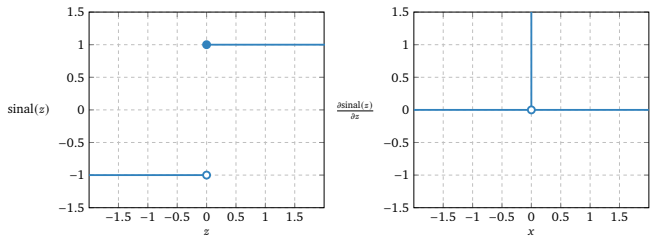
Função Sinal:

$$\varphi(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$

Problemas da função sinal

- Não é diferenciável em $z = 0$
- Sua derivada é 0 em todos os outros pontos

Isso impede o aprendizado via descida de gradiente.



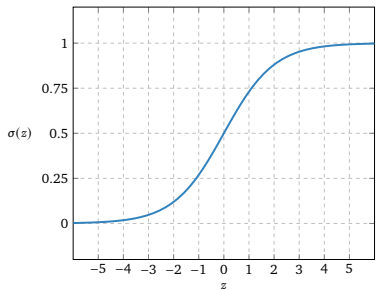
Treinando um MLP
A função de Ativação

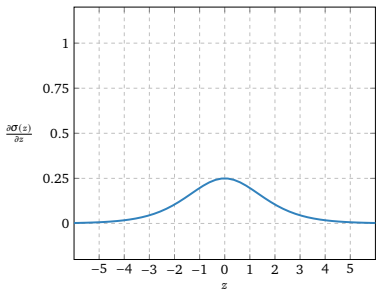
Função Sigmoide:

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

Solução

A função sigmoide é uma versão “suave” da função sinal e é diferenciável em todo seu domínio.





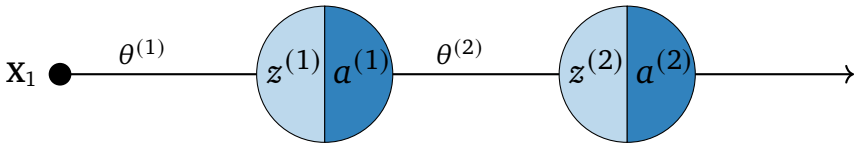
As Etapas do Treinamento de uma Rede Neural

A cada iteração de treinamento, seguimos quatro passos:

1. **Forward Pass:** Computar todas as ativações da rede.
2. **Loss Function:** Computar o valor da *Loss Function* que quantifica o quão diferente as saídas são em relação à variável alvo.
3. **Backward Pass:** Computar o gradiente da *Loss Function* em relação aos parâmetros da rede.
4. **Optimizer:** Usar os gradientes para atualizar os pesos.

```
1  model = MLP()
2  optimizer = SGD(model.parameters(),
3                  lr=0.1)
4  x = torch.tensor(1.0)
5  y = torch.tensor(1.0)
6  model.train()
7  optimizer.zero_grad()
8  y_hat = model(x)
9  loss = (y - y_hat)**2/2
10 loss.backward()
11 optimizer.step()
```

Forward Pass

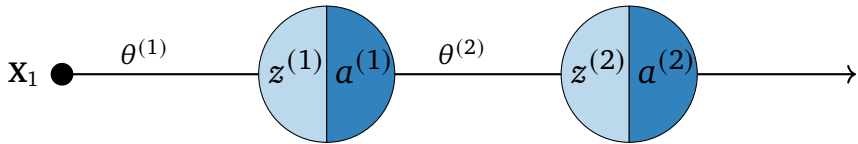


- Vamos considerar os pesos iniciais como: $\theta^{(1)} = -0.5$ e $\theta^{(2)} = 0.5$.
- Computando o loss para o par $X_1 = 1$, $y = 1$.

Passo a passo (Forward Pass):

1. $z^{(1)} = X_1 \cdot \theta^{(1)} = 1 \cdot (-0.5) = -0.5$

Forward Pass

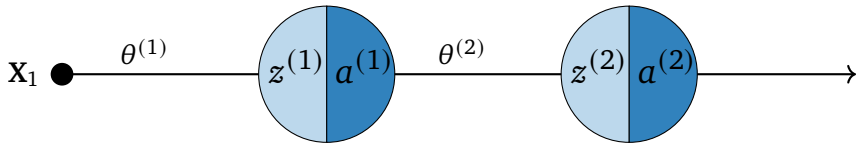


- Vamos considerar os pesos iniciais como: $\theta^{(1)} = -0.5$ e $\theta^{(2)} = 0.5$.
- Computando o loss para o par $X_1 = 1$, $y = 1$.

Passo a passo (Forward Pass):

1. $z^{(1)} = X_1 \cdot \theta^{(1)} = 1 \cdot (-0.5) = -0.5$
2. $a^{(1)} = \sigma(z^{(1)}) = \sigma(-0.5) = 0.3775$

Forward Pass

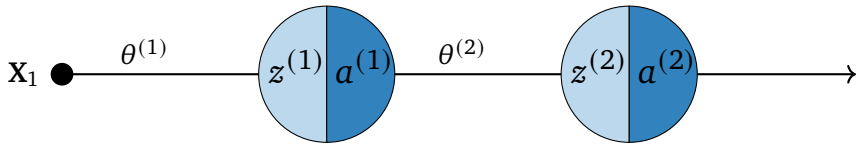


- Vamos considerar os pesos iniciais como: $\theta^{(1)} = -0.5$ e $\theta^{(2)} = 0.5$.
- Computando o loss para o par $X_1 = 1$, $y = 1$.

