

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

O que são métodos de otimização?

Exemplos:

- Gradiente Descendente;
- Momentum;
- Adagrad (Adaptive Gradient Algorithm);
- RMSProp (Root Mean Square Propagation);
- Adam (Adaptive Moment Estimation);
- ...

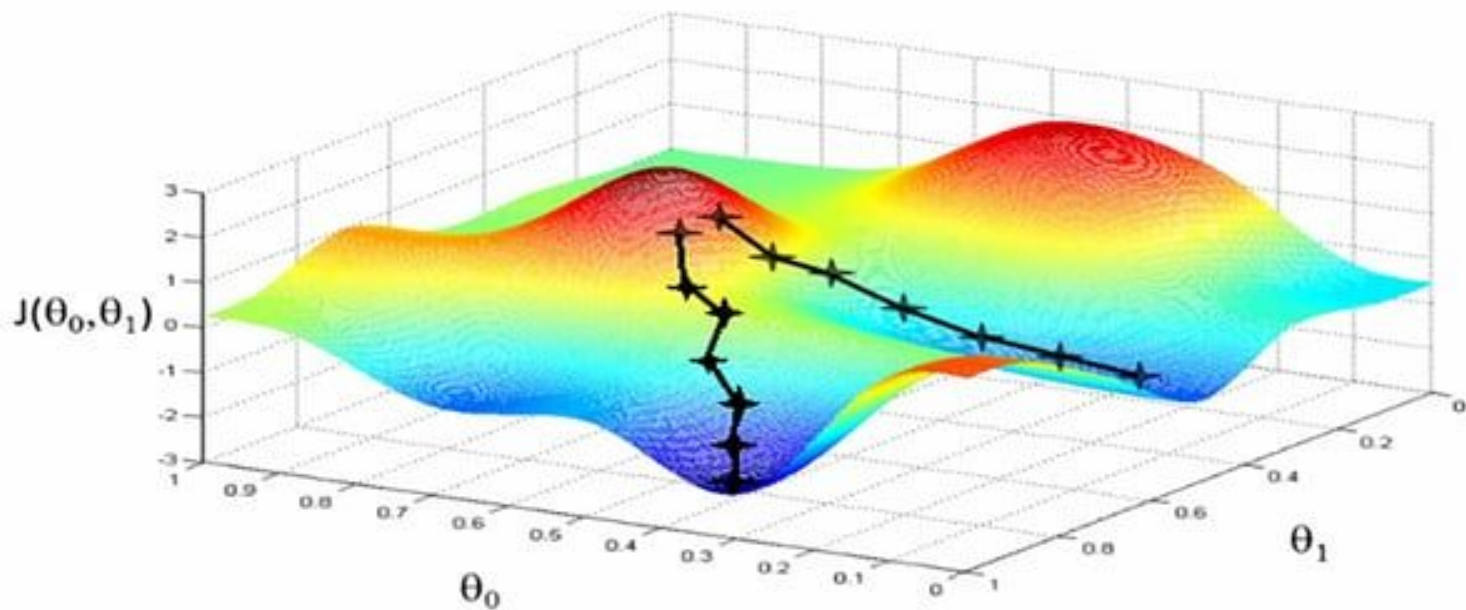
Gradiente Descendente

- Método de otimização iterativo;
- Definida por:

