Aprendizado de Máquina

Treinando Modelos

Prof. Regis Pires Magalhães

regismagalhaes@ufc.br - http://bit.ly/ufcregis

GÉRON, Aurélien; **Mãos à Obra: Aprendizado de Máquina com Scikit-Learn, Keras & TensorFlow:** Conceitos, Ferramentas e Técnicas para a Construção de Sistemas Inteligentes. 2ª Ed. Alta Books, 2021.

PARTE I - Os conceitos básicos do aprendizado de máquina

1. O Cenário do Aprendizado de Máquina
2. Projeto de Aprendizado de Máquina Ponta a Ponta
3. Classificação
4. **Treinando Modelos**
5. Máquinas de Vetores de Suporte
6. Árvores de Decisão
7. Aprendizado Ensemble e Florestas Aleatórias (Bagging, Random Forests, Boosting, Stacking)
8. Redução de Dimensionalidade (PCA, Kernel PCA, LLE)
9. Técnicas de Aprendizado Não Supervisionado (Clusterização, Misturas de gaussianas)

PARTE II - Redes Neurais e Aprendizado Profundo

10. Introdução às Redes Neurais Artificiais com a Biblioteca Keras
11. Treinando Redes Neurais Profundas
12. Modelos Customizados e Treinamento com a Biblioteca TensorFlow
13. Carregando e Pré-processando Dados com a TensorFlow
14. Visão Computacional Detalhada das Redes Neurais Convolucionais
15. Processamento de Sequências Usando RNNs e CNNs
16. Processamento de Linguagem Natural com RNNs e Mecanismos de Atenção
17. Aprendizado de Representação e Aprendizado Gerativo com Autoencoders e GANs
18. Aprendizado por Reforço
19. Treinamento e Implementação de Modelos TensorFlow em Larga Escala

Objetivos

- Entender os modelos de regressão linear e logística
- Explorar técnicas de treinamento: equação normal e gradiente descendente
- Introduzir regularização e curvas de aprendizado
- Apresentar modelos para classificação: regressão logística e softmax

Métrica R^2

- Outra métrica usada em problemas de regressão:
  - R^2 ou coeficiente de determinação
  - Mede a proporção da variância da variável dependente que é explicada pelo modelo de regressão.
  - Indica o quão bem o modelo se ajusta aos dados observados.
  - O valor 1 indica que o modelo explica 100% da variância dos dados (ajuste perfeito).
  - O Valor 0 indica que o modelo não explica nenhuma variância além da média.
  - Um valor < 0 indica que o modelo é pior do que usar a média como previsão.
  - Quanto mais próximo de 1, melhor.
  - Pode ser enganoso em modelos não lineares, pois supõe linearidade da relação entre variáveis.

Métrica R^2

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Onde:

y_i = valor real do ponto *i*

y^i = valor predito pelo modelo para o ponto *i*

$\bar{y}$ = média de todos os *yi*

n = número de observações

Métrica R^2

- Problemas quando usado em modelos não lineares
  - Mede ajuste, não generalização.
  - Não mede overfitting.
  - Não considera complexidade real.
  - Pode ser usado em modelos não lineares como um indicador auxiliar, mas nunca como única métrica de avaliação.
  - Métricas como RMSE e MAE são mais robustas e seguras.

R^2 x r^2

Coeficiente	Significado	Fórmula	Intervalo
r (Correlação de Pearson)	Grau de associação linear entre duas variáveis X e Y	$r = \frac{\sum(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum(x_i-\bar{x})^2}\sqrt{\sum(y_i-\bar{y})^2}}$	-1 a 1
r^2	Proporção da variância compartilhada entre X e Y	$r^2 = r \cdot r$	0 a 1
R^2 (Coeficiente de determinação)	Proporção da variância de Y explicada pelo modelo	$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}$ onde: $SS_{\text{res}} = \sum(y_i - \hat{y}_i)^2$ $SS_{\text{tot}} = \sum(y_i - \bar{y})^2$	0 a 1*

* pode ser < 0 para modelos ruins, quando o modelo não inclui intercepto e se ajusta pior do que uma média constante.

- SSres – Soma dos quadrados residual
- SStot – Soma dos quadrados total
- Em modelos de regressão linear simples, com apenas uma variável preditora, $R^2 = r^2$. É possível provar isso algebricamente.

RMSE normalizado (nRMSE)

Tipo de normalização	Fórmula	Quando usar
Pelo intervalo (min-max)	$\text{nRMSE} = \frac{\text{RMSE}}{y_{\max} - y_{\min}}$	Quando a escala do target é conhecida
Pela média	$\text{nRMSE} = \frac{\text{RMSE}}{\bar{y}}$	Quando o erro relativo ao valor médio importa
Pelo desvio padrão	$\text{nRMSE} = \frac{\text{RMSE}}{\text{std}(y)}$	Para avaliar se o modelo melhora sobre a média

- nRMSE < 10% → geralmente considerado muito bom
- nRMSE entre 10% e 20% → razoável
- nRMSE > 30% → provavelmente alto
- Pela média é o mais usado e tem leitura direta:
  - "Meu modelo erra, em média, x% do valor real."

Regressão Linear

- 2 maneiras de treinar:
  - Usando uma equação direta de "forma fechada" que calcula os parâmetros do modelo que melhor se ajustam ao modelo no conjunto de treinamento.
  - Usando uma abordagem de otimização iterativa chamada gradiente descendente (GD)
    - Ajusta gradualmente os parâmetros do modelo para minimizar a função de custo no conjunto de treinamento.
    - Acaba convergindo para o mesmo conjunto de parâmetros que o primeiro método.
    - Variantes:
      - GD batch
      - GD mini-batch
      - GD estocástico.

Regressão Linear

- Regressão polinomial
  - Modelo mais complexo que pode se ajustar aos conjuntos de dados não lineares.
  - Mais propenso a sobreajustar os dados de treinamento.
    - Técnicas de regularização podem mitigar o risco de sobreajuste no conjunto.
- Modelos de regressão usados para classificação:
  - Regressão logística
  - Regressão softmax

Regressão Linear

- **Modelo linear** - predição calculando uma soma ponderada das características de entrada, além de uma constante chamada viés (intercepto ou coeficiente linear)

$$\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

  - $\hat{y}$ é o valor previsto.
  - n é o número das características.
  - x_i é o valor da i-ésima característica.
  - θ_j é o j-ésimo parâmetro do modelo (incluindo o viés θ_0 e os pesos das características $\theta_1, \theta_2, \ldots, \theta_n$).