Passo a passo (Forward Pass):

1. $z^{(1)} = X_1 \cdot \theta^{(1)} = 1 \cdot (-0.5) = -0.5$
2. $a^{(1)} = \sigma(z^{(1)}) = \sigma(-0.5) = 0.3775$
3. $z^{(2)} = a^{(1)} \cdot \theta^{(2)} = 0.3775 \cdot 0.5 = 0.18875$

Forward Pass



- Vamos considerar os pesos iniciais como: $\theta^{(1)} = -0.5$ e $\theta^{(2)} = 0.5$.
- Computando o loss para o par $X_1 = 1$, $y = 1$.

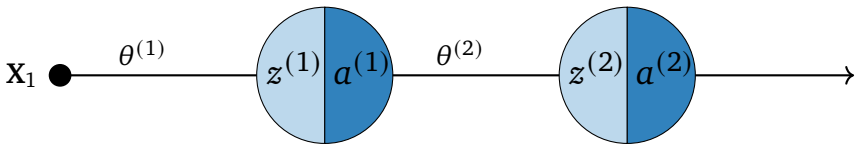
Passo a passo (Forward Pass):

1. $z^{(1)} = X_1 \cdot \theta^{(1)} = 1 \cdot (-0.5) = -0.5$
2. $a^{(1)} = \sigma(z^{(1)}) = \sigma(-0.5) = 0.3775$
3. $z^{(2)} = a^{(1)} \cdot \theta^{(2)} = 0.3775 \cdot 0.5 = 0.18875$
4. $\hat{y} = a^{(2)} = \sigma(z^{(2)}) = \sigma(0.18875) = 0.5470$

Resumindo

Esse é o *forward pass* da rede neural, onde calculamos as ativações e obtemos a saída da rede $\hat{y} = a^{(2)} = 0.5470$

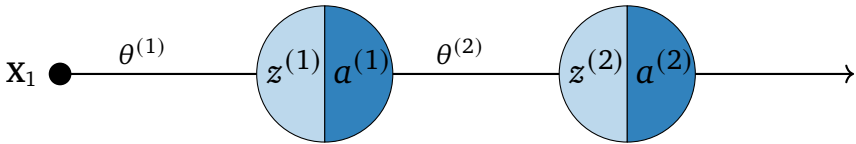
Loss Function



Nessa etapa computamos a *Loss Function* para avaliar o desempenho da rede.
Loss: $J(\theta) = \frac{1}{2}(y - \hat{y})^2$.

$$J(\theta) = \frac{1}{2}(1 - 0.5470)^2 = 0.1026045$$

Loss Function



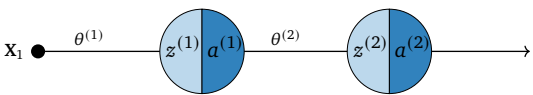
Nessa etapa computamos a *Loss Function* para avaliar o desempenho da rede.
Loss: $J(\theta) = \frac{1}{2}(y - \hat{y})^2$.

$$J(\theta) = \frac{1}{2}(1 - 0.5470)^2 = 0.1026045$$

Resumindo

Essa é a etapa de computar o valor da *Loss Function*. Observe que o valor da *Loss Function* não será usado diretamente nos gradientes nem na atualização dos pesos

Comparando com PyTorch

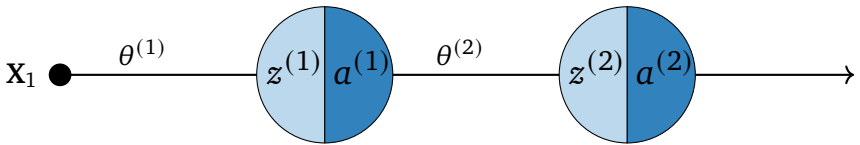


$$\hat{\mathbf{y}} = 0.5470$$

$$J(\theta) = 0.1026045$$

```python
class RedeSimples(nn.Module):
  def __init__(self) -> None:
    super().__init__()
    self.theta_1 = nn.parameter.Parameter(
                    torch.tensor(-0.5),
                    requires_grad=True)
    self.theta_2 = nn.parameter.Parameter(
                    torch.tensor(0.5),
                    requires_grad=True)
  def forward(self, x):
    z1 = x * self.theta_1
    a1 = torch.sigmoid(z1)
    z2 = a1 * self.theta_2
    a2 = torch.sigmoid(z2)
    return a2
x = torch.tensor(1.0)
y = torch.tensor(1.0)
model = RedeSimples()
y_hat = model(x)
loss = ((y - y_hat)**2)/2
print(f'y_hat: {y_hat}')
print(f'loss: {loss}')
# y_hat: 0.5470529198646545
# loss: 0.10258052498102188
```

Backward Pass



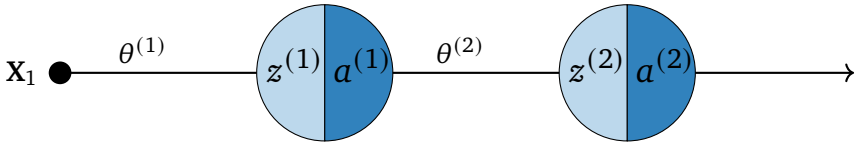
Nossa rede neural computa a seguinte função:

$$a^{(2)} = \sigma(\sigma(X_1 \cdot \theta^{(1)}) \cdot \theta^{(2)})$$

Logo, a função de custo é:

$$J(\theta) = \frac{1}{2}(y - a^{(2)})^2 = \frac{1}{2}\left(1 - \sigma\left(\sigma(X_1 \cdot \theta^{(1)}) \cdot \theta^{(2)}\right)\right)^2$$

Backward Pass



Nossa rede neural computa a seguinte função:

$$a^{(2)} = \sigma(\sigma(X_1 \cdot \theta^{(1)}) \cdot \theta^{(2)})$$

Logo, a função de custo é:

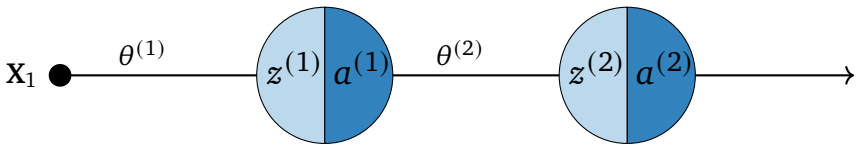
$$J(\theta) = \frac{1}{2}(y - a^{(2)})^2 = \frac{1}{2}\left(1 - \sigma\left(\sigma(X_1 \cdot \theta^{(1)}) \cdot \theta^{(2)}\right)\right)^2$$

Problema

Como computar o gradiente da função de custo em relação aos pesos $\theta^{(1)}$ e $\theta^{(2)}$?

$$\frac{\partial J}{\partial \theta^{(2)}} = ?$$
$$\frac{\partial J}{\partial \theta^{(1)}} = ?$$

Backward Pass



Solução: A função $J(\theta)$ é uma função composta, então usamos a regra da cadeia
Gradiente para $\theta^{(2)}$:

$$\frac{\partial J}{\partial \theta^{(2)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial \theta^{(2)}}$$

Gradiente para $\theta^{(1)}$:

$$\frac{\partial J}{\partial \theta^{(1)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial \theta^{(1)}}$$

Backward Pass

$$\frac{\partial J}{\partial \theta^{(2)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial \theta^{(2)}}$$

- $\frac{\partial J}{\partial a^{(2)}} = a^{(2)} - y$
- $\frac{\partial a^{(2)}}{\partial z^{(2)}} = a^{(2)}(1 - a^{(2)})$
- $\frac{\partial z^{(2)}}{\partial \theta^{(2)}} = a^{(1)}$

$$\begin{aligned}
\frac{\partial J}{\partial a^{(2)}} &= \frac{\partial \left(\frac{1}{2}(y - a^{(2)})^2 \right)}{\partial a^{(2)}} \\
&= (-1)2\frac{1}{2}(y - a^{(2)}) \\
&= a^{(2)} - y
\end{aligned}$$

Backward Pass

$$\frac{\partial J}{\partial \theta^{(2)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial \theta^{(2)}}$$

- $\frac{\partial J}{\partial a^{(2)}} = a^{(2)} - y$
- $\frac{\partial a^{(2)}}{\partial z^{(2)}} = a^{(2)}(1 - a^{(2)})$
- $\frac{\partial z^{(2)}}{\partial \theta^{(2)}} = a^{(1)}$

$$\begin{aligned}
\frac{\partial a^{(2)}}{\partial z^{(2)}} &= \frac{\partial \left(\frac{1}{1+e^{-z^{(2)}}} \right)}{\partial z^{(2)}} = \frac{\partial \left((1+e^{-z^{(2)}})^{-1} \right)}{\partial z^{(2)}} \\
&= (\cancel{-1})(1+e^{-z^{(2)}})^{-2} e^{-z^{(2)}} (\cancel{-1}) \\
&= \frac{1 + e^{-z^{(2)}} - 1}{(1+e^{-z^{(2)}})^2} \\
&= \frac{1}{1+e^{-z^{(2)}}} \left(\frac{\cancel{1+e^{-z^{(2)}}}^{1}}{\cancel{1+e^{-z^{(2)}}}} - \frac{1}{1+e^{-z^{(2)}}} \right) \\
&= \sigma(z^{(2)})(1 - \sigma(z^{(2)})) = a^{(2)}(1 - a^{(2)})
\end{aligned}$$

Backward Pass

$$\frac{\partial J}{\partial \theta^{(2)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial \theta^{(2)}}$$

- $\frac{\partial J}{\partial a^{(2)}} = a^{(2)} - y$
- $\frac{\partial a^{(2)}}{\partial z^{(2)}} = a^{(2)}(1 - a^{(2)})$
- $\frac{\partial z^{(2)}}{\partial \theta^{(2)}} = a^{(1)}$

$$\begin{aligned} \frac{\partial z^{(2)}}{\partial \theta^{(2)}} &= \frac{\partial \left(a^{(1)}\theta^{(2)}\right)}{\partial \theta^{(2)}} \\ &= a^{(1)} \end{aligned}$$

Backward Pass

$$\frac{\partial J}{\partial \theta^{(1)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial \theta^{(1)}}$$

- $\frac{\partial J}{\partial a^{(2)}} = a^{(2)} - y$
- $\frac{\partial a^{(2)}}{\partial z^{(2)}} = a^{(2)}(1 - a^{(2)})$
- $\frac{\partial z^{(2)}}{\partial a^{(1)}} = \theta^{(2)}$
- $\frac{\partial a^{(1)}}{\partial z^{(1)}} = a^{(1)}(1 - a^{(1)})$
- $\frac{\partial z^{(1)}}{\partial \theta^{(1)}} = \mathbf{X}_1$

Os primeiros termos são iguais aos dois primeiros termos do gradiente de $\theta^{(2)}$

Backward Pass

$$\frac{\partial J}{\partial \theta^{(1)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial \theta^{(1)}}$$

- $\frac{\partial J}{\partial a^{(2)}} = a^{(2)} - y$
- $\frac{\partial a^{(2)}}{\partial z^{(2)}} = a^{(2)}(1 - a^{(2)})$
- $\frac{\partial z^{(2)}}{\partial a^{(1)}} = \theta^{(2)}$
- $\frac{\partial a^{(1)}}{\partial z^{(1)}} = a^{(1)}(1 - a^{(1)})$
- $\frac{\partial z^{(1)}}{\partial \theta^{(1)}} = \mathrm{X}_1$

$$\frac{\partial z^{(2)}}{\partial a^{(1)}} = \frac{\partial \left(a^{(1)}\theta^{(2)}\right)}{\partial a^{(1)}} = \theta^{(2)}$$

Backward Pass

$$\frac{\partial J}{\partial \theta^{(1)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial \theta^{(1)}}$$

- $\frac{\partial J}{\partial a^{(2)}} = a^{(2)} - y$
- $\frac{\partial a^{(2)}}{\partial z^{(2)}} = a^{(2)}(1 - a^{(2)})$
- $\frac{\partial z^{(2)}}{\partial a^{(1)}} = \theta^{(2)}$
- $\frac{\partial a^{(1)}}{\partial z^{(1)}} = a^{(1)}(1 - a^{(1)})$
- $\frac{\partial z^{(1)}}{\partial \theta^{(1)}} = \mathbf{X}_1$

$$\begin{aligned}
\frac{\partial a^{(1)}}{\partial z^{(1)}} &= \frac{\partial \left(\frac{1}{1+e^{-z^{(1)}}}\right)}{\partial z^{(1)}} = \frac{\partial \left((1+e^{-z^{(1)}})^{-1}\right)}{\partial z^{(1)}} \\
&= (\cancel{-1})(1+e^{-z^{(1)}})^{-2} e^{-z^{(1)}} (\cancel{-1}) \\
&= \frac{1 + e^{-z^{(1)}} - 1}{(1+e^{-z^{(1)}})^2} \\
&= \frac{1}{1+e^{-z^{(1)}}} \left(\frac{1 + e^{-z^{(1)}}}{1+e^{-z^{(1)}}} - \frac{1}{1+e^{-z^{(1)}}} \right) \quad (\cancel{1+e^{-z^{(1)}}} \to 1) \\
&= \sigma(z^{(1)})(1 - \sigma(z^{(1)})) = a^{(1)}(1 - a^{(1)})
\end{aligned}$$

Backward Pass

$$\frac{\partial J}{\partial \theta^{(1)}} = \frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}} \frac{\partial z^{(1)}}{\partial \theta^{(1)}}$$

- $\frac{\partial J}{\partial a^{(2)}} = a^{(2)} - y$
- $\frac{\partial a^{(2)}}{\partial z^{(2)}} = a^{(2)}(1 - a^{(2)})$
- $\frac{\partial z^{(2)}}{\partial a^{(1)}} = \theta^{(2)}$
- $\frac{\partial a^{(1)}}{\partial z^{(1)}} = a^{(1)}(1 - a^{(1)})$
- $\frac{\partial z^{(1)}}{\partial \theta^{(1)}} = \mathbf{X}_1$

$$\begin{aligned} \frac{\partial z^{(1)}}{\partial \theta^{(1)}} &= \frac{\partial \left(X_1 \theta^{(1)} \right)}{\partial \theta^{(1)}} \\ &= X_1 \end{aligned}$$

Backward Pass

Vamos Computar o Gradiente

$$\frac{\partial J}{\partial \theta^{(2)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot a^{(1)}$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.3775 = \mathbf{-0.04237412}$$

$$\frac{\partial J}{\partial \theta^{(1)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot \theta^{(2)} \cdot a^{(1)}(1 - a^{(1)}) \cdot X_1$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.5 \cdot 0.3775(1 - 0.3775) \cdot 1 = \mathbf{-0.01318894}$$

Backward Pass

Vamos Computar o Gradiente

$$\frac{\partial J}{\partial \theta^{(2)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot a^{(1)}$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.3775 = \mathbf{-0.04237412}$$

$$\frac{\partial J}{\partial \theta^{(1)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot \theta^{(2)} \cdot a^{(1)}(1 - a^{(1)}) \cdot X_1$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.5 \cdot 0.3775(1 - 0.3775) \cdot 1 = \mathbf{-0.01318894}$$

Backward Pass

Vamos Computar o Gradiente

$$\frac{\partial J}{\partial \theta^{(2)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot a^{(1)}$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.3775 = \mathbf{-0.04237412}$$

$$\frac{\partial J}{\partial \theta^{(1)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot \theta^{(2)} \cdot a^{(1)}(1 - a^{(1)}) \cdot X_1$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.5 \cdot 0.3775(1 - 0.3775) \cdot 1 = \mathbf{-0.01318894}$$

Backward Pass

Vamos Computar o Gradiente

$$\frac{\partial J}{\partial \theta^{(2)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot a^{(1)}$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.3775 = \mathbf{-0.04237412}$$

$$\frac{\partial J}{\partial \theta^{(1)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot \theta^{(2)} \cdot a^{(1)}(1 - a^{(1)}) \cdot X_1$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.5 \cdot 0.3775(1 - 0.3775) \cdot 1 = \mathbf{-0.01318894}$$

Backward Pass

Vamos Computar o Gradiente

$$\frac{\partial J}{\partial \theta^{(2)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot a^{(1)}$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.3775 = \mathbf{-0.04237412}$$

$$\frac{\partial J}{\partial \theta^{(1)}} = \underbrace{(a^{(2)} - y) \cdot a^{(2)}(1 - a^{(2)})}_{\delta^{(2)}} \cdot \theta^{(2)} \cdot a^{(1)}(1 - a^{(1)}) \cdot X_1$$

$$= (0.5470 - 1) \cdot 0.5470(1 - 0.5470) \cdot 0.5 \cdot 0.3775(1 - 0.3775) \cdot 1 = \mathbf{-0.01318894}$$

Resumindo

No *Backward Pass* calculamos os gradientes da função de custo em relação a todos os parâmetros treináveis

Descida de Gradiente

- Usando $\eta = 0.1$: $\theta_t = \theta_{t-1} - \eta \nabla J$
- $\theta_t^{(2)} = 0.5 - 0.1(-0.042374) =$ **0.5042374**
- $\theta_t^{(1)} = -0.5 - 0.1(-0.013189) =$ **-0.4986811**

Descida de Gradiente

- Usando $\eta = 0.1$: $\theta_t = \theta_{t-1} - \eta \nabla J$
- $\theta_t^{(2)} = 0.5 - 0.1(-0.042374) =$ **0.5042374**
- $\theta_t^{(1)} = -0.5 - 0.1(-0.013189) =$ **-0.4986811**

Resumindo

Na *Descida de Gradiente* usamos o gradiente para atualizar os parâmetros treináveis

Resultado da Atualização

- Após a atualização, os pesos da rede são:
  - $\theta^{(1)} = -0.4986811$
  - $\theta^{(2)} = 0.5042374$
- O novo valor de $a^{(2)}$ é:

$$a^{(2)} = \sigma(\sigma(X_1 \cdot \theta^{(1)}) \cdot \theta^{(2)}) = \sigma(\sigma(1 \cdot (-0.4986811)) \cdot 0.5042374) = 0.5474$$

- O novo valor de $J(\theta)$ é:

$$J(\theta) = \frac{1}{2}(y - a^{(2)})^2 = \frac{1}{2}(1 - 0.5474)^2 = 0.1024234$$

Resultado da Atualização

- Após a atualização, os pesos da rede são:
  - $\theta^{(1)} = -0.4986811$
  - $\theta^{(2)} = 0.5042374$
- O novo valor de $a^{(2)}$ é:

$$a^{(2)} = \sigma(\sigma(X_1 \cdot \theta^{(1)}) \cdot \theta^{(2)}) = \sigma(\sigma(1 \cdot (-0.4986811)) \cdot 0.5042374) = 0.5474$$

- O novo valor de $J(\theta)$ é:

$$J(\theta) = \frac{1}{2}(y - a^{(2)})^2 = \frac{1}{2}(1 - 0.5474)^2 = 0.1024234$$

- O novo valor de $J(\theta)$ é menor que o anterior 0.1026045, indicando que o erro do MLP é menor

Comparando com PyTorch

$$\frac{\partial J}{\partial \boldsymbol{\theta}^{(1)}} = -0.01318894$$
$$\frac{\partial J}{\partial \boldsymbol{\theta}^{(2)}} = -0.04237412$$
$$\theta_t^{(2)} = 0.5042374$$
$$\theta_t^{(1)} = -0.4986811$$

```
optimizer = SGD(model.parameters(),
                lr=0.1)
x = torch.tensor(1.0)
y = torch.tensor(1.0)
model.train()
optimizer.zero_grad()
y_hat = model(x)
loss = (y - y_hat)**2/2
loss.backward()
print(f'Grad: {model.theta_2.grad}, {model.theta_1.grad}')
optimizer.step()
print(f'Pesos: {model.theta_2.data}, {model.theta_1.data}')
# Grad: -0.04237288236618042, -0.01318769808858633
# Pesos: 0.5042372941970825, -0.4986812174320221
```

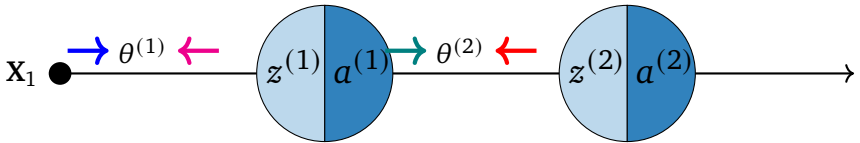
Visão Geral

1. Introdução

2. Treinamento de um MLP
   2.1 Forward Pass
   2.2 Função de Custo
   2.3 Backward Pass
   2.4 Descida de Gradiente

3. Vetorização do Treinamento

Generalizando o Treinamento



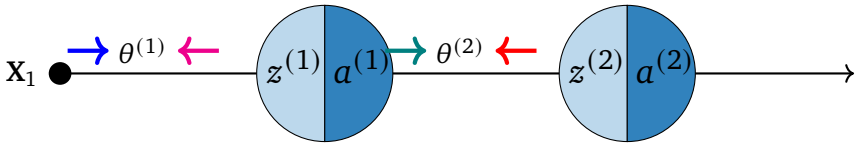
Repare que:

- Ambas as derivadas são computadas com um termo $\delta^{(l)}$ multiplicado pela entrada da camada $\frac{\partial z^{(l)}}{\partial \theta^{(l)}}$

$$\frac{\partial J}{\partial \theta^{(2)}} = \underbrace{\frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}}}_{\delta^{(2)}} \frac{\partial z^{(2)}}{\partial \theta^{(2)}}$$

$$\frac{\partial J}{\partial \theta^{(1)}} = \underbrace{\underbrace{\frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}}}_{\delta^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}}}_{\delta^{(1)}} \frac{\partial z^{(1)}}{\partial \theta^{(1)}}$$

Generalizando o Treinamento



Repare que:

- Ambas as derivadas são computadas com um termo $\delta^{(l)}$ multiplicado pela entrada da camada $\frac{\partial z^{(l)}}{\partial \theta^{(l)}}$
- $\delta^{(1)}$ pode ser obtido a partir de $\delta^{(2)}$

$$\frac{\partial J}{\partial \theta^{(2)}} = \underbrace{\frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}}}_{\delta^{(2)}} \frac{\partial z^{(2)}}{\partial \theta^{(2)}}$$

$$\frac{\partial J}{\partial \theta^{(1)}} = \underbrace{\underbrace{\frac{\partial J}{\partial a^{(2)}} \frac{\partial a^{(2)}}{\partial z^{(2)}}}_{\delta^{(2)}} \frac{\partial z^{(2)}}{\partial a^{(1)}} \frac{\partial a^{(1)}}{\partial z^{(1)}}}_{\delta^{(1)}} \frac{\partial z^{(1)}}{\partial \theta^{(1)}}$$

Generalizando o Treinamento

O gradiente em cada camada pode ser dividido em dois componentes:

- **Delta da Camada:** $\delta^{(l)} = \frac{\partial J}{\partial z^{(l)}}$
- **Ativação da Camada Anterior:** $a^{(l-1)}$

Backpropagation:

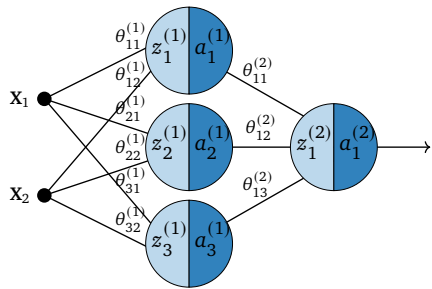
1. **Erro na Camada de Saída:** $\delta^{(L)} = \frac{\partial J}{\partial a^{(L)}} \odot \frac{\partial a^{(L)}}{\partial z^{(L)}}$
2. **Propagação do Erro:** $\delta^{(l)} = (\delta^{(l+1)}\theta^{(l+1)}) \odot a'^{(l)}$
3. **Gradiente do Parâmetros:** $\frac{\partial J}{\partial \theta^{(i)}} = \delta^{(i)^\top} a^{(i-1)}$

Exemplo Vetorizado

- Dataset (*Batch Size* = 2)

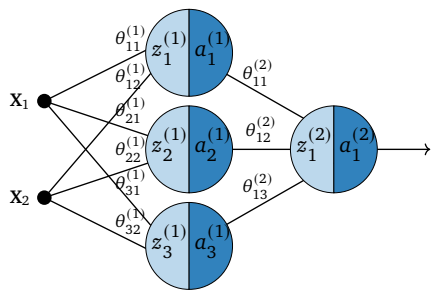
$$X = \begin{bmatrix} 0.5 & 0.1 \\ 0.2 & 0.6 \end{bmatrix}, \quad y = \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}$$

- Arquitetura
  - Camada de Entrada: 2 unidades
  - Camada Oculta 1: 3 unidades
  - Camada de Saída: 1 unidade
- Pesos Iniciais
  - Camada 1: $\theta^{(1)} = \begin{bmatrix} 0.5 & 0.2 \\ 0.6 & -0.1 \\ -0.4 & -0.3 \end{bmatrix}$
  - Camada 2: $\theta^{(2)} = \begin{bmatrix} 0.7 & -0.1 & 0.2 \end{bmatrix}$



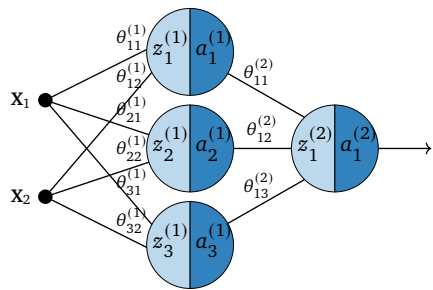
Exemplo Vetorizado: Forward Pass

$$
\begin{aligned}
Z^{(1)} &= X \cdot {\theta^{(1)}}^{\top} \\
&= \begin{bmatrix} 0.5 & 0.1 \\ 0.2 & 0.6 \end{bmatrix}_{2\times 2} \cdot \begin{bmatrix} 0.5 & 0.6 & -0.4 \\ 0.2 & -0.1 & -0.3 \end{bmatrix}_{2\times 3} \\
&= \begin{bmatrix} 0.27 & 0.29 & -0.23 \\ 0.22 & 0.06 & -0.26 \end{bmatrix}_{2\times 3} \\
A^{(1)} &= \sigma(Z^{(1)}) \\
&= \sigma\left(\begin{bmatrix} 0.27 & 0.29 & -0.23 \\ 0.22 & 0.06 & -0.26 \end{bmatrix}_{2\times 3} \right) \\
&= \begin{bmatrix} 0.5671 & 0.5720 & 0.4428 \\ 0.5548 & 0.5150 & 0.4354 \end{bmatrix}_{2\times 3}
\end{aligned}
$$



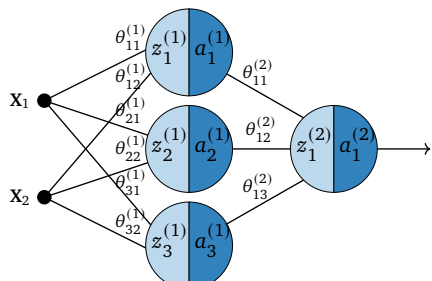
Exemplo Vetorizado: Forward Pass

$$
\begin{aligned}
Z^{(2)} &= A^{(1)} \cdot \theta^{(2)^T} \\
&= \begin{bmatrix} 0.5671 & 0.5720 & 0.4428 \\ 0.5548 & 0.5150 & 0.4354 \end{bmatrix}_{2\times 3} \cdot \begin{bmatrix} 0.7 \\ -0.1 \\ 0.2 \end{bmatrix}_{3\times 1} \\
&= \begin{bmatrix} 0.4283 \\ 0.4239 \end{bmatrix}_{2\times 1} \\
A^{(2)} &= \hat{y} = \sigma(Z^{(2)}) \\
&= \sigma\left(\begin{bmatrix} 0.4283 \\ 0.4239 \end{bmatrix}_{2\times 1} \right) \\
&= \begin{bmatrix} 0.6055 \\ 0.6044 \end{bmatrix}_{2\times 1}
\end{aligned}
$$



Exemplo Vetorizado: Cálculo da *Loss*

$$
\begin{aligned}
J &= \frac{1}{2}\sum (y - \hat{y})^2 \\
&= \frac{1}{2}\sum \left(\begin{bmatrix} 0.6055 \\ 0.6044 \end{bmatrix}_{2\times 1} - \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix}_{2\times 1} \right)^2 \\
&= 0.01914
\end{aligned}
$$



Exemplo Vetorizado: *Backward Pass*

$$
\begin{aligned}
\delta^{(2)} &= \left(\begin{bmatrix} 0.6055 \\ 0.6044 \end{bmatrix} - \begin{bmatrix} 0.6 \\ 0.8 \end{bmatrix} \right) \odot \begin{bmatrix} 0.2389 \\ 0.2391 \end{bmatrix} \\
&= \begin{bmatrix} 0.0055 \\ -0.1956 \end{bmatrix} \odot \begin{bmatrix} 0.2389 \\ 0.2391 \end{bmatrix} \\
&= \begin{bmatrix} 0.00131 \\ -0.04676 \end{bmatrix} \\
\delta^{(1)} &= (\delta^{(2)} \cdot \theta^{(2)}) \odot \sigma'(Z^{(1)}) \\
&= \left(\begin{bmatrix} 0.00131 \\ -0.04676 \end{bmatrix} \cdot \begin{bmatrix} 0.7 & -0.1 & 0.2 \end{bmatrix} \right) \odot \sigma' \left(\begin{bmatrix} 0.27 & 0.29 & -0.23 \\ 0.22 & 0.06 & -0.26 \end{bmatrix} \right) \\
&= \begin{bmatrix} 0.00022 & -0.00003 & 0.00006 \\ -0.00809 & 0.00117 & -0.00230 \end{bmatrix}
\end{aligned}
$$

Exemplo Vetorizado: *Backward Pass*

$$\frac{\partial J}{\partial \theta^{(2)}} = {\delta^{(2)}}^T \cdot A^{(1)}$$

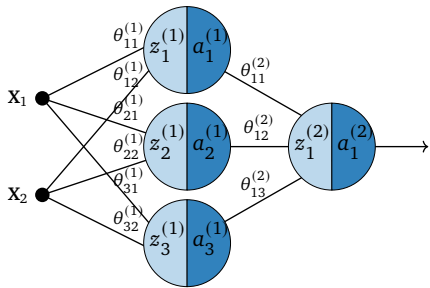
$$= \begin{bmatrix} 0.00131 & -0.04676 \end{bmatrix} \cdot \begin{bmatrix} 0.5671 & 0.5720 & 0.4428 \\ 0.5548 & 0.5150 & 0.4354 \end{bmatrix}$$

$$= \begin{bmatrix} -0.0252 & -0.0233 & -0.0198 \end{bmatrix}$$

$$\frac{\partial J}{\partial \theta^{(1)}} = {\delta^{(1)}}^T \cdot X$$

$$= \begin{bmatrix} 0.00022 & -0.00809 \\ -0.00003 & 0.00117 \\ 0.00006 & -0.00230 \end{bmatrix} \cdot \begin{bmatrix} 0.5 & 0.1 \\ 0.2 & 0.6 \end{bmatrix}$$

$$= \begin{bmatrix} -0.00150 & -0.00483 \\ 0.00022 & 0.00070 \\ -0.00043 & -0.00137 \end{bmatrix}$$



Exemplo Vetorizado: *Optimizer*

Atualização dos Pesos ($\eta = 0.1$):

$$\theta_t^{(l)} = \theta_{t-1}^{(l)} - \eta \nabla_\theta J$$

Novos Pesos da Camada 2:

$$\begin{aligned}\theta_{new}^{(2)} &= \begin{bmatrix}0.7 & -0.1 & 0.2\end{bmatrix} - 0.1 \cdot \begin{bmatrix}-0.0252 & -0.0233 & -0.0198\end{bmatrix} \\ &= \begin{bmatrix}0.7025 & -0.0977 & 0.2020\end{bmatrix}\end{aligned}$$

Novos Pesos da Camada 1:

$$\begin{aligned}\theta_{new}^{(1)} &= \begin{bmatrix}0.5 & 0.2 \\ 0.6 & -0.1 \\ -0.4 & -0.3\end{bmatrix} - 0.1 \cdot \begin{bmatrix}-0.00150 & -0.00483 \\ 0.00022 & 0.00070 \\ -0.00043 & -0.00137\end{bmatrix} \\ &= \begin{bmatrix}0.5002 & 0.2005 \\ 0.6000 & -0.1001 \\ -0.4000 & -0.2999\end{bmatrix}\end{aligned}$$

Resumindo

- Podemos treinar redes neurais de múltiplas camadas usando descida de gradiente.

- O processo de treinamento consiste em repetir 4 etapas:
  1. Computar todas as saídas da rede (*Forward Pass*)
  2. Computar o quão diferente as saídas são em relação a variável alvo (*Loss Function*)
  3. Computar o gradiente do erro em relação aos parâmetros da rede (*Backward Pass*)
  4. Atualizar os pesos (*Optimizer*)

- Todo o treinamento pode ser implementado de forma vetorizada.

Leituras Recomendadas

- Primeira publicação do Backpropagation: David E Rumelhart, Geoffrey E Hinton e Ronald J Williams. “Learning Internal Representations by Error Propagation, Parallel Distributed Processing, Explorations in the Microstructure of Cognition, ed. DE Rumelhart and J. McClelland. Vol. 1. 1986”. Em: Biometrika 71 (1986), pp. 599–607
- Capítulo 2: Michael A Nielsen. Neural networks and deep learning. Vol. 25. Determination press San Francisco, CA, USA, 2015

Referências

[1] George Cybenko. “Approximation by superpositions of a sigmoidal function”. Em: Mathematics of control, signals and systems 2.4 (1989), pp. 303–314.

[2] Kurt Hornik, Maxwell Stinchcombe e Halbert White. “Multilayer feedforward networks are universal approximators”. Em: Neural networks 2.5 (1989), pp. 359–366.

[3] Marvin Minsky e Seymour Papert. “Perceptron: an introduction to computational geometry”. Em: The MIT Press, Cambridge, expanded edition 19.88 (1969), p. 2.

[4] Michael A Nielsen. Neural networks and deep learning. Vol. 25. Determination press San Francisco, CA, USA, 2015.

[5] David E Rumelhart, Geoffrey E Hinton e Ronald J Williams. “Learning Internal Representations by Error Propagation, Parallel Distributed Processing, Explorations in the Microstructure of Cognition, ed. DE Rumelhart and J. McClelland. Vol. 1. 1986”. Em: Biometrika 71 (1986), pp. 599–607.