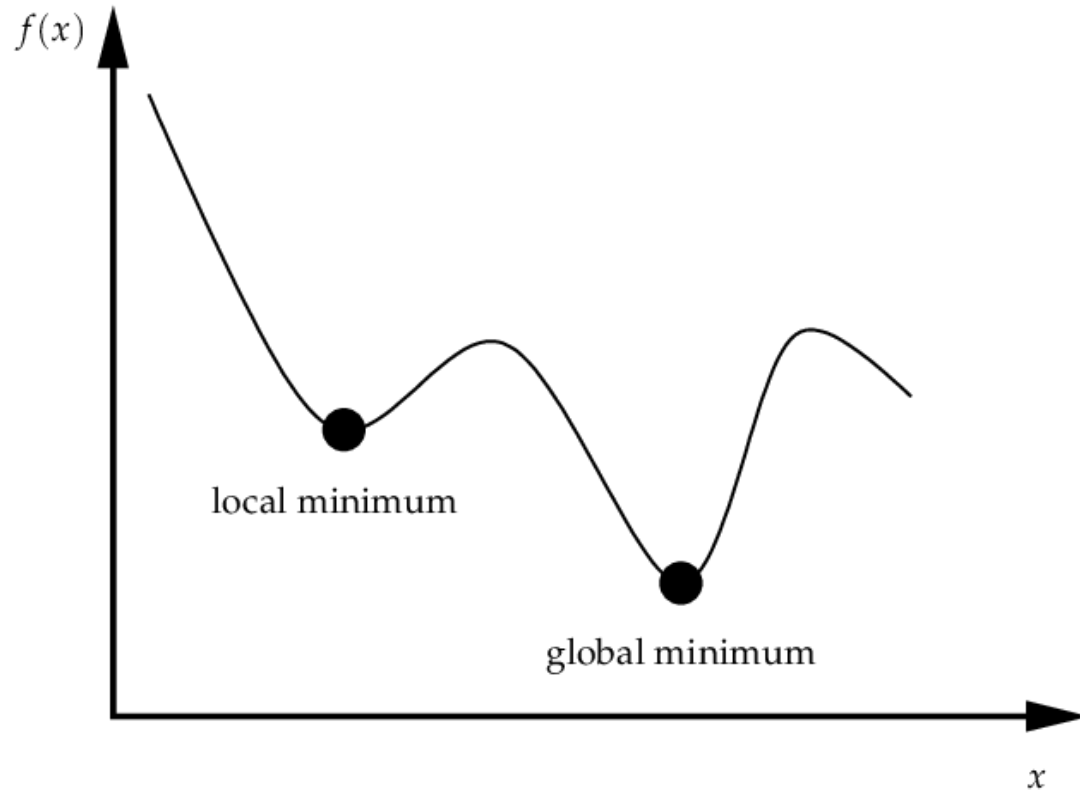
$$\theta := \theta - \alpha \cdot \nabla J(\theta)$$

- θ : parâmetro a ajustar
- α : taxa de aprendizado
- $\nabla J(\theta)$: 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):

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

- \hat{y} : 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

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:

- $x = [1, 2]$
- $y = [2, 4]$

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:

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

Calculando a diferença entre o valor predito e o valor real:

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

Gradiente Descendente 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 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 Batch

Fluxo de uma iteração:

- Definir os parâmetros θ ;
- 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):

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

- \hat{y} : predição
- θ : parâmetros
- x : característica passada
- i : exemplo

Gradiente Descendente Estocástico (SGD)

- Função de custo (MSE):

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

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:

- $x = [1, 2]$
- $y = [2, 4]$

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:

- $x = [1, 2]$
- $y = [2, 4]$

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)

Fluxo de uma iteração:

- Definir os parâmetros θ ;
- 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 Batches.

Gradiente Descendente Mini-Batch

- Demonstração matemática (regressão):

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

- \hat{y} : 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

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:

- $x = [1, 2, 3, 4, 5, 6]$
- $y = [2, 4, 6, 8, 10, 12]$

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 Batches.

Aqui o tamanho do batch será 2:

- $x = [[1, 2], [3, 4], [5, 6]]$
- $y = [[2, 4], [6, 8], [10, 12]]$

Gradiente Descendente Mini-Batch

- Exemplo numérico (1ª iteração):

Para a primeira iteração, será feito o \hat{y} para $x = [1, 2]$ e $y = [2, 4]$, junto a sua diferença:

- $\hat{y}_1 = 0 + 0 \cdot 1 = 0$
- $\hat{y}_2 = 0 + 0 \cdot 2 = 0$
- $\hat{y}_1 - y_1 = 0 - 2 = -2$
- $\hat{y}_2 - y_2 = 0 - 4 = -4$

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 Batches:

- $x = [[1, 2], [3, 4], [5, 6]]$
- $y = [[2, 4], [6, 8], [10, 12]]$

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 \hat{y} para $x = [3, 4]$ e $y = [6, 8]$, junto a sua diferença:

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

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

- Exemplo numérico (3ª iteração):

Relembrando os Batches:

- $x = [[1, 2], [3, 4], [5, 6]]$
- $y = [[2, 4], [6, 8], [10, 12]]$

Onde, com os parâmetros:

- $\theta_0 = 0.448$
- $\theta_1 = 1.31$
- $\alpha = 0.05$

Gradiente Descendente Mini-Batch

- Exemplo numérico (3ª iteração):

Para a segunda iteração, será feito o \hat{y} para $x = [5, 6]$ e $y = [10, 12]$, junto a sua diferença:

- $\hat{y}_1 = 1.31 \cdot 5 + 0.448 = 7.03$

- $\hat{y}_2 = 1.31 \cdot 6 + 0.448 = 8.35$

- $\hat{y}_1 - y_1 = 7.03 - 10 = -2.96$

- $\hat{y}_2 - y_2 = 8.35 - 12 = -3.64$

Gradiente Descendente Mini-Batch

- Exemplo numérico (3ª iteração):

Em seguida é realizado o cálculo do gradiente:

- $\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{2}(-2.96 + (-3.64)) = -3.30$

- $\frac{\partial J(\theta)}{\partial \theta_1} = \frac{1}{2}(-2.96 \cdot 5 + (-3.64) \cdot 6) = -18.34$

Gradiente Descendente Mini-Batch

- Exemplo numérico (3ª iteração):

Atualização dos parâmetros:

- $\theta_0 = 0.448 - 0.05 \times (-3.30) = 0.614$

- $\theta_1 = 1.31 - 0.05 \times (-18.34) = 2.23$

Gradiente Descendente Mini-Batch

Fluxo de uma iteração:

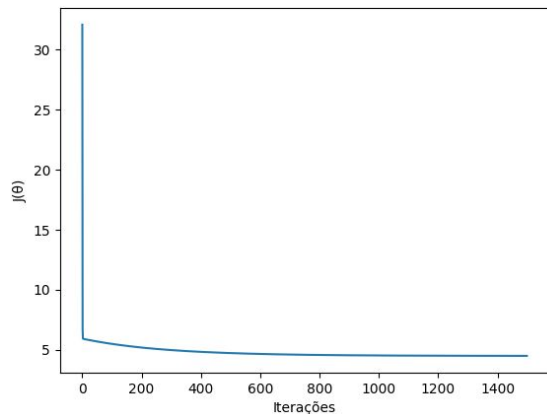
- Definir os parâmetros θ ;
- Definir os Batches e separar os dados de acordo;
- Calcular a predição para o Batch atual;
- Calcular diferença entre predição e valor real;
- Calcular gradiente para o Batch atual;
- Atualizar θ .

Comparativo

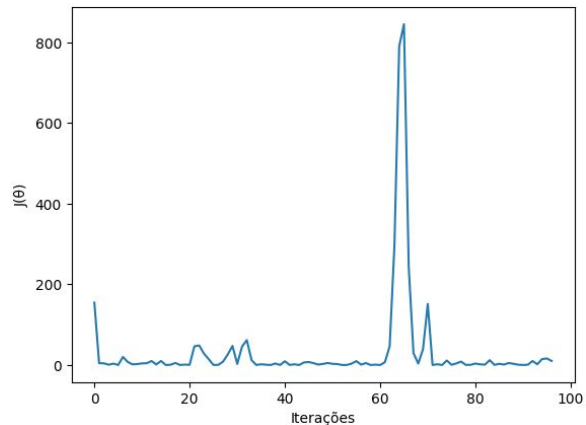
Critério	Batch GD	SGD	Mini-Batch GD
Tamanho do passo	Todos os dados	1 exemplo	Pequeno grupo
Atualização	1x por batch	A cada exemplo	1x por mini-batch
Velocidade	Lenta	Muito rápida	Intermediária
Estabilidade	Alta	Baixa (ruidosa)	Boa
Convergência	Precisa	Rápida	Rápida e estável
Custo computacional	Alto	Baixo	Intermediário
Vetorização	Alta (GPU eficiente)	Baixa	Moderada a alta
Mínimos locais	Mais propenso	Menos propenso	Intermediário
Uso ideal	Dados pequenos	Dados grandes, online	Deep learning, grandes datasets

Partindo para o
código...