Regressão Linear

- Forma vetorizada:

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta} \cdot \mathbf{x}$$

- θ é o vetor de parâmetro do modelo, que contém o viés θ_0 e os pesos das características θ_1 a θ_n. x é o vetor da característica da instância, que contém de x_0 a x_n, com x_0 sempre igual a 1.
- h_θ é a função de hipótese, usando os parâmetros do modelo θ.

Regressão Linear

- O treino busca encontrar o valor de θ que minimize a RMSE.
- É mais simples minimizar o erro quadrático médio (MSE) do que o RMSE.
- Função de custo MSE para um modelo de regressão linear:

$$\mathrm{MSE}(\mathbf{X}, h_{\boldsymbol{\theta}}) = \frac{1}{m} \sum_{i=1}^{m} \left(\boldsymbol{\theta}^{\top} \mathbf{x}^{(i)} - y^{(i)} \right)^2$$

Regressão Linear

- Equação normal
  - forma analítica (ou fechada) de resolver a regressão linear sem precisar de métodos iterativos como o gradiente descendente.

$$\widehat{\theta} = (\mathbf{X}^{\intercal}\mathbf{X})^{-1} \quad \mathbf{X}^{\intercal} \quad \mathbf{y}$$

- $\hat{\theta}$ - vetor de parâmetros (pesos) que minimizam a função de custo $J(\theta)$.
- X: matriz de entrada
- $X^{\intercal}$: transposta da matriz *X*X
- $(X^{\intercal}X)^{-1}$: inversa da matriz produto $X^{\intercal}X$
- y é o vetor de valores-alvo contendo $y^{(1)}$ a $y^{(m)}$.

Regressão Linear

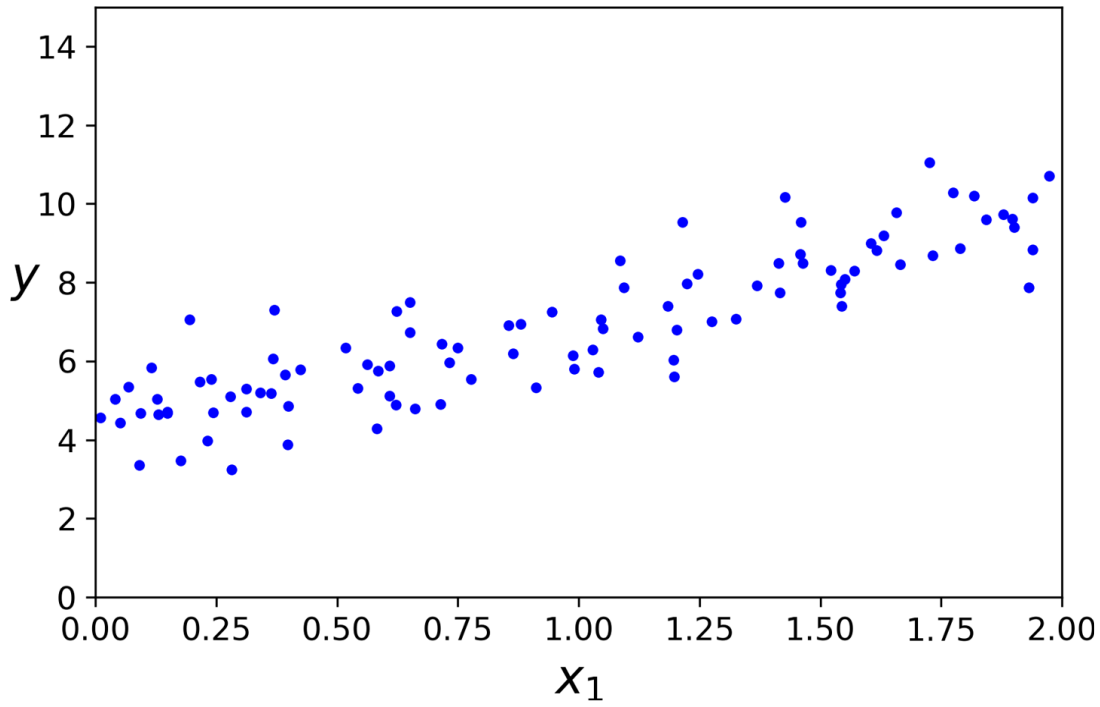
- Equação normal
  - Para encontrar o valor de θ que minimiza a função de custo, existe uma solução de forma fechada, que fornece o resultado diretamente.
  - Custo computacional alto: $O(n^3)$

Regressão Linear

- Geração de conjunto de dados linear
  - $y = 4 + 3x_1 + \text{ruído gaussiano}$

```
import numpy as np

X = 2 * np.random.rand(100, 1)
y = 4 + 3 * X + np.random.randn(100, 1)
```



Regressão Linear

- Equação normal

```
X_b = np.c_[np.ones((100, 1)), X]  # add x0 = 1 to each instance
theta_best = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y)
```

```
>>> theta_best
array([[4.21509616],
       [2.77011339]])
```

$$\hat{\theta} = (X^T X)^{-1} X^T y$$

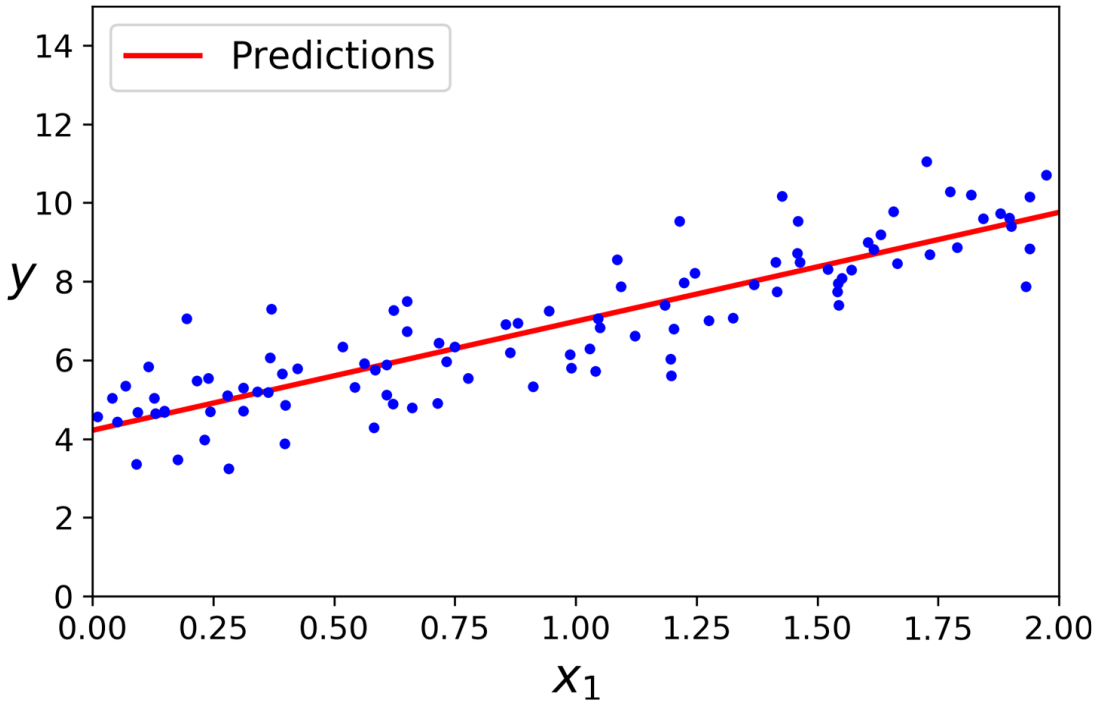
- Fazendo predições usando $\hat{\theta}$:

```
>>> X_new = np.array([[0], [2]])
>>> X_new_b = np.c_[np.ones((2, 1)), X_new] # add x0 = 1 to each instance
>>> y_predict = X_new_b.dot(theta_best)
>>> y_predict
array([[4.21509616],
       [9.75532293]])
```

