# Regularização e Dropout em Redes Neurais

Eduardo Adame

**Redes Neurais**

10 de setembro de 2025

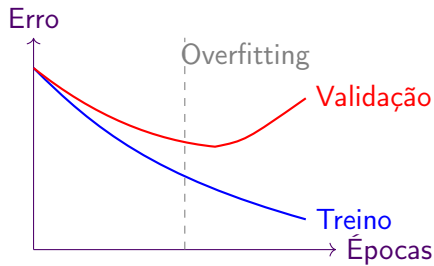
# O Desafio do Overfitting


**O que é Overfitting?**

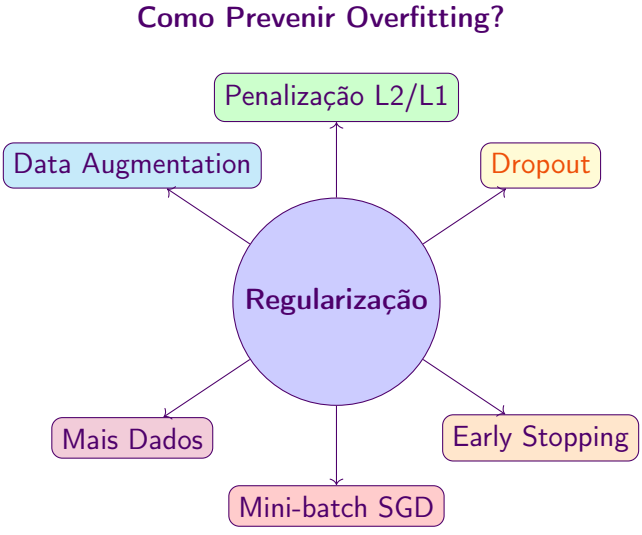
- Modelo memoriza dados de treino
- Não generaliza para novos dados
- Performance ruim em validação/teste

**Sinais de Overfitting:**

- ▷ Erro de treino ↓ muito baixo
- ▷ Erro de validação ↑ alto
- ▷ Grande diferença entre os dois



# Estratégias de Regularização



# Regularização com Penalização de Pesos


**Ideia Principal:**

- Adicionar termo de penalização à função de custo
- Penalizar pesos grandes
- Forçar modelo mais simples

**Função de Custo Regularizada:**

$$J_{reg} = J_{original} + \lambda \cdot R(W)$$

Onde:

- $\lambda$: força da regularização
- $R(W)$: termo de regularização

**Tipos de Regularização:**

L2 (Ridge):

$$R(W) = \sum_{i,j} W_{ij}^2$$

+ Pesos pequenos
+ Suave

L1 (Lasso):

$$R(W) = \sum_{i,j} |W_{ij}|$$

+ Esparsidade
+ Seleção de features

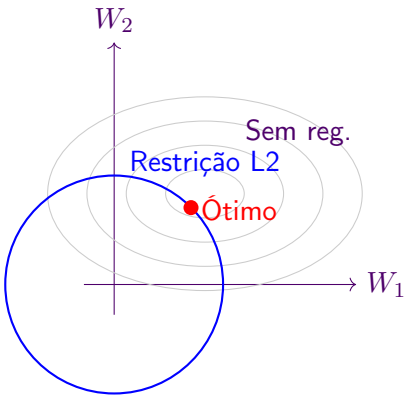
# Efeito da Regularização L2

**Atualização com L2:**

$$W_{novo} = W_{antigo} - \alpha \left( \frac{\partial J}{\partial W} + 2\lambda W_{antigo} \right)$$

**Interpretação:**

- "Weight decay" (decaimento de peso)
- Pesos diminuem a cada iteração
- Previne valores extremos



# Dropout: Desligando Neurônios Aleatoriamente

**Conceito do Dropout:**

- Durante treinamento:
  - Desligar neurônios aleatoriamente
  - Probabilidade $p$ de manter ativo
  - Diferente a cada batch
- Durante teste:
  - Todos neurônios ativos
  - Pesos multiplicados por $p$

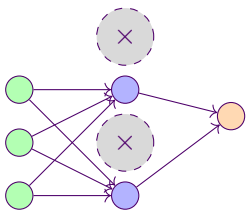
**Por que funciona?**

✓ Previne co-adaptação

✓ Como treinar ensemble

✓ Força redundância

**Treinamento**



p = 0.5

# Implementação do Dropout

**Durante o Treinamento:**

1. Gerar máscara aleatória
2. Aplicar à camada
3. Forward e backward pass normal

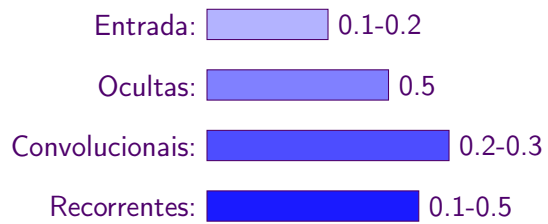
**Pseudocódigo:**

```
mask = random(0,1) >
dropout_rate
output = input * mask
output = output /
(1-dropout_rate)
```