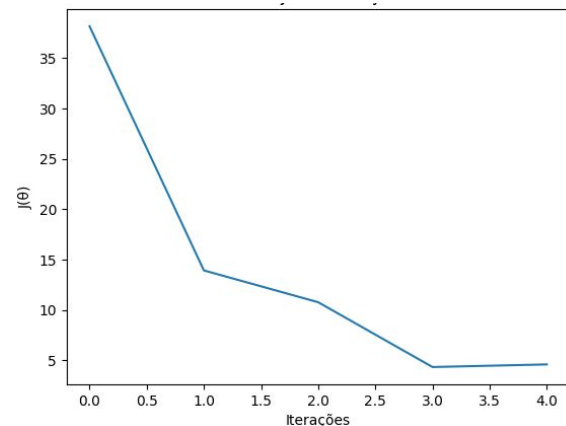
Comparativo do Custo x Iterações



Função de custo
Batch

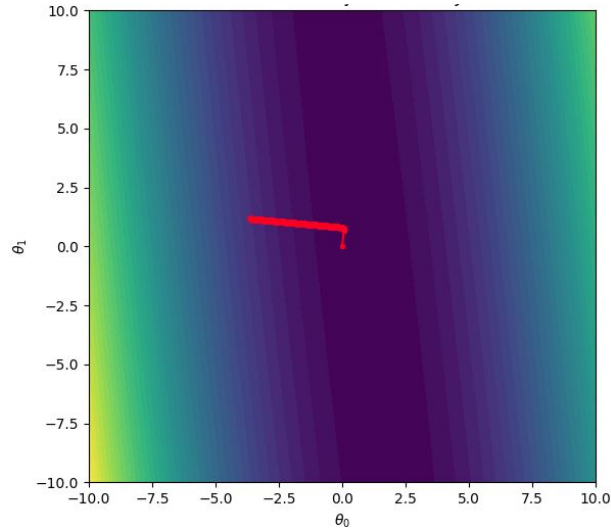


Função de custo
Estocástico

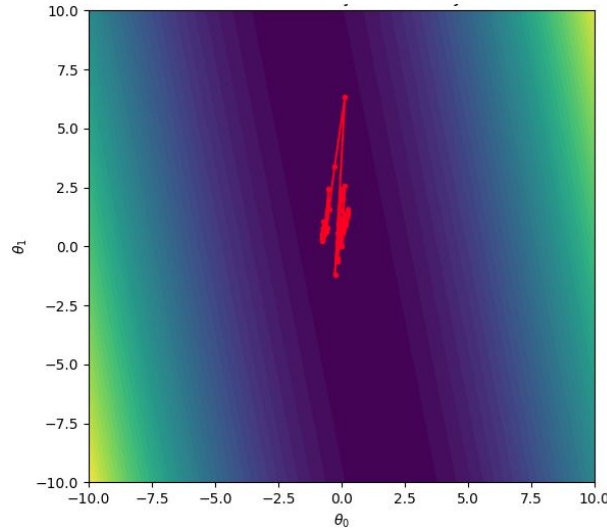


Função de custo
Mini-Batch

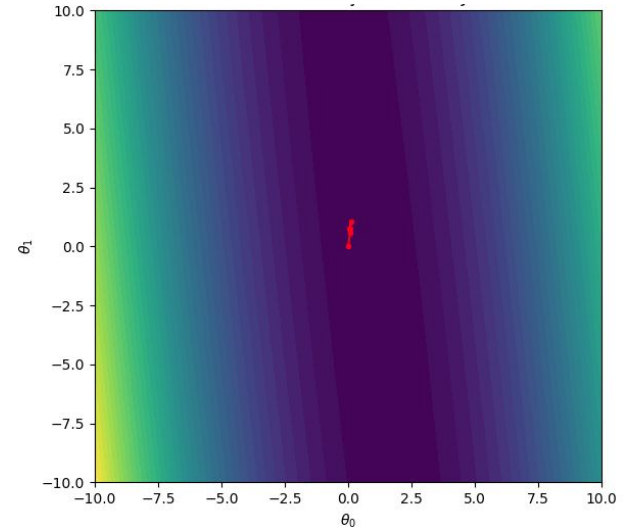
Comparativo do Contorno da Função de Custo