Regressão Linear

- Equação normal

```
plt.plot(X_new, y_predict, "r-")
plt.plot(X, y, "b.")
plt.axis([0, 2, 0, 15])
plt.show()
```



Regressão Linear

- Usando o scikit-learn:

```
>>> from sklearn.linear_model import LinearRegression
>>> lin_reg = LinearRegression()
>>> lin_reg.fit(X, y)
>>> lin_reg.intercept_, lin_reg.coef_
(array([4.21509616]), array([[2.77011339]]))
>>> lin_reg.predict(X_new)
array([[4.21509616],
       [9.75532293]])
```

- Baseada na função dos mínimos quadrados do scipy/numpy:

```
>>> theta_best_svd, residuals, rank, s = np.linalg.lstsq(X_b, y, rcond=1e-6)
>>> theta_best_svd
array([[4.21509616],
       [2.77011339]])
```

Calcula $\hat{\theta} = X^{+} y$, em que X^{+} é o pseudoinverso de X.
rcond controla o limiar de corte para considerar valores singulares como não nulos durante o cálculo da pseudo-inversa da matrix.

Regressão Linear

- Cálculo direto da pseudo-inversa:

```
>>> np.linalg.pinv(X_b).dot(y)
array([[4.21509616],
       [2.77011339]])
```

- O próprio pseudoinverso é calculado usando uma técnica de fatoração de matriz padrão chamada decomposição em valores singulares (SVD), que pode decompor a matriz do conjunto de treinamento X na multiplicação da matriz de três matrizes $U\,\Sigma\,V^T$.
- O pseudoinverso é calculado como $X^+ = V\,\Sigma^+\,U^T$.

Regressão Linear

- SVD
  - A **Decomposição em Valores Singulares (SVD)** é uma técnica que decompõe uma matriz qualquer X em três matrizes:
  - $X = U \Sigma V^T$
    - U: matriz ortogonal de dimensão m×m.
    - Σ (sigma): matriz **diagonal** (ou quase diagonal, se X não for quadrada), com os **valores singulares** σ1,σ2,... na diagonal.
    - V^T: transposta da matriz ortogonal V, de dimensão n×n.
    - A partir da decomposição $X = U \Sigma V^T$ pode-se calcular o pseudoinverso como:
      - $X^+ = V \Sigma^+ U^T$

Regressão Linear

- Complexidade computacional
  - A equação normal calcula o inverso de $X^T X$ - matriz $(n + 1) \times (n + 1)$, em que n é o número de características.
  - A complexidade computacional da inversão dessa matriz é normalmente de $O(n^{2.4})$ a $O(n^3)$, dependendo da implementação.
  - A abordagem SVD usada pela classe LinearRegression do Scikit-Learn é $O(n^2 \cdot m)$.
    - Onde:
      - n é o número de características (colunas de X)
      - m é o número de amostras (linhas de X)
    - Usa: `np.linalg.lstsq(X, y)`

Regressão Linear

- Complexidade computacional
  - Equação normal e SVD ficam muito lentas quando se aumenta o número de características.
  - Ambas são lineares em relação ao número de instâncias no conjunto de treinamento (são O(m)), lidando com grandes conjuntos de treinamento, desde que se tenha memória para tal.
  - Após treinado, a complexidade computacional é linear para as predições, em relação ao número de instâncias e ao número de características.

Gradiente descendente

- É um algoritmo de otimização genérico que consegue identificar ótimas soluções para um leque amplo de problemas.
- Ideia geral: ajustar iterativamente os parâmetros com o intuito de minimizar uma função de custo.

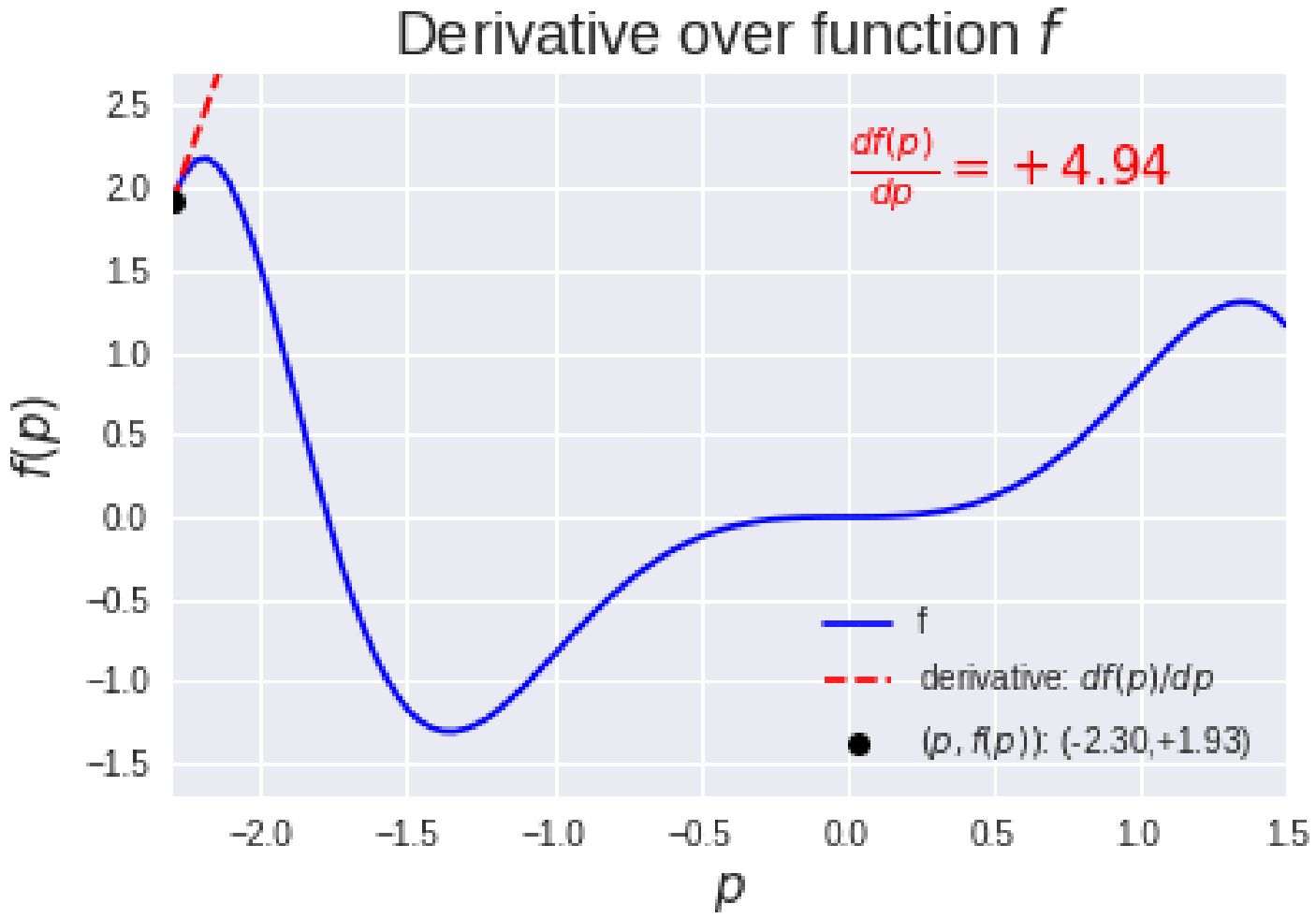
Gradiente descendente - Passos

- Comece com valores aleatórios para os parâmetros θ.
- Calcule o gradiente da função de custo (em que direção o erro aumenta).
- Atualize os parâmetros movendo na direção contrária ao gradiente.
- Repita até o erro parar de diminuir (ou diminuir muito pouco).
- As etapas gradualmente ficam menores à medida que os parâmetros se aproximam do mínimo.
- Quando o gradiente é zero, você atingiu o mínimo!

Gradiente descendente - Termos

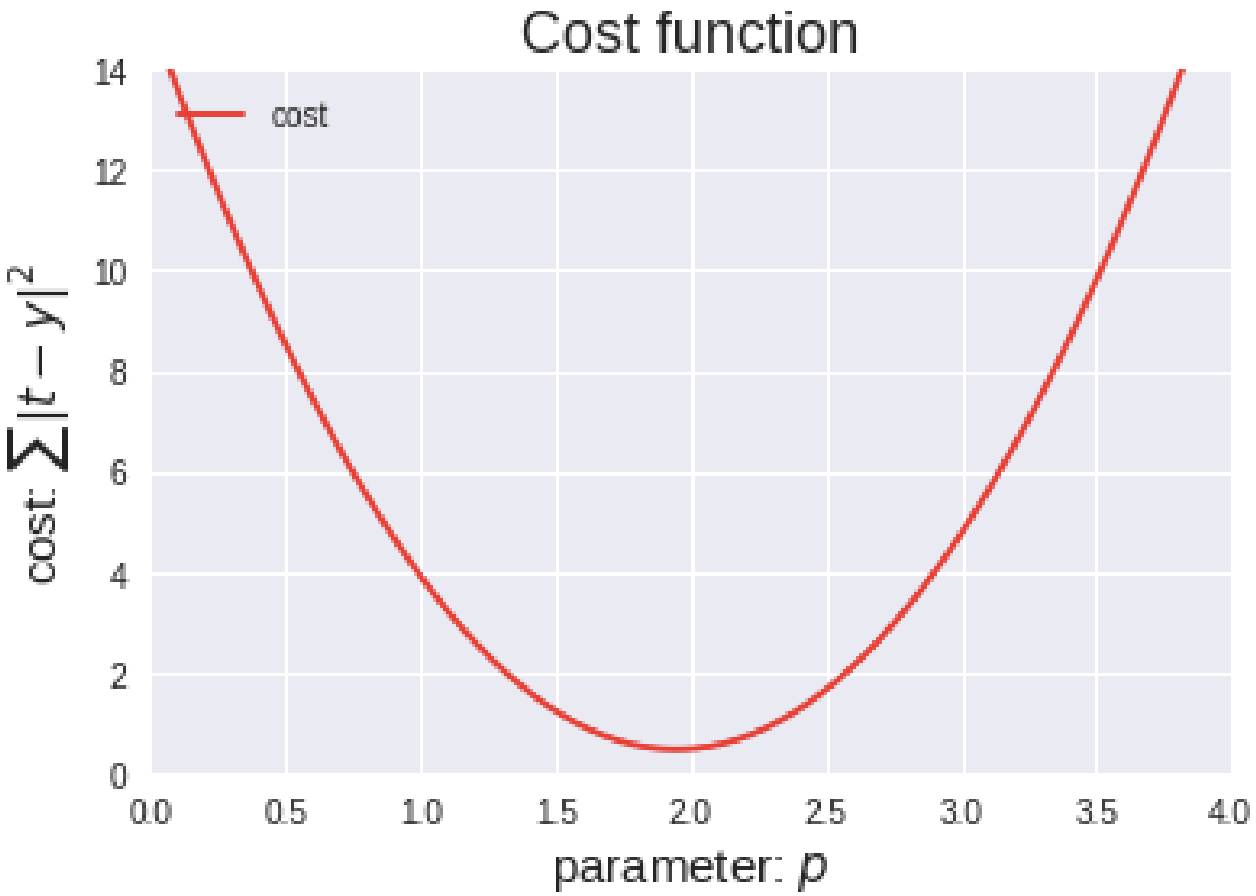
- α (learning rate)
  - Tamanho do passo que você dá em cada iteração
- Gradiente
  - Vetor que aponta para a direção onde o erro cresce
- θ
  - Parâmetros do modelo (ex.: inclinação e intercepto)
- Iterações
  - Quantas vezes você atualiza os parâmetros

Derivada de uma função f



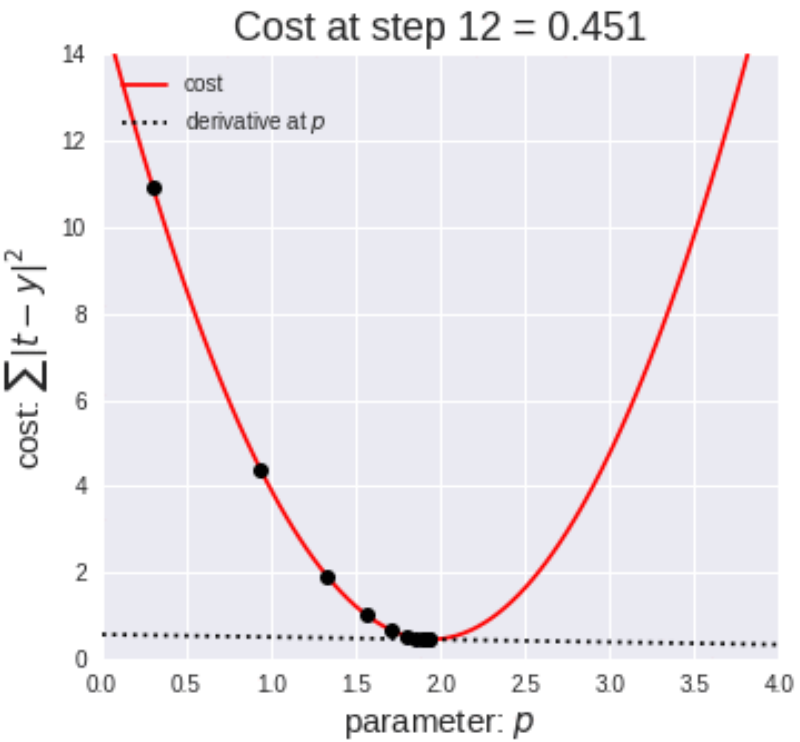
Função de custo

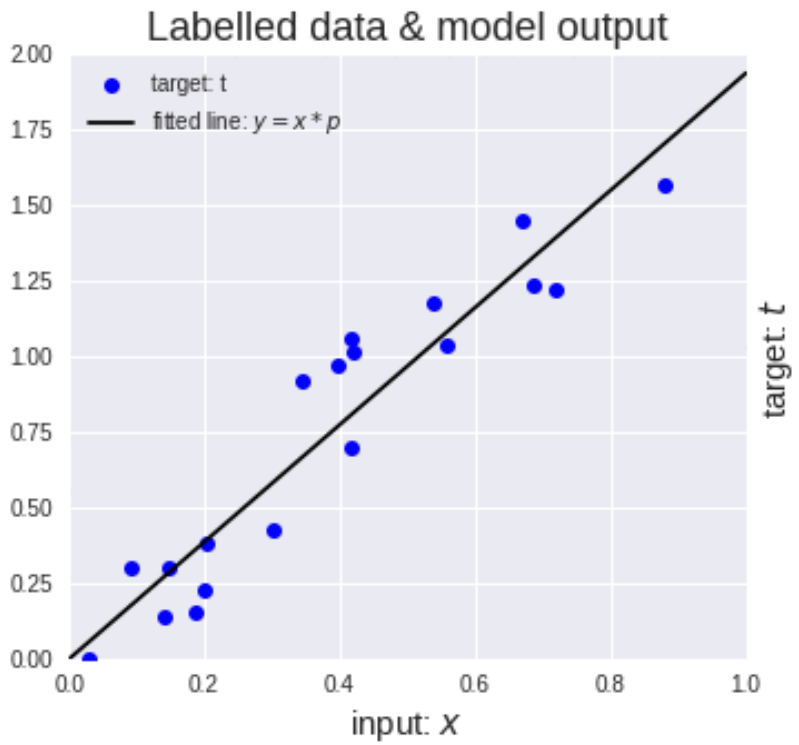
objetivo: minimizar o erro quadrado



Minimizando a função de custo

$$Error_{\beta_0,\beta_1} = \frac{1}{N}\sum_{i=1}^{N}(y_i - (\beta_1 x_i + \beta_0))^2$$

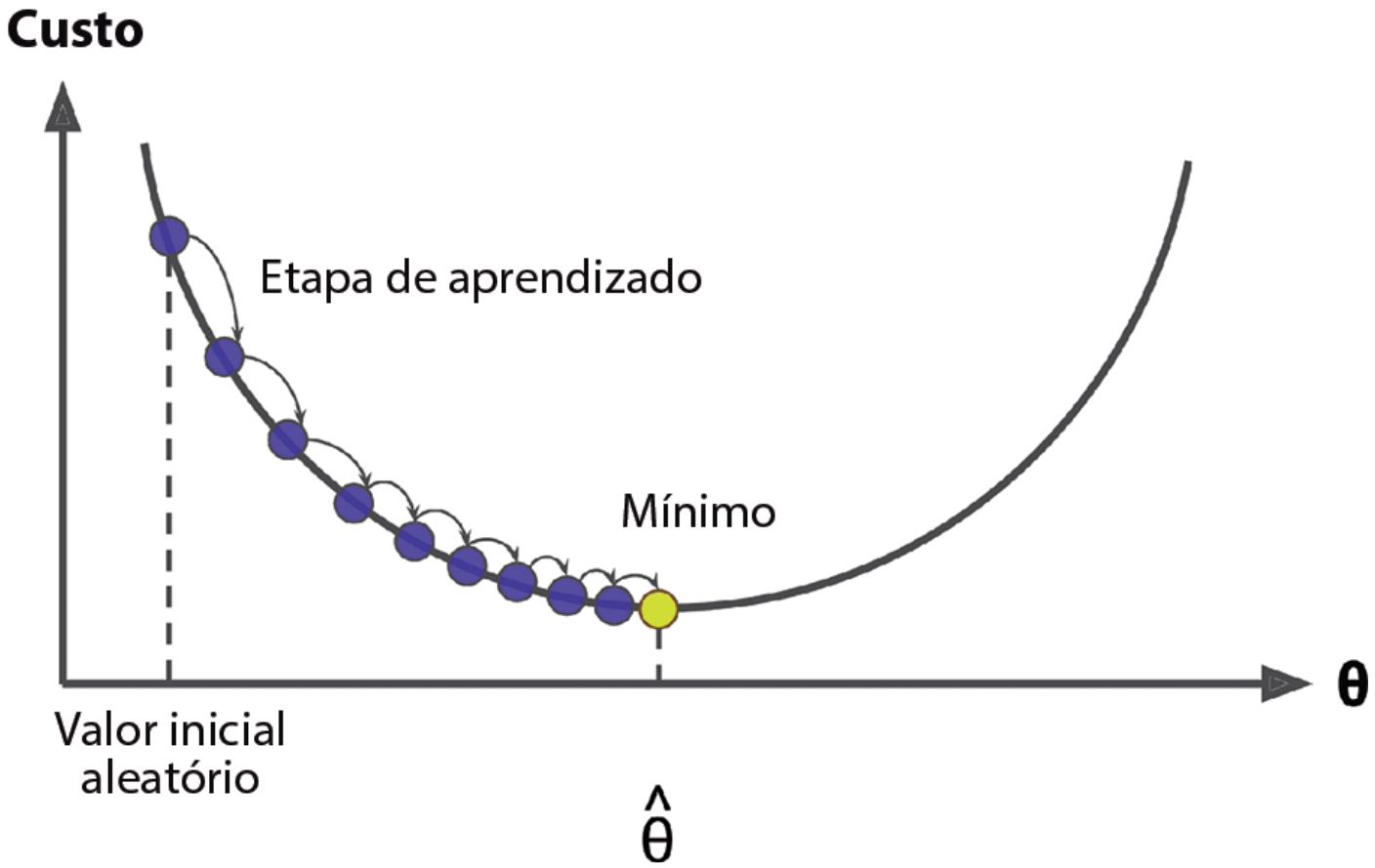




Fonte: https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/

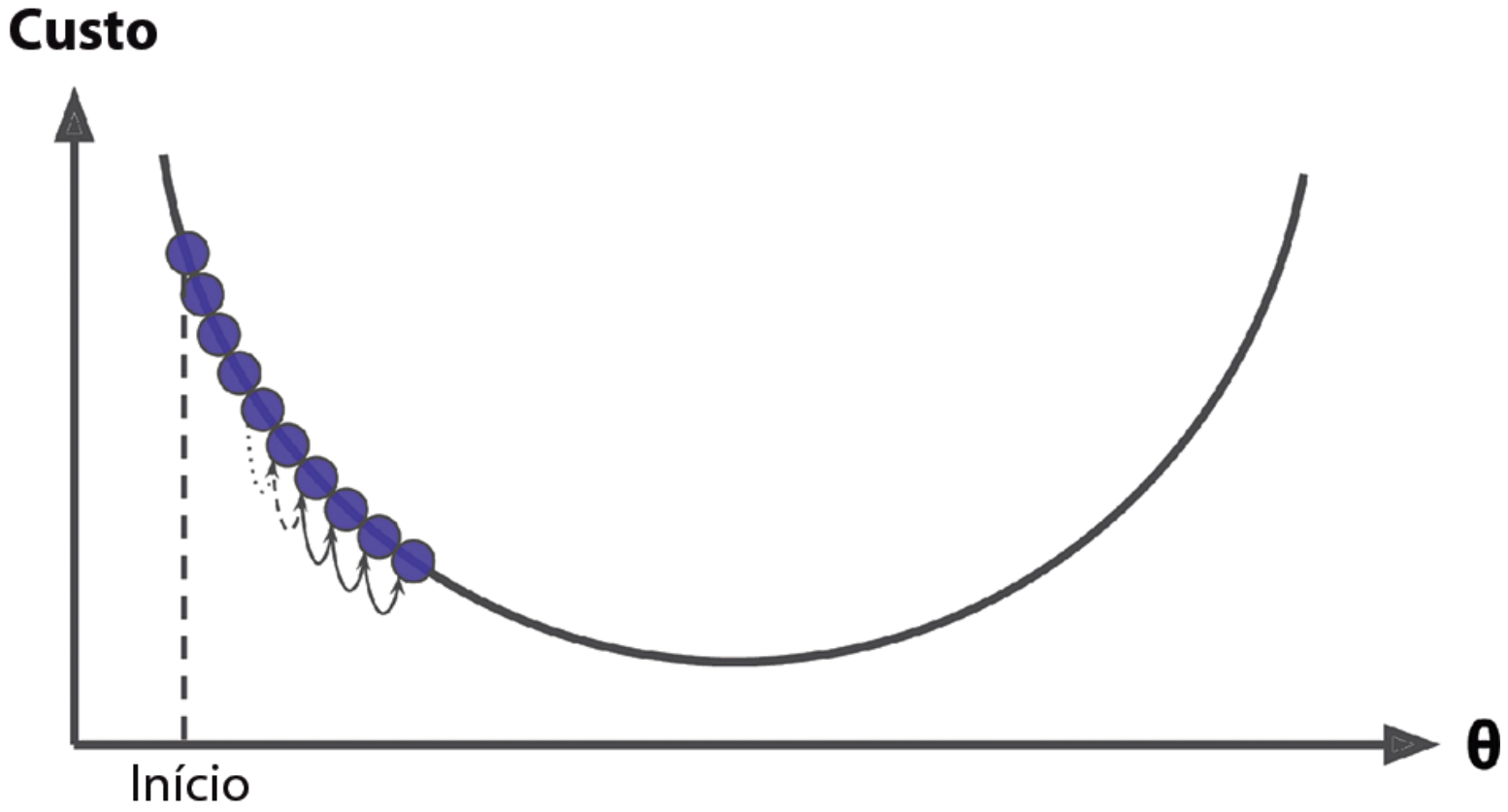
Gradiente descendente

Se a taxa de aprendizado for muito pequena, o algoritmo precisará passar por muitas iterações para convergir, o que levará muito tempo.



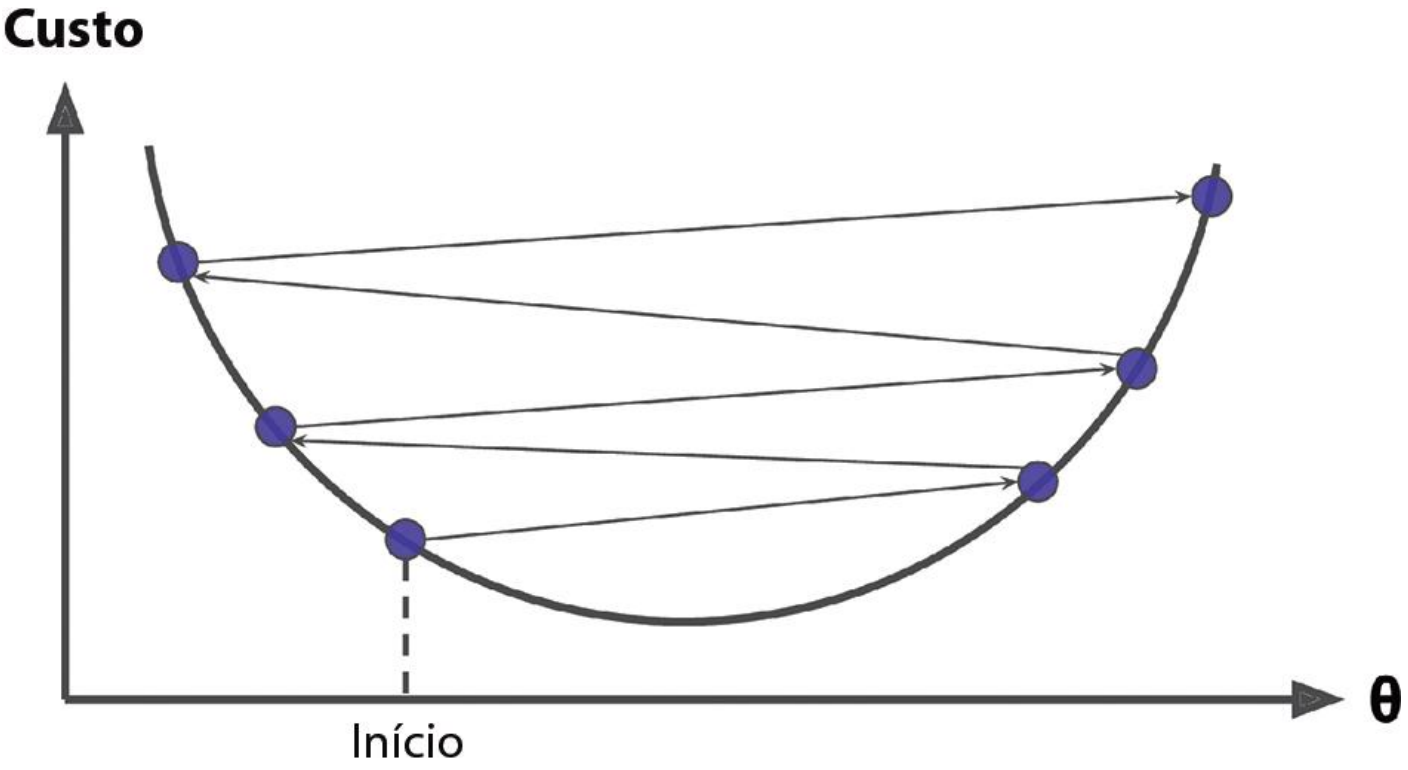
Gradiente descendente

Uma taxa de aprendizado muito pequena.



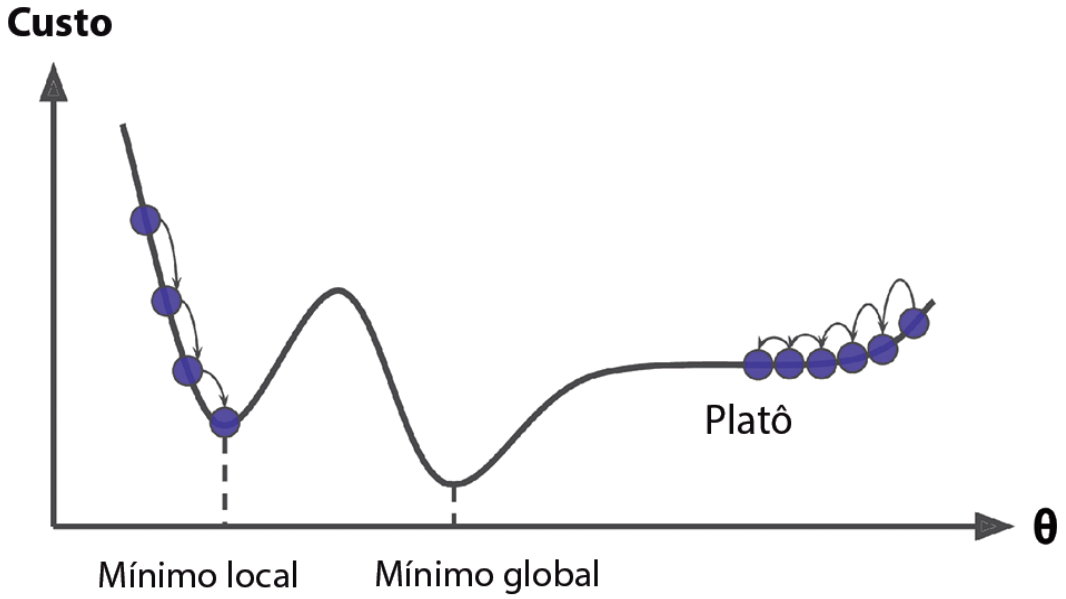
Gradiente descendente

- Se a taxa de aprendizado for muito alta, você pode atravessar o vale e acabar do outro lado, possivelmente em um lugar mais alto do que estava antes. Nesse caso, o algoritmo pode divergir, com valores cada vez maiores, e não encontrar uma boa solução.



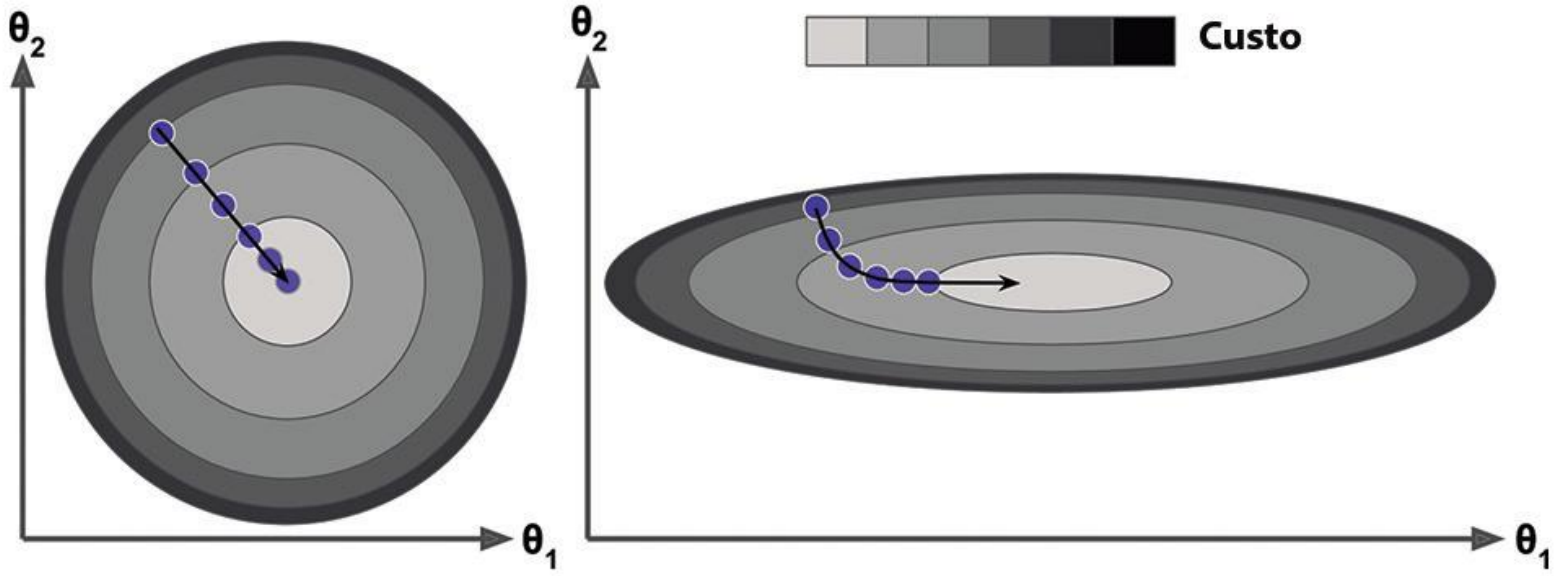
Obstáculos do gradiente descendente

- Dois principais desafios do gradiente descendente:
  - Se a inicialização aleatória iniciar o algoritmo à esquerda, ele convergirá para um mínimo local, que não é tão bom quanto o mínimo global.
  - Se iniciar à direita, levará muito tempo para atravessar o platô. E, se você parar cedo demais, nunca alcançará o mínimo global.



Gradiente descendente com e sem escalonamento de características

- Ao usar o gradiente descendente, você deve garantir que todas as características tenham uma escala semelhante (por exemplo, usar a classe StandardScaler do Scikit-Learn) ou levará um bom tempo para convergir.



No 1º gráfico, as variáveis θ_1 e θ_2 estão na mesma escala e o caminho do gradiente descendente (as bolinhas azuis) vai direto ao centro.

Gradiente descendente

- Alternativa iterativa para otimização
- Atualização dos parâmetros:
  - $\theta = \theta - \eta \nabla_\theta \mathrm{MSE}(\theta)$

Gradiente descendente

- Tipos
  - Batch (em lote)
    - usa todos os dados de treino em cada passo.
  - Estocástico (SGD) = aleatório
    - atualiza os parâmetros com apenas uma amostra por vez.
    - Erro e gradiente mudam a cada iteração porque são baseados em uma amostra aleatória.
    - Como cada exemplo é diferente, isso introduz variabilidade (aleatoriedade) no caminho da descida.
  - Mini-batch
    - Atualiza os parâmetros com apenas um pequeno grupo de amostras por vez.

Gradiente descendente

Algoritmo	Grande m	Grande n	Out-of-core	Hiperparâmetros
Equação Normal	Sim	Não	Não	Nenhum
SVD	Sim	Não	Não	Nenhum
GD (Batch)	Não	Sim	Não	η, epochs
SGD	Sim	Sim	Sim	η, schedule
Mini-batch	Sim	Sim	Sim	η, tamanho

m → linhas
n → colunas

Regressão polinomial

- Estende atributos com potências e interações
- Capaz de modelar dados não-lineares
- Risco maior de overfitting

Curvas de aprendizado

- Treinamento vs Validação
- Diagnóstico:
  - Underfitting: Erros altos e semelhantes
  - Overfitting: Grande distância entre erros
- Soluções:
  - Mais dados (para overfitting)
  - Modelo mais complexo (para underfitting)

Curvas de aprendizado

- Custo benefício do Viés/ Variância
- **Viés**: suposições incorretas (ex: modelo linear em dado não-linear)
- **Variância**: sensível a variações nos dados
- Erro de generalização = viés + variância + ruído

Modelos Lineares Regularizados

- Reduz overfitting ao penalizar pesos altos
- Modelos:
  - Ridge (L2): penaliza soma dos quadrados
  - Lasso (L1): zera pesos irrelevantes (seleção de atributos)
  - Elastic Net: combina L1 e L2

Modelos Lineares Regularizados

- Parada Antecipada (Early Stopping)
  - Regularização alternativa para modelos iterativos
  - Interrompe o treinamento no melhor desempenho de validação

Regressão Logística

- Usada para classificação binária
- Saída: probabilidade via função sigmoide
- Predição: y = 1y = 1 se p^≥0.5
- Função de custo: log loss

Regressão Logística

- Estimando as probabilidades
  - Calcula score linear: $t = \theta^T x$
  - Aplica função sigmoide:

$$\hat{p} = \sigma(t) = \frac{1}{1+e^{-t}}$$

  - Resultado é a probabilidade da classe positiva
  - Se $\hat{p} \geq 0.5$, prediz classe 1, senão 0

Regressão Logística

- Treinamento e função de custo
  - Objetivo: estimar parâmetros θ que maximizem as probabilidades
  - Função de custo: log loss ou entropia cruzada
  - Forma vetorizada: $J(\theta) = -1m \sum i=1m [y(i) \log(p\hat{}(i)) + (1 - y(i)) \log(1 - p\hat{}(i))]$
  - Convexa: permite otimização com gradiente descendente

Regressão Logística

- Fronteiras de decisão
  - Exemplo: detecção da espécie *Iris virginica*
  - Fronteira é linear em atributos
  - Probabilidade define decisão

Regressão Logística

- Regressão softmax
  - Classifica em múltiplas classes mutuamente exclusivas
  - Generaliza a regressão logística
  - Calcula *scores* por classe e aplica *softmax*
  - Predição: classe com maior probabilidade

Regressão Logística

- Entropia Cruzada
  - Mede a diferença entre distribuição real e prevista.
  - Equivalente à *log loss* na regressão logística.
  - Fórmula para um exemplo:
    - $c(\theta) = -[\, y \log(\hat{p}) + (1 - y) \log(1 - \hat{p})]$
  - Penaliza previsões erradas com mais intensidade.
  - Ótima para classificação binária com saída probabilística.

Conclusão

- Capítulo essencial para compreender redes neurais
- Regressão linear como base
- Gradiente descendente é amplamente utilizado
- Regularização é chave para evitar overfitting

Obrigado!

Dúvidas, comentários, sugestões?

Regis Pires Magalhães
regismagalhaes@ufc.br

UFC