**Inverted Dropout:**

- Escala durante treino
- Teste fica inalterado

**Valores Típicos de Dropout:**




Dica: Começar sem dropout, adicionar se houver overfitting

# Dropout vs Regularização L2

| Aspecto | Dropout | L2 |
|---|---|---|
| Tipo | Estocástico | Determinístico |
| Aplicação | Por camada | Global |
| Interpretação | Ensemble | Weight decay |
| Hiperparâmetros | Taxa por camada | $\lambda$ único |
| Tempo de treino | Mais lento | Normal |
| Teste | Precisa ajuste | Direto |

**Quando usar Dropout:**

- Redes profundas
- Muitos parâmetros
- Dados limitados

**Quando usar L2:**

- Sempre (baseline)
- Redes menores
- Combinado com dropout

# Early Stopping

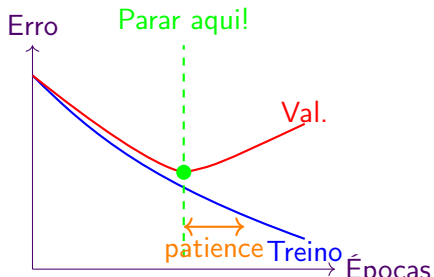
**Conceito:**

- Parar treinamento antes da convergência
- Monitorar erro de validação
- Guardar melhor modelo

**Estratégia Prática:**

1. Definir "patience"(paciência)
2. Se validação não melhora por N épocas
3. Restaurar melhor checkpoint
4. Parar treinamento

Vantagem: Simples e efetivo!



# Implementação em Keras

## Listing 1: Exemplo Completo com Regularização

```python
from tensorflow.keras import layers, regularizers

model = Sequential([
    # Camada com L2 e Dropout
    layers.Dense(128,
                 activation='relu',
                 kernel_regularizer=regularizers.l2(0.01)),
    layers.Dropout(0.5),

    # Segunda camada oculta
    layers.Dense(64,
                 activation='relu',
                 kernel_regularizer=regularizers.l2(0.01)),
    layers.Dropout(0.3),

    # Saída
    layers.Dense(10, activation='softmax')
])
```

# Implementação em Keras

## Listing 2: Exemplo Completo com Regularização

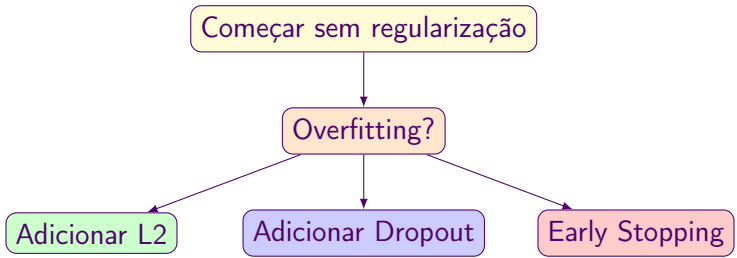
```python
# Compilar
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(learning_rate=0.01),
              metrics=['accuracy'])

# Early stopping
early_stop = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

model.fit(X_train, y_train,
          validation_split=0.2,
          callbacks=[early_stop])
```

# Combinando Técnicas de Regularização

**Estratégia Recomendada:**




Dicas Práticas:

- Começar com early stopping (sempre!)
- L2 com $\lambda$ pequeno (0.001 a 0.01)
- Dropout gradual ($0.2 \rightarrow 0.5$)

# Resumo: Regularização e Dropout


## Conceitos Principais:

- Overfitting: memorização vs generalização
- Regularização L2/L1: penalizar complexidade
- Dropout: desativar neurônios
- Early Stopping: parar no momento certo

## Boas Práticas:

- ✓ Sempre usar validação
- ✓ Começar simples
- ✓ Adicionar regularização gradualmente
- ✓ Monitorar curvas de aprendizado

## Checklist de Regularização:

1. Dividir dados (treino/val/teste)
2. Treinar modelo base
3. Identificar overfitting
4. Aplicar técnicas:
   - Early stopping
   - L2 regularization
   - Dropout (se necessário)
5. Ajustar hiperparâmetros
6. Validar no conjunto de teste

Lembre-se: Regularização é essencial para modelos que generalizam bem!

# Obrigado!

Dúvidas?

Próximo: Introdução às CNNs