Contorno do
Batch



Contorno do
Estocástico



Contorno do
Mini-Batch

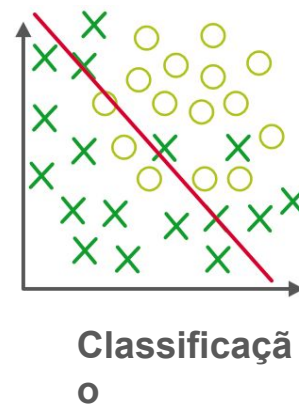
Underfitting, Overfitting e Goodfitting

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.

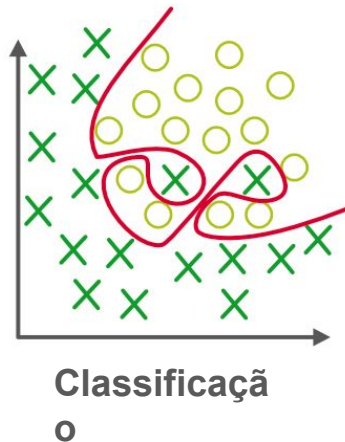
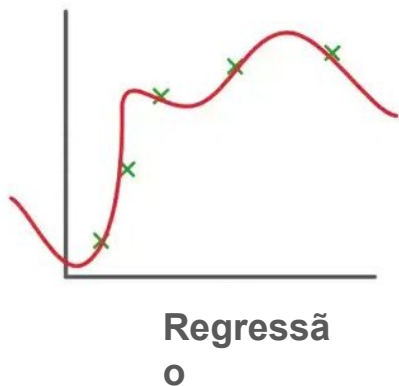
Underfitting (Aprendizado insuficiente)

- Modelo não consegue aprender nem o básico;
- Resultado: baixa performance no treino e no teste;
- Sinal de viés alto;
- Causas comuns:
 - Modelo muito simples;
 - Pouco tempo de treino;
 - Poucos dados ou dados mal representados.



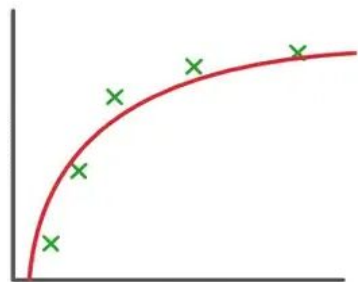
Overfitting (Aprendeu demais)

- Modelo memoriza demais os dados de treino;
- Vai muito bem no treino, mas mal no teste;
- Sinal de variância alta ;
- Causas comuns:
 - Modelo muito complexo;
 - Poucos dados;
 - Treinou por tempo demais.



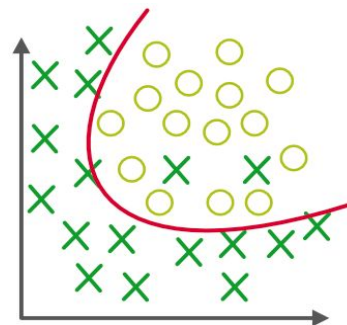
Goodfitting (Ajuste ideal)

- 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.



Regressã

o



Classificaçã

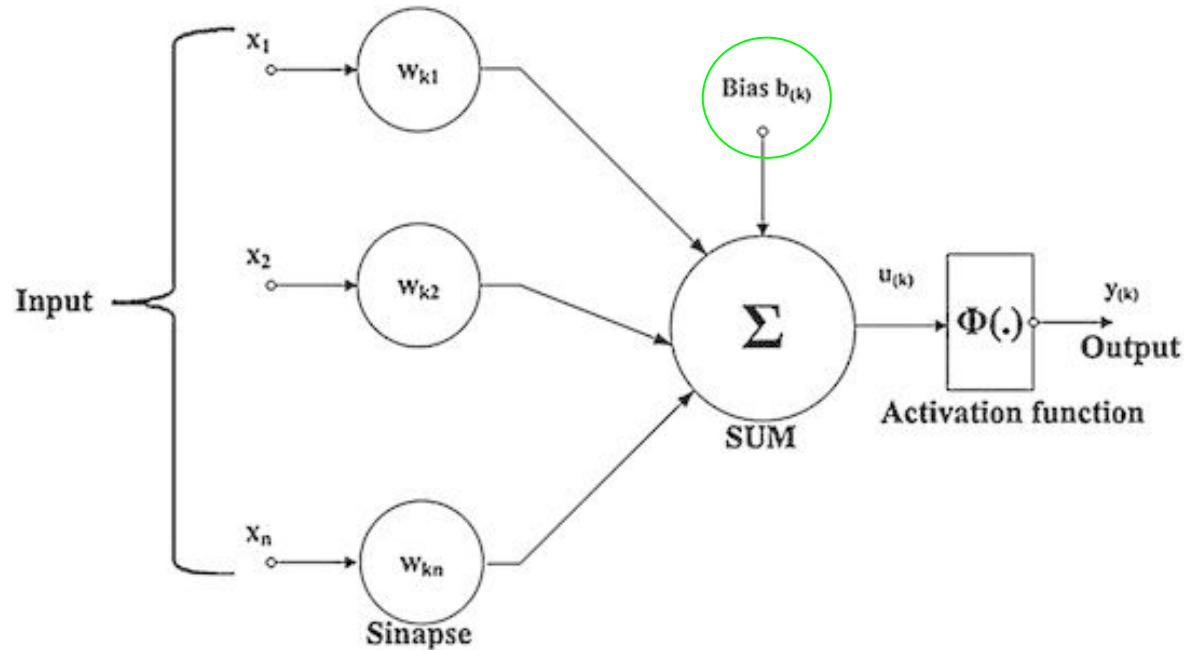
o

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)



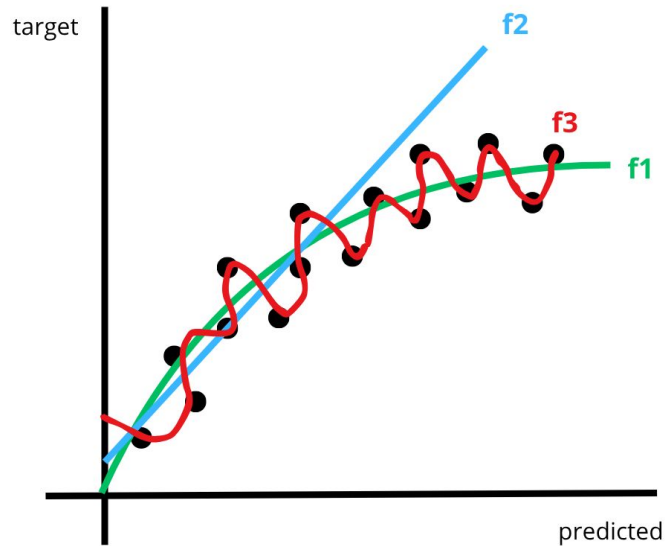
Viés (como componente de erro do modelo)

- 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

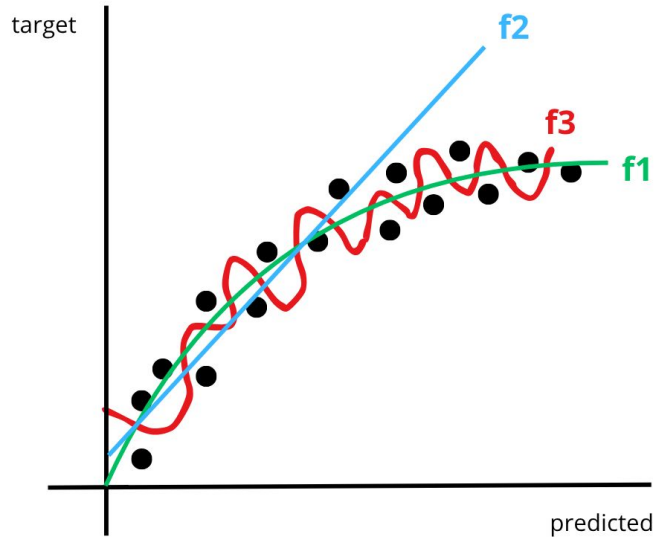
- Mede o quanto o modelo muda se os dados mudarem;
- Um modelo com alta variância:
 - Aprende até os ruídos dos dados de treino;
 - Vai muito bem no treino, mas mal nos dados novos;
 - Sofre de overfitting.
- Causa: rede muito complexa, com poucos dados ou sem regularização.

Bias durante o Treinamento



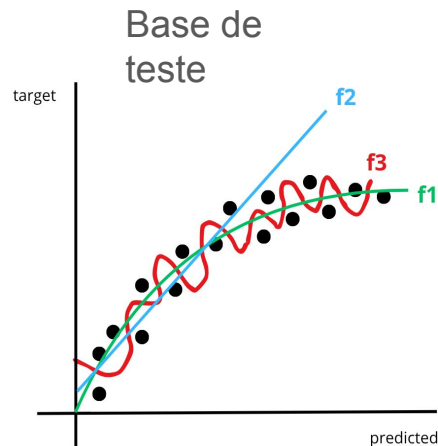
MODELO	BIAS
f1	MÉDIO
f2	ALTO
f3	BAIXO

Bias durante o Teste



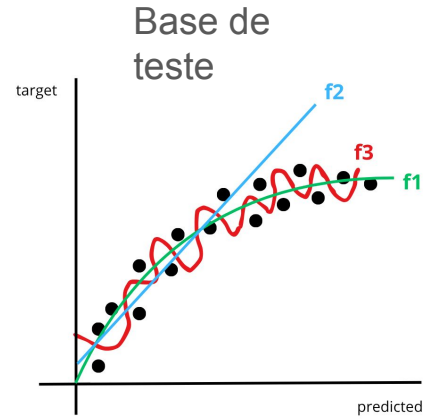
MODELO	BIAS
f1	MÉDIO
f2	ALTO
f3	ALTO

Comparação viés (bias) treino e teste



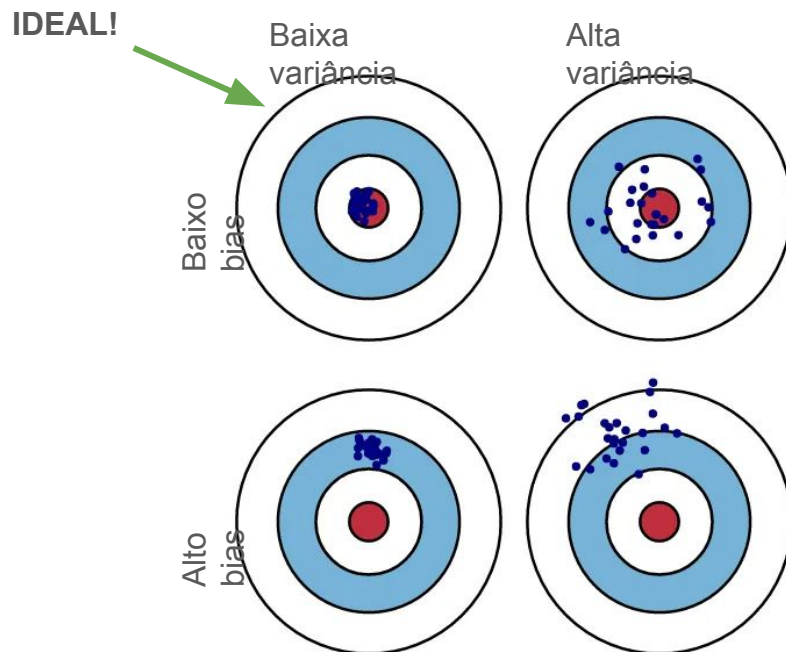
- f3 tem baixo viés por ser flexível, mas alta variância, com pior desempenho em novos dados;
- f3 é overfitting: trabalha bem com os dados de treino, mas é ruim no dataset de teste.

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

1ª camada: 500.000 pesos + 500 bias terms

2ª camada: 25.000 pesos + 50 bias terms

Camada de saída: 50 pesos + 1 bias term

Total: 525.601 params

Resultando, assim, em modelos muito
complexos

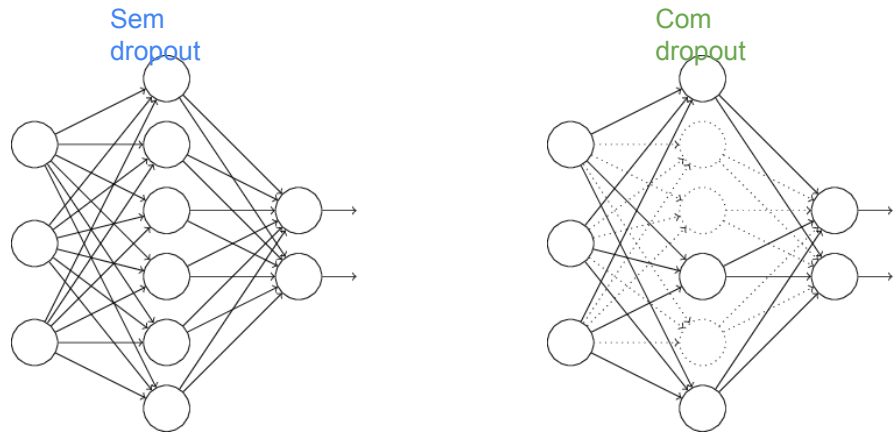
Técnicas de Regularização de Modelos

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.

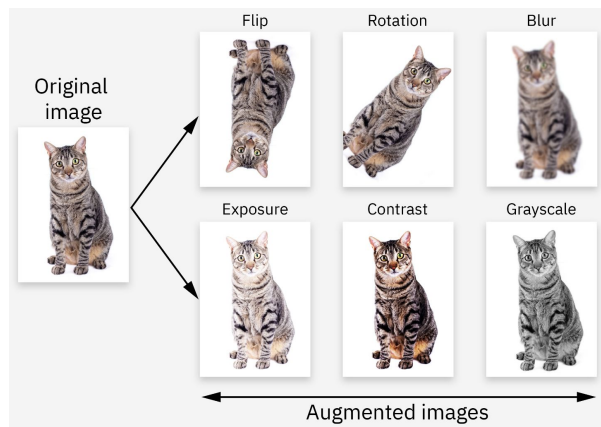
Dropout

- Durante o treino, alguns neurônios são desligados aleatoriamente;
- A rede aprende a não depender de um caminho só;
- Melhora a robustez do modelo;
- No teste, todos os neurônios voltam.



Data Augmentation e Mais Dados

- Adiciona variações nos dados de treino;
- Exemplo: rotação, corte, espelhamento em imagens;
- Torna o modelo mais generalista;
- Quando possível, mais dados reais também ajudam.



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).

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.

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

<https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/>

<https://www.deeplearningbook.com.br/o-neuronio-biologico-e-matematico/>

<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

Este material é resultado de um trabalho acadêmico para a disciplina EECPP0053 - TÓPICOS EM ENGENHARIA DA COMPUTAÇÃO II - FUNDAMENTOS DE REDES NEURAIS, sob a orientação do professor Dr. Thales Levi Azevedo Valente, semestre letivo 2025.1, curso Engenharia da Computação, na Universidade Federal do Maranhão (UFMA).

Todo o material sob esta licença é software livre: pode ser usado para fins acadêmicos e comerciais sem nenhum custo.

Não há papelada, nem royalties, nem restrições de "copyleft" do tipo GNU.

Ele é licenciado sob os termos da Licença MIT, conforme descrito abaixo, e, portanto, é compatível com a GPL e também se qualifica como software de código aberto.

É de domínio público. O espírito desta licença é que você é livre para usar este material para qualquer finalidade, sem nenhum custo.

O único requisito é que, se você usá-lo, nos dê crédito.

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: