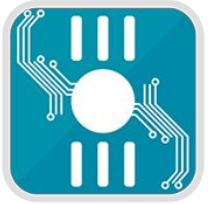


Descidas de gradiente (batch, mini-batch, estocástica), Overfitting e Underfitting (Viés e Variância) e Técnicas de Regularização de Modelos

Filipe Pinheiro e Guilherme Matos

Gradiente Batch, Estocástico e Mini-Batch

Sumário

- Gradiente Descendente (Batch, Estocástico e Mini-Batch);
- Underfitting, Overfitting e Goodfitting;
- Viés e Variância;
- Técnicas de Regularização (Dropout, Data augmentation, L2, L1 e Early Stopping).

O que são métodos de otimização?

Exemplos:

- Gradiente Descendente Batch;
- Gradiente Descendente Estocástico;
- Gradiente Descendente Mini-Batch;
- …

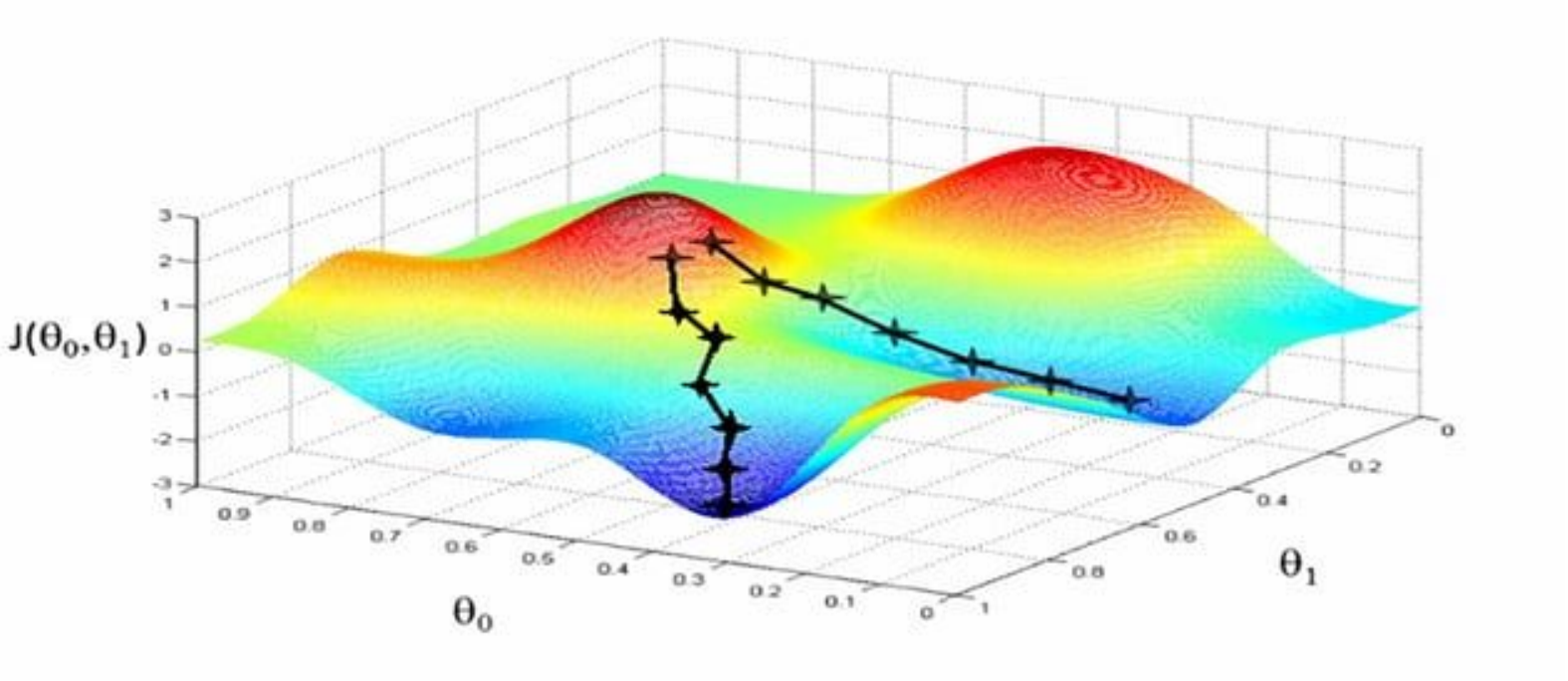
Gradiente Descendente

- Método de otimização iterativo;
- Definida por:

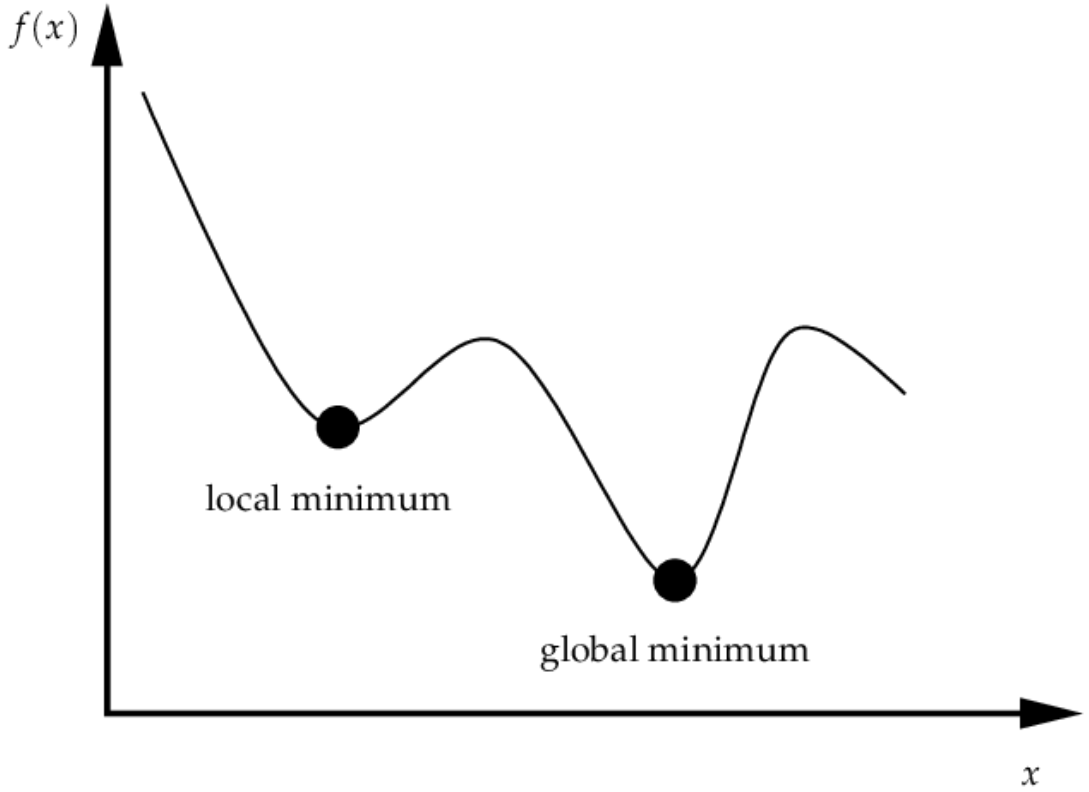
$$\theta := \theta - \alpha \cdot \nabla J(\theta)$$

- θ: parâmetro a ajustar
- α: taxa de aprendizado
- ∇J(θ): gradiente da função de custo em relação a θ

Gradiente Descendente



Gradiente Descendente



Gradiente Descendente Batch

- Calcula o gradiente com todos os dados de uma vez;
- Batch = grupo de dados;
- “Você espera ver todos os dados antes de dar um passo”;
- Mais preciso;
- Mais lento.

Gradiente Descendente Batch

- Demonstração matemática (regressão):

$$\hat{y}^{(i)} = h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

- ŷ: predição
- θ: parâmetros
- x: característica passada
- i: exemplo

Gradiente Descendente Batch

- Função de custo (MSE):

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

- m: número de exemplos
- ŷ - y: erro das predições

Gradiente Descendente Batch

- Gradiente da função de custo:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right) x^{(i)}$$

Gradiente Descendente Batch

- Atualização dos parâmetros:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Gradiente Descendente Batch

- Exemplo numérico (1ª iteração):

Os dados a serem calculados são:

- $x = [1, 2, 3, 4]$
- $y = [2, 4, 6, 8]$

Onde, com a inicialização:

- $\theta_0 = 0$
- $\theta_1 = 0$
- $\alpha = 0.05$

Gradiente Descendente Batch

- Exemplo numérico (1ª iteração):

Inicialmente, calcular a predição para cada exemplo:

- $\hat{y}_1 = 0 + 0 \cdot 1 = 0$
- $\hat{y}_2 = 0 + 0 \cdot 2 = 0$
- $\hat{y}_3 = 0 + 0 \cdot 3 = 0$
- $\hat{y}_4 = 0 + 0 \cdot 4 = 0$

Gradiente Descendente Batch

- Exemplo numérico (1ª iteração):

Calculando o erro, sendo a diferença entre o valor predito e o valor real:

- $\hat{y}_1 - y_1 = 0 - 2 = -2$
- $\hat{y}_2 - y_2 = 0 - 4 = -4$
- $\hat{y}_3 - y_3 = 0 - 6 = -6$
- $\hat{y}_4 - y_4 = 0 - 8 = -8$

Gradiente Descendente Batch

- Exemplo numérico (1ª iteração):

Em seguida é realizado o cálculo do gradiente:

- $\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{4}(-2 + (-4) + (-6) + (-8)) = -5$
- $\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{4}(-2 \cdot 1 + (-4) \cdot 2 + (-6) \cdot 3 + (-8) \cdot 4) = -15$

Gradiente Descendente Batch

- Exemplo numérico (1ª iteração):

Atualização dos parâmetros:

- $\theta_0 = 0 - 0.05 \times (-5) = 0.25$
- $\theta_1 = 0 - 0.05 \times (-15) = 0.75$

Gradiente Descendente Batch

Fluxo de uma iteração:

- Definir os parâmetros theta;
- Calcular a predição para todo o conjunto;
- Calcular diferença entre predição e valor real;
- Calcular gradiente para todos os dados;
- Atualizar θ.

Gradiente Descendente Estocástico (SGD)

- O que é um processo estocástico?
- Calcula o gradiente usando apenas um dado por vez;
- “Como subir uma montanha no nevoeiro dando passos baseados só no que você vê”;
- Mais rápido, porém errático.

Gradiente Descendente Estocástico (SGD)

- Demonstração matemática (regressão):

$$\hat{y}^{(i)} = h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

- ŷ: predição
- θ: parâmetros
- x: característica passada
- i: exemplo

Gradiente Descendente Estocástico (SGD)

- Função de custo (MSE):

$$J(\theta) = \frac{1}{2}(\hat{y}_i - y_i)^2$$

- ŷ - y: erro da predição

Gradiente Descendente Estocástico (SGD)

- Gradiente da função de custo:

$$\frac{\partial J(\theta)}{\partial \theta_0} = (\hat{y}_i - y_i)$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = (\hat{y}_i - y_i)x_i$$

Gradiente Descendente Estocástico (SGD)

- Atualização dos parâmetros:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (1ª iteração):

Os dados a serem calculados são:

- $x = [1, 2, 3, 4]$
- $y = [2, 4, 6, 8]$

Onde, com a inicialização:

- $\theta_0 = 0$
- $\theta_1 = 0$
- $\alpha = 0.05$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (1ª iteração):

É calculado o gradiente somente para o primeiro exemplo:

- $\frac{\partial J(\theta)}{\partial \theta_0} = \hat{y}_i - y_i = 0 - 2 = -2$
- $\frac{\partial J(\theta)}{\partial \theta_1} = (\hat{y}_i - y_i) \cdot x_i = -2 \cdot 1 = -2$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (1ª iteração):

Atualização dos parâmetros conforme os gradientes para o primeiro exemplo:

- $\theta_0 := 0 - 0.05 \cdot (-2) = 0.1$
- $\theta_1 := 0 - 0.05 \cdot (-2) = 0.1$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (2ª iteração):

Os dados a serem calculados são:

- $x = [1, 2, 3, 4]$
- $y = [2, 4, 6, 8]$

Onde, com a inicialização:

- $\theta_0 = 0.1$
- $\theta_1 = 0.1$
- $\alpha = 0.05$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (2ª iteração):

É calculado o gradiente somente para o segundo exemplo:

- $\frac{\partial J(\theta)}{\partial \theta_0} = \hat{y}_i - y_i = 0.3 - 4 = -3.7$
- $\frac{\partial J(\theta)}{\partial \theta_1} = (\hat{y}_i - y_i) \cdot x_i = (0.3 - 4) \cdot 2 = -7.4$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (2ª iteração):

Atualização dos parâmetros conforme os gradientes para o segundo exemplo:

- $\theta_0 := 0.1 - 0.05 \cdot (-3.7) = 0.285$
- $\theta_1 := 0.1 - 0.05 \cdot (-7.4) = 0.47$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (3ª iteração):

