Treinamento de Redes Neurais

Eduardo Adame

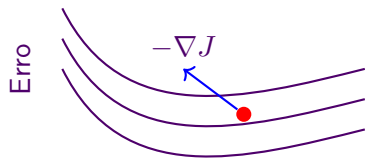
Redes Neurais

03 de setembro 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:

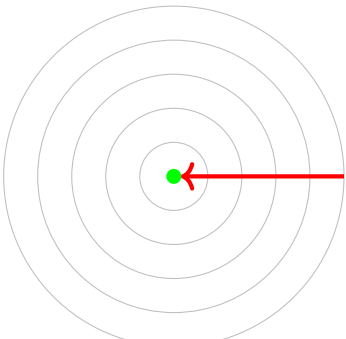
- Calcular gradiente para todo o conjunto
- Dar um passo na direção oposta
- Repetir até convergência

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:

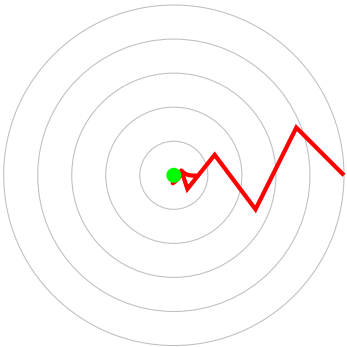
- Calcular gradiente para um exemplo
- Atualizar imediatamente
- Mais passos, menos informados

Vantagens:

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

Desvantagens:

– Convergência ruidosa
– Precisa de taxa de aprendizado menor



Convergência Ruidosa

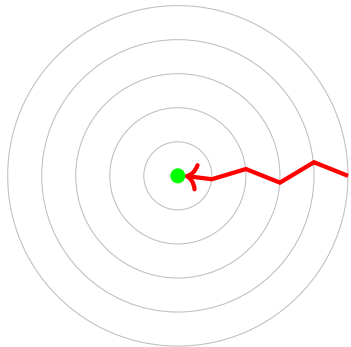
Mini-batch: O Melhor dos Dois Mundos

Abordagem Híbrida:

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

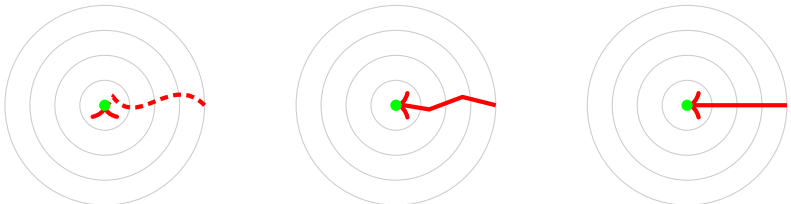
Benefícios:

- ✓ 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)

Rápido

Preciso

Terminologia: Época e Iteração

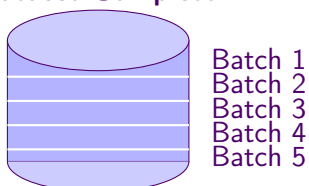
Época (Epoch):

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

Passos por Época:

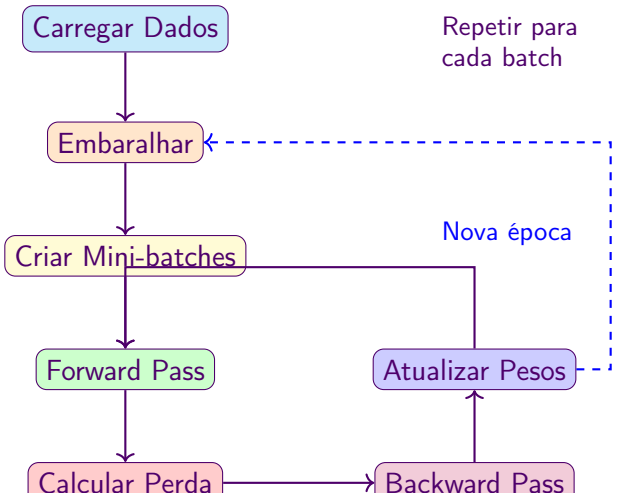
- Batch completo: 1 passo
- SGD: N passos
- Mini-batch: N/batch_size passos

Dataset Completo



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

Fluxo de Treinamento: Visão Prática



Implementação em Keras/TensorFlow

Listing 1: Estrutura Típica de Código

```
# 1. Construir o modelo
model = Sequential()
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(10, activation='softmax'))
# 2. Compilar
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
# 3. Treinar
history = model.fit(
    X_train, y_train,
    batch_size=32,      # Mini-batch
    epochs=50,          # Épocas
    validation_split=0.2
)
```

Implementação em Keras/TensorFlow

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)

Normalização de Entrada

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

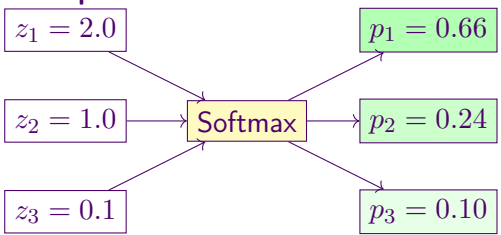
Fórmula:

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

Com Cross-Entropy:

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

Exemplo Visual:



Classe prevista: Argmax = Classe 1

Resumo: Detalhes Práticos do Treinamento

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!