

# Treinamento de Redes Neurais

Eduardo Adame

Redes Neurais

20 de agosto de 2025



# Como Atualizar os Pesos?

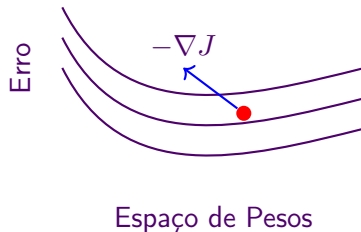


## Após calcular os gradientes:

- Sabemos a derivada para cada peso
- Como exatamente atualizamos?
- Com que frequência?
  - ▶ Após cada exemplo?
  - ▶ Após todos os dados?
  - ▶ Algo intermediário?

## Regra de Atualização:

$$W_{novo} = W_{antigo} - \alpha \cdot \frac{\partial J}{\partial W}$$



# Descida de Gradiente Clássica (Batch)



## Abordagem Tradicional:

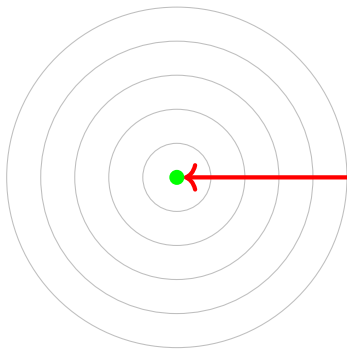
- Calcular gradiente para **todo o conjunto**
- Dar um passo na direção oposta
- Repetir até convergência

## Vantagens:

- + Cada passo usa toda informação
- + Convergência mais estável

## Desvantagens:

- Muito lento para datasets grandes
- Pode ficar preso em mínimos locais



Convergência Suave

# Descida de Gradiente Estocástica (SGD)



## Abordagem Online:

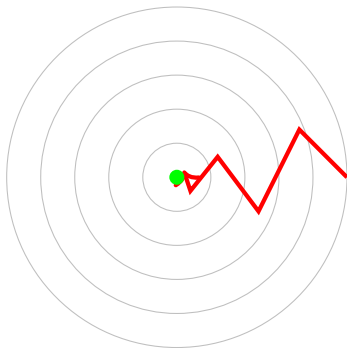
- Calcular gradiente para **um exemplo**
- Atualizar imediatamente
- Mais passos, menos informados

## Vantagens:

- + Muito mais rápido
- + Pode escapar de mínimos locais
- + Funciona bem online

## Desvantagens:

- Convergência ruidosa
- Precisa de taxa de aprendizado menor



Convergência Ruidosa

# Mini-batch: O Melhor dos Dois Mundos

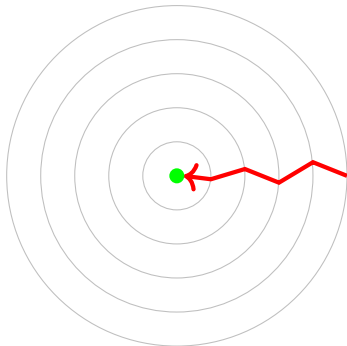


## Abordagem Híbrida:

- Usar **pequenos lotes** (16, 32, 64...)
- Equilíbrio entre velocidade e estabilidade
- Padrão na prática moderna

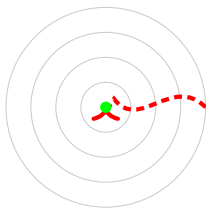
## Benefícios:

- ✓ Aproveita paralelização (GPU)
- ✓ Reduz variância do gradiente
- ✓ Ainda permite escapar de mínimos
- ✓ Boa relação custo-benefício

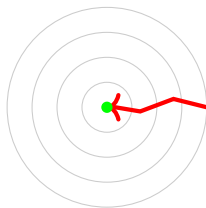


Convergência Balanceada

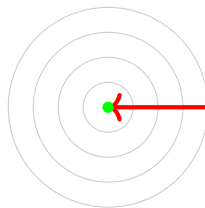
# Comparação das Estratégias de Batch



**SGD**  
(Batch = 1)



**Mini-batch**  
(Batch = 32)



**Batch Completo**  
(Batch = N)



# Terminologia: Época e Iteração



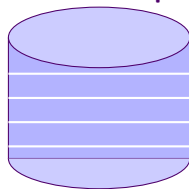
## Época (Epoch):

- Uma passada completa pelos dados
- Todos exemplos vistos uma vez
- Métrica comum de progresso

## Passos por Época:

- **Batch completo**: 1 passo
- **SGD**: N passos
- **Mini-batch**:  $N/\text{batch\_size}$  passos

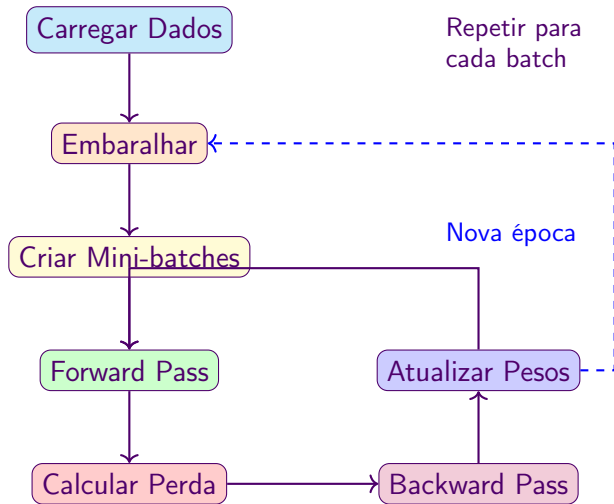
## Dataset Completo



Batch 1  
Batch 2  
Batch 3  
Batch 4  
Batch 5

**Dica:** Embaralhar os dados após cada época melhora a convergência!

# Fluxo de Treinamento: Visão Prática





## Listing 1: Estrutura Típica de Código

```
1 # 1. Construir o modelo
2 model = Sequential()
3 model.add(Dense(64, activation='relu'))
4 model.add(Dense(32, activation='relu'))
5 model.add(Dense(10, activation='softmax'))
6 # 2. Compilar
7 model.compile(
8     optimizer='adam',
9     loss='categorical_crossentropy',
10    metrics=['accuracy']
11 )
12 # 3. Treinar
13 history = model.fit(
14     X_train, y_train,
15     batch_size=32,      # Mini-batch
16     epochs=50,         # Épocas
17     validation_split=0.2
18 )
```



## Parâmetros Importantes:

- **batch\_size**: Tamanho do mini-batch
- **epochs**: Número de épocas
- **shuffle**: Embaralhar dados (padrão: True)
- **validation\_split**: Proporção para validação

## Valores Típicos:

- Batch size: 16, 32, 64, 128
- Épocas: 10-100 (depende do problema)
- Learning rate: 0.001 (Adam), 0.01 (SGD)



## Por que normalizar?

- Acelera convergência
- Evita saturação de neurônios
- Estabiliza o treinamento
- Permite learning rates maiores

## Métodos Comuns:

- **Min-Max:**  $[0, 1]$  ou  $[-1, 1]$
- **Padronização:** média 0, desvio 1
- **Normalização L2:** vetores unitários

## Fórmulas:

Min-Max para  $[0, 1]$ :

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Padronização (Z-score):

$$x' = \frac{x - \mu}{\sigma}$$

# Ativação Softmax para Classificação Multiclasse



## Extensão da Sigmoid:

- Saída: vetor de probabilidades
- Soma sempre igual a 1
- Interpretação probabilística

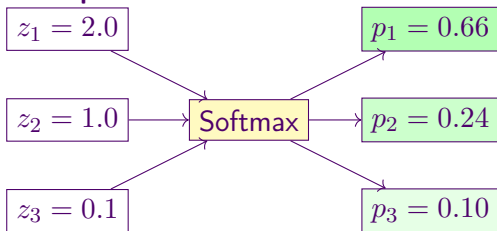
## Fórmula:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^K e^{z_k}}$$

## Com Cross-Entropy:

$$J = - \sum_{i=1}^n y_i \log(\hat{y}_i)$$

## Exemplo Visual:



Classe prevista: Argmax = Classe 1



## Decisões Importantes:

- **Estratégia de batch**
  - ▶ Mini-batch (padrão)
  - ▶ Tamanho: 32-128
- **Número de épocas**
  - ▶ Monitorar validação
  - ▶ Early stopping
- **Normalização de entrada**
  - ▶ Sempre normalizar!
  - ▶ Método depende dos dados

## Boas Práticas:

- ✓ Embaralhar dados a cada época
- ✓ Usar validação para monitorar
- ✓ Começar com learning rate pequeno
- ✓ Visualizar curvas de aprendizado
- ✓ Salvar checkpoints do modelo

## Próximos Tópicos:

- Otimizadores avançados (Adam, RMSprop)
- Regularização (Dropout, L2)
- Learning rate scheduling

# Obrigado!

Dúvidas?

Vamos para as implementações!