Os dados a serem calculados são:

- $x = [1, 2, 3, 4]$
- $y = [2, 4, 6, 8]$

Onde, com a inicialização:

- $\theta_0 = 0.285$
- $\theta_1 = 0.47$
- $\alpha = 0.05$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (3ª iteração):

É calculado o gradiente somente para o terceiro exemplo:

- $\frac{\partial J(\theta)}{\partial \theta_0} = \hat{y}_i - y_i = 1.69 - 6 = -4.3$
- $\frac{\partial J(\theta)}{\partial \theta_1} = (\hat{y}_i - y_i) \cdot x_i = (1.69 - 6) \cdot 3 = -12.9$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (3ª iteração):

Atualização dos parâmetros conforme os gradientes para o terceiro exemplo:

- $\theta_0 = 0.285 - 0.05 \times (-4.3) = 0.5$
- $\theta_1 = 0.47 - 0.05 \times (-12.9) = 1.11$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (4ª iteração):

Os dados a serem calculados são:

- $x = [1, 2, 3, 4]$
- $y = [2, 4, 6, 8]$

Onde, com a inicialização:

- $\theta_0 = 0.5$
- $\theta_1 = 1.11$
- $\alpha = 0.05$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (4ª iteração):

É calculado o gradiente somente para o quarto exemplo:

- $\frac{\partial J(\theta)}{\partial \theta_0} = \hat{y}_i - y_i = 4.98 - 8 = -3.04$
- $\frac{\partial J(\theta)}{\partial \theta_1} = (\hat{y}_i - y_i) \cdot x_i = (4.98 - 8) \cdot 4 = -12.16$

Gradiente Descendente Estocástico (SGD)

- Exemplo numérico (4ª iteração):

Atualização dos parâmetros conforme os gradientes para o terceiro exemplo:

- $\theta_0 = 0.5 - 0.05 \times (-3.04) = 0.65$
- $\theta_1 = 1.11 - 0.05 \times (-12.16) = 1.72$

Gradiente Descendente Estocástico (SGD)

Fluxo de uma iteração:

- Definir os parâmetros theta;
- Calcular a predição somente para o exemplo atual;
- Calcular diferença entre predição e valor real;
- Calcular gradiente para o exemplo atual;
- Atualizar θ.

Gradiente Descendente Mini-Batch

- Calcula o gradiente com pequenos grupos de dados;
- Batch = grupo de dados;
- “O melhor dos dois mundos”;
- Compromisso entre precisão e velocidade;
- Depende do número de Batchs.

Gradiente Descendente Mini-Batch

- Demonstração matemática (regressão):

$$\hat{y}^{(i)} = h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$$

- ŷ: predição
- θ: parâmetros
- x: característica passada
- i: exemplo

Gradiente Descendente Mini-Batch

- Função de custo (MSE):

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)^2$$

- m: número de exemplos
- ŷ - y: erro das predições

Gradiente Descendente Mini-Batch

- Gradiente da função de custo:

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right)$$

$$\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right) x^{(i)}$$

Gradiente Descendente Mini-Batch

- Atualização dos parâmetros:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Gradiente Descendente Mini-Batch

- Exemplo numérico (1ª iteração):

Os dados a serem calculados são:

- $x = [1, 2, 3, 4]$
- $y = [2, 4, 6, 8]$

Onde, com a inicialização:

- $\theta_0 = 0$
- $\theta_1 = 0$
- $\alpha = 0.05$

Gradiente Descendente Mini-Batch

- Exemplo numérico (1ª iteração):

Inicialmente é feito a separação dos Batchs.
Aqui o tamanho do batch será 2:

- $x = [[1, 2], [3, 4]]$
- $y = [[2, 4], [6, 8]]$

Gradiente Descendente Mini-Batch

- Exemplo numérico (1ª iteração):

Para a primeira iteração, será feito o ŷ para x = [1, 2] e y = [2, 4], junto a seu erro de predição:

- $\hat{y}_1 = 0 + 0 \cdot 1 = 0$
- $\hat{y}_2 = 0 + 0 \cdot 2 = 0$

- $\hat{y}_1 - y_1 = 0 - 2 = -2$
- $\hat{y}_2 - y_2 = 0 - 4 = -4$

Gradiente Descendente Mini-Batch

- Exemplo numérico (1ª iteração):

Em seguida é realizado o cálculo do gradiente:

- $\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{2}(-2 + (-4)) = -3$
- $\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{2}(-2 \cdot 1 + (-4) \cdot 2) = -5$

Gradiente Descendente Mini-Batch

- Exemplo numérico (1ª iteração):

Atualização dos parâmetros:

- $\theta_0 := \theta_0 - 0.05 \cdot (-3) = 0.15$
- $\theta_1 := \theta_1 - 0.05 \cdot (-5) = 0.25$

Gradiente Descendente Mini-Batch

- Exemplo numérico (2ª iteração):

Relembrando os Batchs:

- $x = [[1, 2], [3, 4]]$
- $y = [[2, 4], [6, 8]]$

Onde, com os parâmetros:

- $\theta_0 = 0.15$
- $\theta_1 = 0.25$
- $\alpha = 0.05$

Gradiente Descendente Mini-Batch

- Exemplo numérico (2ª iteração):

Para a segunda iteração, será feito o ŷ para x = [3, 4] e y = [6, 8], junto a seu erro de predição:

- $\hat{y}_1 = 0.25 \cdot 3 + 0.15 = 0.9$
- $\hat{y}_2 = 0.25 \cdot 4 + 0.15 = 1.15$
- $\hat{y}_1 - y_1 = 0.9 - 6 = -5.1$
- $\hat{y}_2 - y_2 = 1.15 - 8 = -6.85$

Gradiente Descendente Mini-Batch

- Exemplo numérico (2ª iteração):

Em seguida é realizado o cálculo do gradiente:

- $\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{2}(-5.1 + (-6.85)) = -5.97$
- $\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{2}(-5.1 \cdot 3 + (-6.85) \cdot 4) = -21.35$

Gradiente Descendente Mini-Batch

- Exemplo numérico (2ª iteração):

Atualização dos parâmetros:

- $\theta_0 = 0.15 - 0.05 \times (-5.975) = 0.448$
- $\theta_1 = 0.25 - 0.05 \times (-21.35) = 1.31$

Gradiente Descendente Mini-Batch

Fluxo de uma iteração:

- Definir os parâmetros theta;
- Definir os Batchs e separar os dados de acordo;
- Calcular a predição para o Batch atual;
- Calcular diferença entre predição e valor real;
- Calcular gradiente para o Batch atual;
- Atualizar θ.

Comparativo

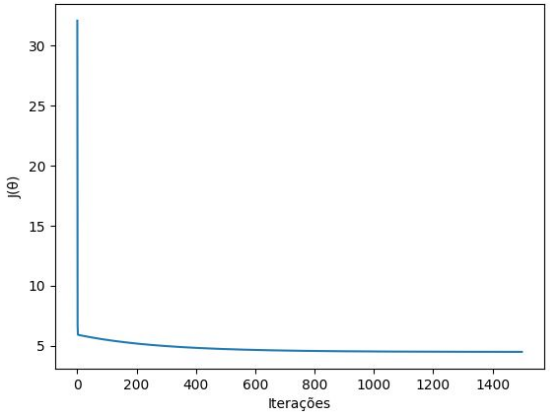
Algoritmo	m	Suporte out-of-core	n	Hiperparâmetros	Escalonamento Exigido
Gradiente Descendente Batch	Lento	Não	Rápido	2	Sim
Gradiente Descendente Estocástico	Rápido	Sim	Rápido	≥2	Sim
Gradiente Descendente Mini-Batch	Rápido	Sim	Rápido	≥2	Sim

Fonte: GÉRON, Aurélien., 2021. Página 102.

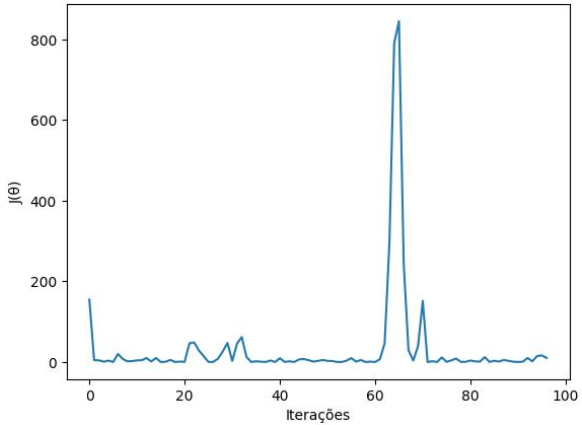
Onde m é o número de instâncias de treinamento e n é o número de características.

Partindo para o código…

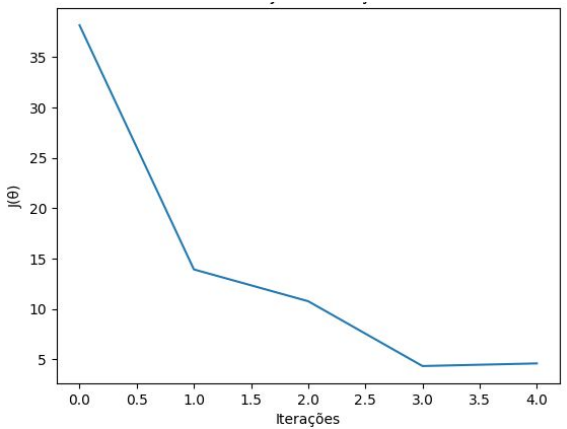
Comparativo do Custo x Iterações



Função de custo
Batch

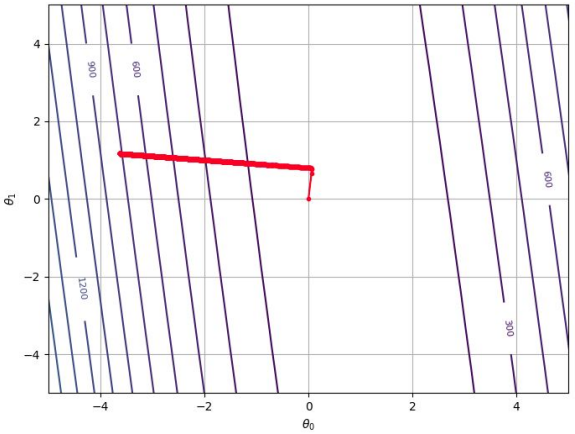


Função de custo
Estocástico

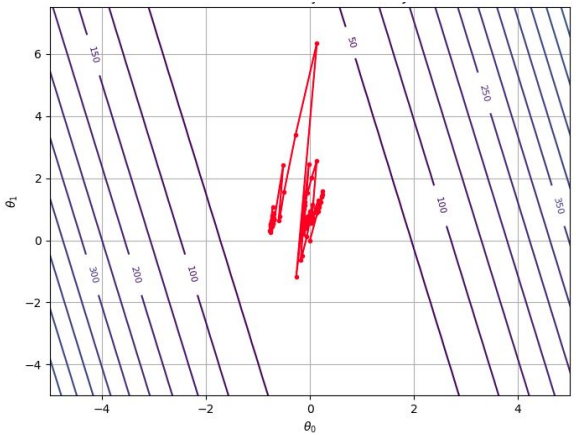


Função de custo
Mini-Batch

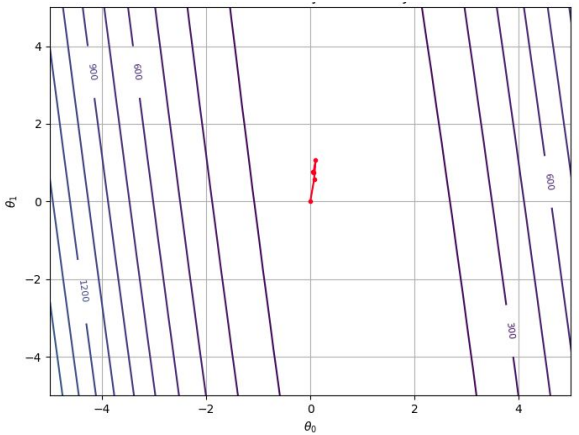
Comparativo do Contorno da Função de Custo



Contorno do Batch



Contorno do Estocástico



Contorno do Mini-Batch

Underfitting, Overfitting e Goodfitting

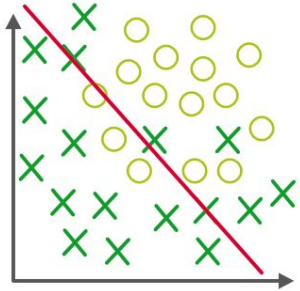
O Problema da Generalização

- Modelos de machine learning aprendem com dados de treino;
- O desafio: ir bem nos dados novos, não só nos que já viu;
- Isso se chama generalização;
- Mas o modelo pode errar por aprender pouco ou demais.

Underfitting (Aprendizado insuficiente)

- Modelo não consegue aprender nem o básico;
- Resultado: baixa performance no treino, validação e teste;
- Sinal de viés alto;
- Causas comuns:
  - Modelo muito simples;
  - Pouco tempo de treino;
  - Taxa de aprendizado muito alta;
  - Regularização excessiva;
  - Aumento de dados mal implementado;
  - Poucos dados ou dados mal representados.

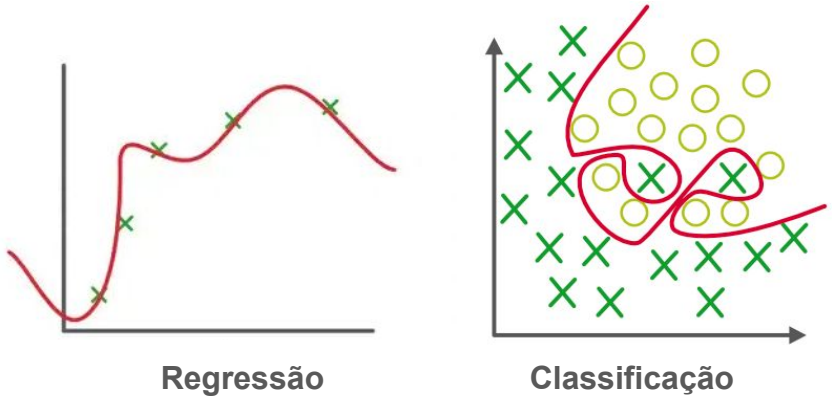
Regressão



Classificação

Overfitting (Aprendeu demais)

- Modelo memoriza demais os dados de treino;
- Vai muito bem no treino, mas mal na validação e teste;
- Sinal de variância alta ;
- Causas comuns:
  - Modelo muito complexo;
  - Poucos dados;
  - Falta de regularização;
  - Vazamento de dados (data leakag
  - Treinou por tempo demais.

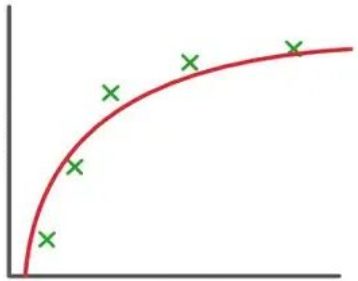


Regressão

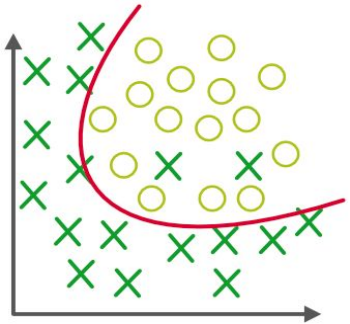
Classificação

Goodfitting (Ajuste ideal)

- Modelo aprende os padrões reais dos dados;
- Vai bem tanto no treino quanto no teste;
- Nem simples demais, nem complexo demais;
- Esse é o objetivo final ao treinar qualquer modelo.



Regressão



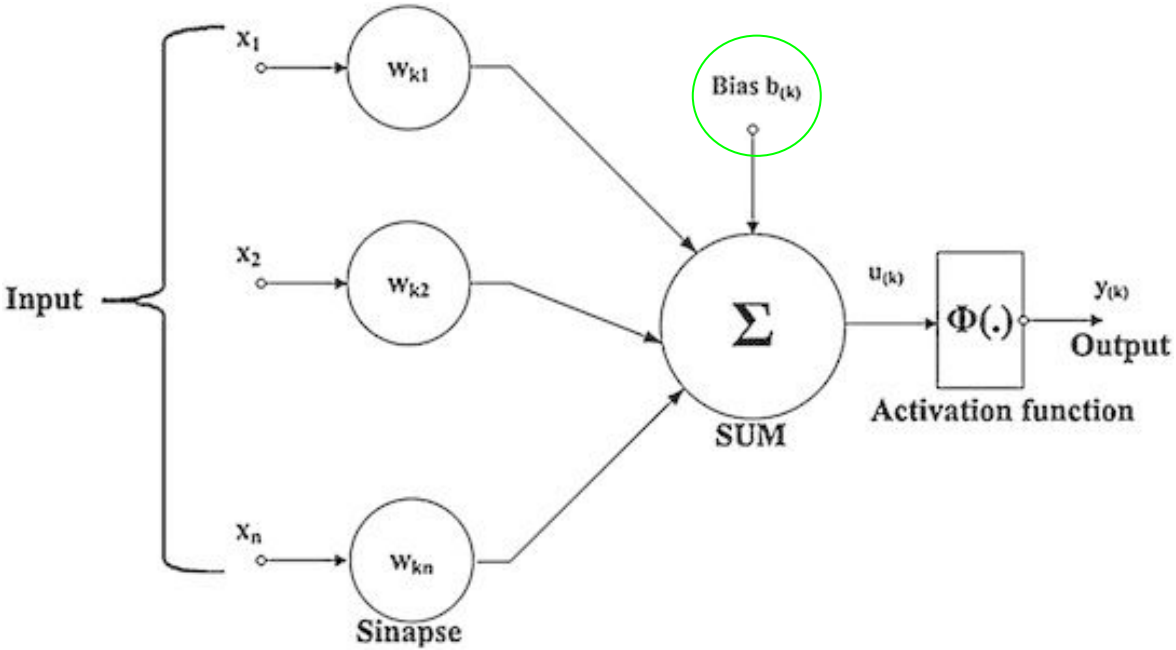
Classificação

Viés e Variância em Redes Neurais Artificiais

Viés (como parâmetro da rede)

- É um valor ajustável que é somado à saída de cada neurônio;
- Funciona como um "ajuste fino" que ajuda a rede a aprender funções mais complexas;
- Não depende das entradas;
- Não precisa ser pequeno ou grande, apenas o valor certo para a tarefa;
- Treinado junto com os pesos durante o aprendizado.

Viés (como parâmetro da rede)



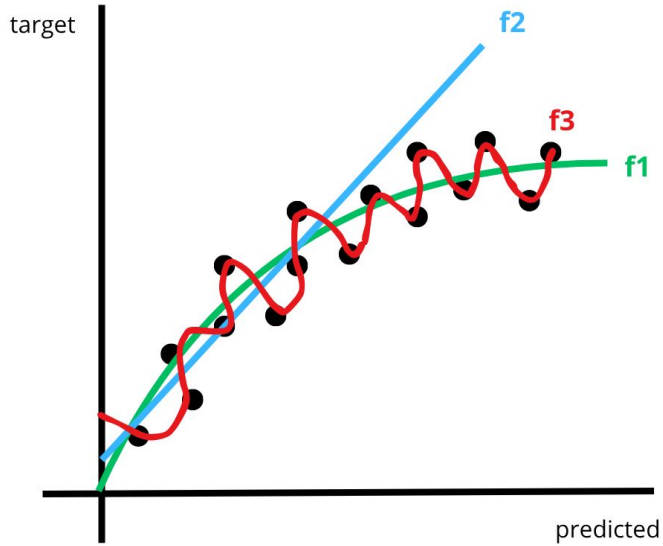
Viés (como componente de erro do modelo)

- Refere-se à simplicidade do modelo;
- Um modelo com alto viés (bias):
  - É muito simples para aprender os padrões;
  - Comete muitos erros mesmo nos dados de treino;
  - Sofre de underfitting.
- Causa: rede pequena demais, mal configurada ou treinada.

Variância

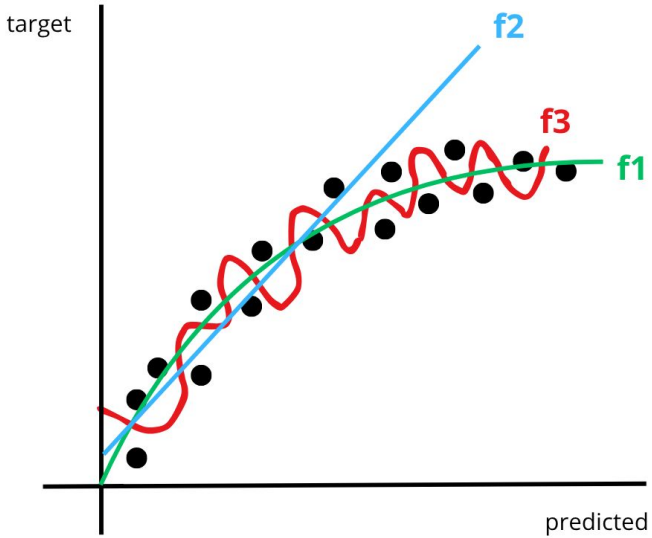
- Mede o quanto o modelo muda se os dados mudarem;
- Um modelo com alta variância:
  - Aprende até os ruídos dos dados de treino;
  - Vai muito bem no treino, mas mal nos dados novos;
  - Sofre de overfitting.
- Causa: rede muito complexa, com poucos dados ou sem regularização.

Bias durante o Treinamento



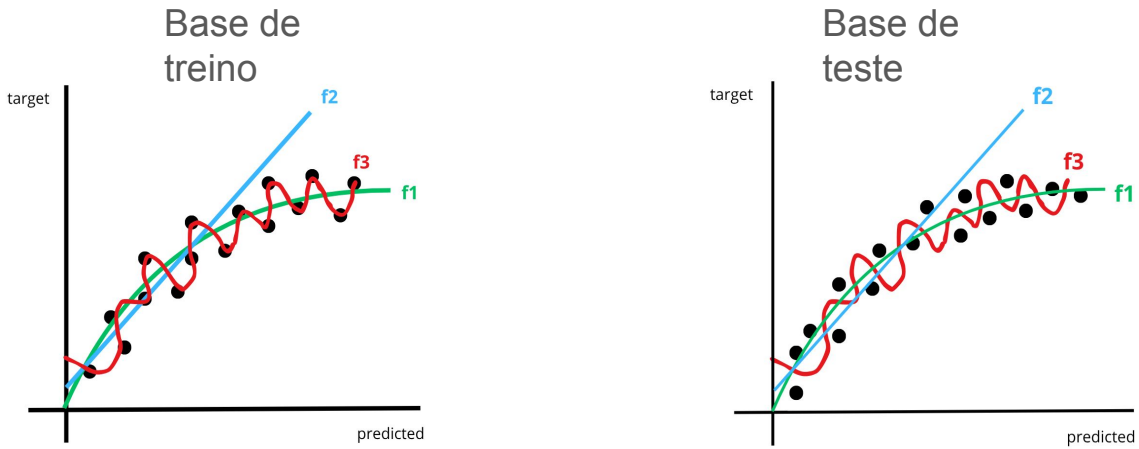
MODELO	BIAS
f1	MÉDIO
f2	ALTO
f3	BAIXO

Bias durante o Teste



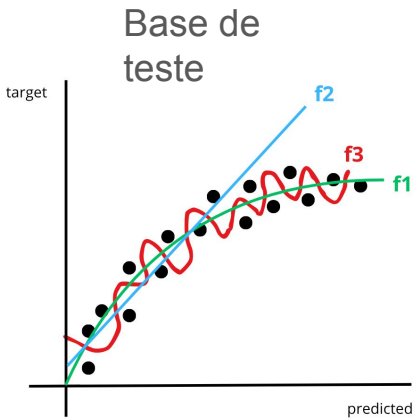
MODELO	BIAS
f1	MÉDIO
f2	ALTO
f3	ALTO

Comparação viés (bias) treino e teste



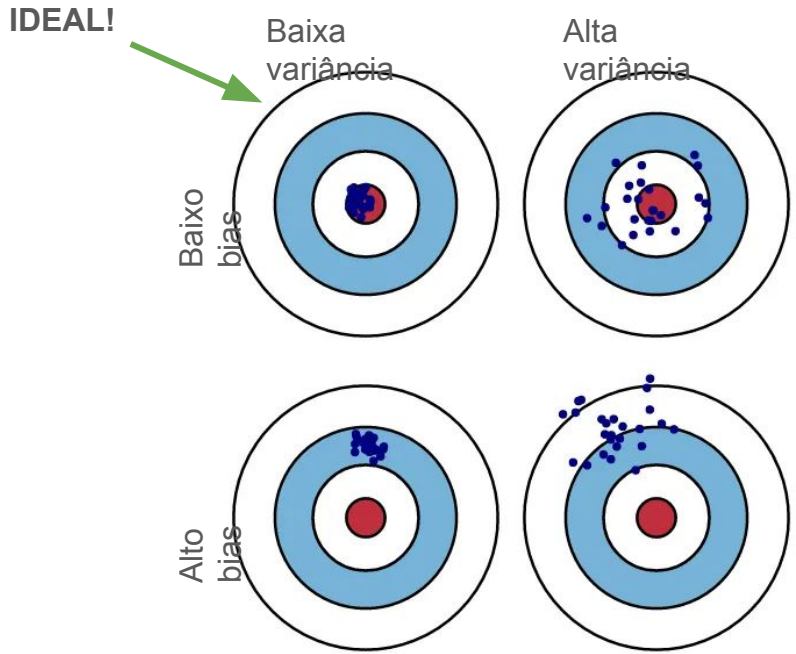
- f3 tem baixo viés por ser flexível, mas alta variância, com pior desempenho em novos dados;
- f3 é overfitting: trabalha bem com os dados de treino, mas é ruim no dataset de teste.

Comparação viés (bias) treino e teste



- f2 é underfitting (possui um viés alto, mas ao mesmo tempo, uma baixa variância)

Ilustração gráfica de bias e variância



Dilema Viés x Variância

- Modelos simples demais = alto viés;
- Modelos complexos demais = alta variância;
- O desafio é encontrar o ponto ideal:
  - Aprender bem os dados;
  - Generalizar para novos exemplos.

Resumo

- Viés pode significar parâmetro do neurônio ou erro do modelo;
- Variância mede a sensibilidade do modelo aos dados de treino;
- Buscar o equilíbrio entre os dois é essencial;
- Técnicas como regularização, validação cruzada, e arquitetura adequada ajudam nesse equilíbrio.

Quantos parâmetros uma rede neural profunda tem?

Uma rede neural profunda pode possuir muitos parâmetros (pesos e bias) a serem aprendidos:

FC: fully connected layer

Camada FC1: 1000 • 500 = 500.000 pesos + 500 bias terms
Camada FC2: 500 • 50 = 25.000 pesos + 50 bias terms
Camada de saída: 50 • 1 = 50 pesos + 1 bias term
Total: 525.601 params
Resultando, assim, em modelos muito complexos

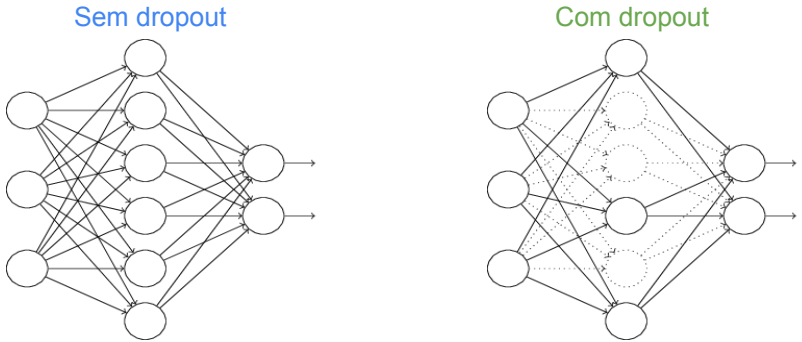
Técnicas de Regularização de Modelos

O que é Regularização?

- Técnicas que ajudam a evitar o overfitting;
- Fazem o modelo generalizar melhor para novos dados;
- Agem como um "freio" na complexidade da rede;
- Objetivo: equilíbrio entre aprender e não exagerar.

Dropout

- Técnica de regularização para redes neurais;
- Ajuda a evitar overfitting;
- Durante o treino, neurônios são desligados aleatoriamente;
- Reduz a dependência entre neurônios;
- A cada passo, uma sub-rede diferente é treinada;
- No teste, todos os neurônios são usados, com ajuste nos pesos.



Dropout

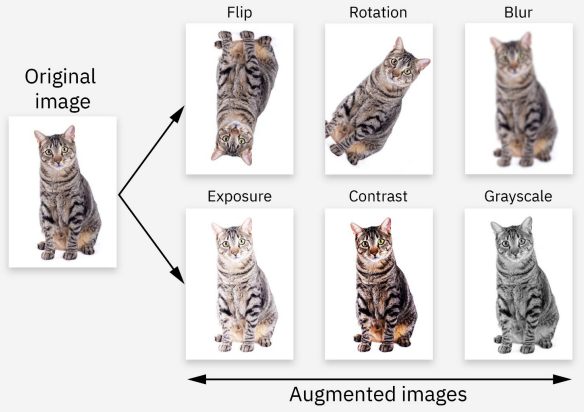
- Evita coadaptação entre neurônios;
- Atua como um ensemble de sub-redes;
- Taxa de dropout define quantos neurônios desligar (ex: 0.5);
- Usado principalmente em camadas densas;
- Simples e eficaz em muitos casos;
- Cuidado com excesso: pode atrapalhar o aprendizado.

Data Augmentation (Aumento de Dados)

- Técnica para criar mais dados a partir dos dados existentes;
- Ajuda a reduzir overfitting em modelos de machine learning;
- Torna os modelos mais robustos e generalizáveis;
- Pode ser aplicada em imagens, textos, áudios e mais.

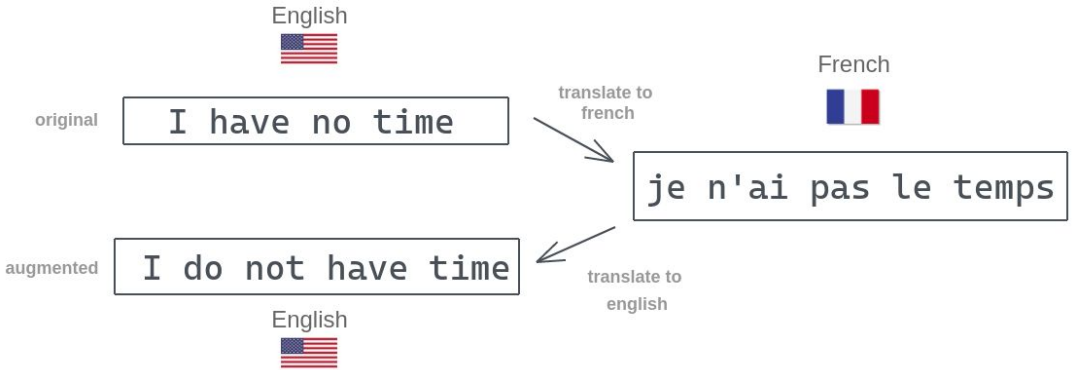
Aumento em Imagens

- Rotação, espelhamento, corte e escala;
- Alterações de brilho, contraste e ruído;
- Aplicação de transformações geométricas;
- Simula variações do mundo real.



Aumento em Texto

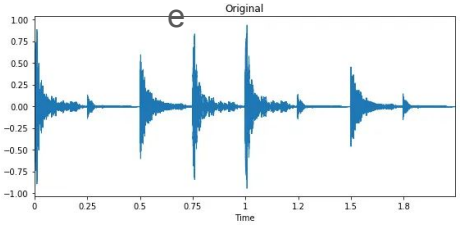
- Substituição por sinônimos;
- Tradução e retrotradução (back-translation);
- Inserção, remoção e troca de palavras;
- Gera novas frases com o mesmo sentido.

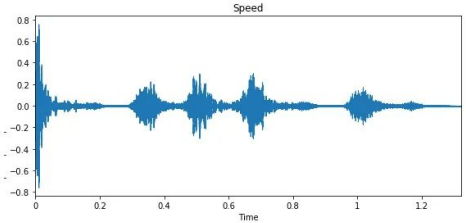


Aumento em Áudio

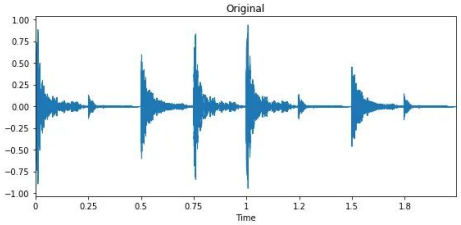
- Mudança de velocidade e pitch;
- Adição de ruídos de fundo;
- Corte e mistura de trechos;
- Simula diferentes ambientes de gravação.

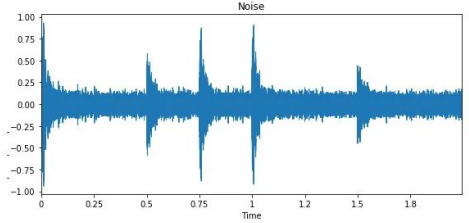
Velocidade



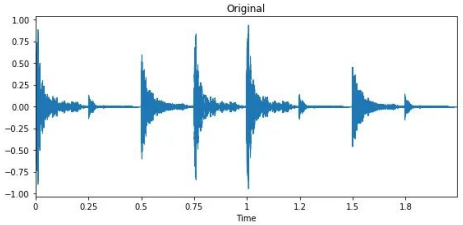


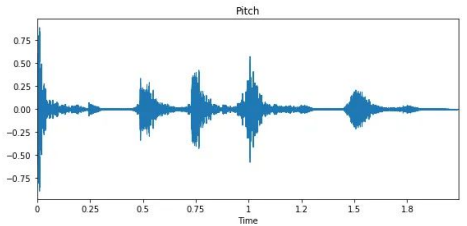
Ruído





Tom





Regularizações L2 (Weight Decay)

- Reduz pesos grandes, mantendo o modelo simples;
- Adiciona um termo de penalização na função de erro;
- Evita que o modelo dependa demais de conexões específicas;

$$Custo\ total = Erro + \lambda \cdot \sum w^2$$

- W são os pesos;
- λ (lambda) é um número que diz quanto a gente se importa com a regularização (ex: 0.01 ou 0.1).

Regularização L2 (Ridge) – Exemplo

$$Custo\ total = Erro + \lambda \cdot \sum w^2$$

θ = [2, −3, 0.5], λ = 0.1

Penalidade:
0.1×(22 + (−3)×2 + 0.52) = 0.1 × (4 + 9 + 0.25) = 0.1×13.25 = 1.325

- Resultado:
  - Todos os pesos são reduzidos suavemente
  - Evita overfitting sem eliminar variáveis

Regularizações L1 (Lasso)

- Penaliza pesos, mas tende a zerar alguns deles;
- Adiciona um termo de penalização mais agressivo na função de custo;
- Modelo com menos conexões ativas;

$$Custo\ total = Erro + \lambda \cdot \sum | w |$$

- w são os pesos;
- λ (lambda) é um número que diz importância dada à regularização (ex: 0.01 ou 0.1).

Pode-se combinar L1 + L2, o que é conhecido como Elastic Net.

Regularização L1 (Lasso) – Exemplo

$$Custo\ total = Erro + \lambda \cdot \sum |\ w\ |$$

θ = [2, −3, 0.5], λ = 0.1

Penalidade:
0.1×(|2| + |−3| + |0.5|) = 0.1×5.5 = 0.55

A penalização é somada ao erro do modelo

- Resultado:
  - Modelo tende a zerar coeficientes menos relevantes
  - Ideal quando só algumas variáveis são úteis

Early Stopping

- Observa o erro em dados de validação;
- Se o erro piora mesmo com mais épocas, paramos o treinamento;
- Evita que o modelo comece a memorizar os dados;
- Simples e eficaz.

Resumo

- Técnicas abordadas:
  - Dropout;
  - Aumento de Dados;
  - Regularização L1 e L2;
  - Early Stopping;
- Regularização = controle contra overfitting;
- O segredo está no equilíbrio entre aprender e não decorar.

Partindo para o código…

Referências

[1] Underfitting and Overfitting in Machine Learning. https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

[2] O Neurônio Biológico e Matemático. https://www.deeplearningbook.com.br/o-neuronio-biologico-e-matematico/

[3] Data Augmentation. https://www.ibm.com/br-pt/think/topics/data-augmentation

Reconhecimentos e Direitos Autorais

@autor: [Filipe das Chagas Pinheiro e Guilherme Roberto Matos Silva]

@contato: [filipe.pinheiro@discente.ufma.br - matos.guilherme@discente.ufma.br]

@data última versão: [13/06/2025]

@versão: 1.0

@outros repositórios: [https://github.com/filipe-pinheiro - https://github.com/guilherme-rms-cv]

@Agradecimentos: Universidade Federal do Maranhão (UFMA), Professor Doutor Thales Levi Azevedo Valente, e colegas de curso.

Copyright / License

Licenciado sob a Licença MIT.

Permissão é concedida, gratuitamente, a qualquer pessoa que obtenha uma cópia deste software e dos arquivos de documentação associados (o "Software"), para lidar no Software sem restrição, incluindo sem limitação os direitos de usar, copiar, modificar, mesclar, publicar, distribuir, sublicenciar e/ou vender cópias do Software, e permitir pessoas a quem o Software é fornecido a fazê-lo, sujeito às seguintes condições:

Este aviso de direitos autorais e este aviso de permissão devem ser incluídos em todas as cópias ou partes substanciais do Software.

O SOFTWARE É FORNECIDO "COMO ESTÁ", SEM GARANTIA DE QUALQUER TIPO,

EXPRESSA OU IMPLÍCITA, INCLUINDO MAS NÃO SE LIMITANDO ÀS GARANTIAS DE COMERCIALIZAÇÃO,

ADEQUAÇÃO A UM DETERMINADO FIM E NÃO INFRINGÊNCIA.

EM NENHUM CASO OS AUTORES OU DETENTORES DE DIREITOS AUTORAIS SERÃO RESPONSÁVEIS

POR QUALQUER RECLAMAÇÃO, DANOS OU OUTRA RESPONSABILIDADE,

SEJA EM AÇÃO DE CONTRATO, TORT OU OUTRA FORMA,

DECORRENTE DE, FORA DE OU EM CONEXÃO COM O SOFTWARE OU O USO OU OUTRAS NEGOCIAÇÕES NO SOFTWARE.

Para mais informações sobre a Licença MIT: