Regressão Linear Aplicada em Machine Learning: Um Guia para Iniciantes

Autor do Artigo

April 4, 2025

Abstract

A regressão linear é um dos algoritmos fundamentais de aprendizado de máquina supervisionado, utilizado para prever valores numéricos contínuos com base em dados históricos. Este artigo apresenta uma visão detalhada sobre regressão linear, iniciando dos conceitos básicos até tópicos mais avançados. Exploramos a definição e importância da regressão linear em Machine Learning, discutimos suas vantagens e desvantagens, descrevemos formalmente as equações e cálculos envolvidos (incluindo o método dos mínimos quadrados para estimar coeficientes) e introduzimos variações importantes como regressão múltipla, Ridge e Lasso. Também apresentamos exemplos de aplicação em cenários reais, como previsão de preços de imóveis e consumo de energia, acompanhados de implementações práticas em Python (usando bibliotecas como NumPy, scikit-learn e matplotlib) com visualizações gráficas. O conteúdo é estruturado de forma didática para atender iniciantes, com explicações passo a passo, equações formatadas e códigos comentados.

1 Introdução

A **regressão linear** é uma técnica estatística e de aprendizado de máquina que modela a relação entre uma *variável dependente* (também chamada de variável alvo ou de saída) e uma ou mais *variáveis independentes* (também conhecidas como preditores, atributos ou features). Em termos simples, a regressão linear tenta ajustar uma linha reta (no caso de uma variável independente) ou um hiperplano (no caso de múltiplas variáveis) aos dados de

forma que essa linha represente da melhor maneira possível a tendência dos pontos observados. O objetivo é poder usar essa relação linear estimada para prever o valor da variável dependente para novos dados.

A importância da regressão linear em *Machine Learning* decorre de sua simplicidade, interpretabilidade e eficácia em muitos problemas práticos. Por ser um dos algoritmos mais simples, serve frequentemente como ponto de partida para problemas de regressão, oferecendo um baseline (referência básica) de desempenho. Além disso, os coeficientes obtidos pelo modelo linear fornecem insights sobre a influência de cada variável independente no resultado, algo valioso para interpretabilidade do modelo. A regressão linear tem sido amplamente utilizada por décadas em estatística e econometria, e no contexto de ciência de dados e ML continua sendo relevante, seja como modelo primário, seja como componente de modelos mais complexos.

Este artigo abordará a regressão linear de forma abrangente. Na Seção 2, apresentamos os conceitos fundamentais da regressão linear, incluindo sua formulação matemática e o método dos mínimos quadrados para cálculo dos coeficientes. Discutiremos também as vantagens e desvantagens desse método (Seção 3), fornecendo uma visão crítica de quando a regressão linear é apropriada ou não. Na Seção 4, exploraremos os principais tipos e variações da regressão linear, como a regressão linear simples versus múltipla, e técnicas de regularização como Ridge e Lasso, incluindo suas fórmulas. Em seguida, na Seção 5, veremos exemplos de aplicações reais da regressão linear, ilustrando como problemas do cotidiano (previsão de preços, salários, consumo etc.) podem ser tratados por esse modelo. Na Seção 6, apresentaremos implementações práticas em Python, utilizando bibliotecas populares (NumPy, scikit-learn, matplotlib) para construir e visualizar modelos de regressão linear. Por fim, na Seção 7, concluímos o artigo resumindo os pontos principais e oferecendo recomendações para quem deseja aprofundar-se no tema.

2 Conceitos Básicos e Formulação Matemática

2.1 Regressão Linear Simples

Em uma regressão linear simples, temos uma única variável independente x explicando uma variável dependente y. A relação é modelada pela equação de uma reta:

$$y = \beta_0 + \beta_1 x + \varepsilon, \tag{1}$$

onde β_0 é o intercepto (coeficiente linear, correspondente ao valor de y quando $x = 0$), β_1 é o coeficiente angular (que representa a inclinação da reta, isto é, o quanto y varia a cada unidade de aumento em x) e ε representa o erro (ou termo de ruído), que captura as discrepâncias entre os valores previstos pelo modelo e os valores reais observados. O objetivo do modelo é encontrar os valores de β_0 e β_1 que produzam a *melhor reta* ajustada aos dados.

Para ilustrar, suponha que queiramos prever o preço de venda de um imóvel (y) usando apenas a área construída (x) como preditor. Uma regressão linear simples modelaria algo como:

$$\text{preço} = \beta_0 + \beta_1 \times \text{área}.$$

Se β_1 for positivo, indica que imóveis maiores tendem a valer mais, e β_0 seria o preço base (teoricamente o preço de um imóvel de área zero, que aqui serve apenas como referência para ajustar a reta).

2.2 Regressão Linear Múltipla

Na *regressão linear múltipla*, estendemos o conceito para múltiplas variáveis independentes. Em vez de uma reta num plano 2D, estamos ajustando um *hiperplano* em um espaço de dimensões mais altas. A equação geral de um modelo linear com n variáveis explicativas é:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon, \tag{2}$$

onde $x_1, x_2, \ldots, x_n$ são as variáveis independentes (preditoras) e $\beta_1, \beta_2, \ldots, \beta_n$ são os coeficientes associados a cada uma delas. Assim como no caso simples, β_0 é o intercepto e ε é o termo de erro.

Por exemplo, para prever o preço de um imóvel, poderíamos utilizar múltiplos fatores: $y = \beta_0 + \beta_1(\text{área}) + \beta_2(\text{número de quartos}) + \beta_3(\text{idade do imóvel}) + \cdots + \varepsilon$. Cada coeficiente β_i indicará o impacto estimado daquela variável no preço, mantendo as demais constantes. A regressão múltipla permite incorporar mais informações no modelo, potencialmente melhorando as previsões se as variáveis adicionais forem relevantes.

2.3 Ajuste do Modelo por Mínimos Quadrados

Para determinar os coeficientes β que fornecem o melhor ajuste, a abordagem mais comum é o *método dos mínimos quadrados ordinários* (em inglês, *Ordinary Least Squares* ou OLS). A ideia central do OLS é escolher $\beta_0, \beta_1, \ldots, \beta_n$

de forma a minimizar a soma dos quadrados dos *resíduos* (diferenças entre os valores observados y_i e os valores previstos $\hat{y}_i$ pelo modelo).

Matematicamente, definimos a *função de custo* $J(\beta_0, \beta_1, \ldots, \beta_n)$ como:

$$J(\beta_0, \ldots, \beta_n) = \sum_{i=1}^{m} \left(y_i - \hat{y}_i\right)^2, \tag{3}$$

onde m é o número de observações (pontos de dados), y_i é o valor real da variável dependente na i-ésima observação, e $\hat{y}_i$ é o valor previsto pelo modelo para essa observação. Em geral, utilizamos a média dos quadrados (Erro Quadrático Médio, ou EQM) para facilitar comparações, mas como m é constante para um dado conjunto de dados, minimizar a soma dos quadrados ou a média dos quadrados resulta nos mesmos coeficientes ótimos.

O objetivo é encontrar os parâmetros β que minimizam J. No caso da regressão linear (sem termos de regularização, discutidos posteriormente), existe uma solução analítica direta. Para a regressão linear simples, podemos obter fórmulas fechadas para β_1 e β_0:

$$\beta_1 = \frac{\sum_{i=1}^{m}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{m}(x_i - \bar{x})^2}, \tag{4}$$

$$\beta_0 = \bar{y} - \beta_1 \bar{x}, \tag{5}$$

onde $\bar{x}$ e $\bar{y}$ são as médias de x e y no conjunto de dados. A Equação (4) mostra que β_1 é proporcional à covariância entre x e y dividida pela variância de x – em outras palavras, o coeficiente angular será maior se houver uma forte associação positiva (covariância alta) entre x e y, e será próximo de zero se não houver tendência linear.

No caso geral (regressão múltipla), podemos expressar o problema em notação matricial. Seja X a matriz $m \times (n+1)$ que contém uma coluna de 1's (para o termo intercepto) e as colunas com os valores das n variáveis independentes para cada uma das m observações, e seja $\mathbf{y}$ o vetor coluna dos m valores observados de y. Os coeficientes podem ser organizados em um vetor $\beta = [\beta_0, \beta_1, ..., \beta_n]^T$. A solução que minimiza a soma dos quadrados dos resíduos é dada pela *equação normal*:

$$\hat{\beta} = (X^T X)^{-1} X^T \mathbf{y}. \tag{6}$$

Essa fórmula fornece diretamente os coeficientes que minimizam a função de custo quadrático. É importante notar que essa solução exige que $X^T X$ seja

inversível (não singular). Na prática, isso significa que não pode haver colinearidade perfeita entre as variáveis independentes, caso contrário a matriz torna-se não invertível. Quando há problemas de colinearidade ou quando o número de variáveis é muito grande em relação ao número de observações, técnicas de regularização (ver Seção 4) ou métodos iterativos podem ser necessários.

2.4 Resolução por Otimização Iterativa

Além da solução analítica via equação normal, outra maneira de encontrar os coeficientes é utilizando métodos de otimização iterativa, sendo o mais comum o *gradiente descendente*. Nessa abordagem, iniciamos com valores arbitrários para $\beta_0, \beta_1, \ldots, \beta_n$ e iterativamente ajustamos esses valores na direção que reduz a função de custo. Em particular, calcula-se o gradiente (vetor de derivadas parciais de J em relação a cada β_j) e atualiza-se cada parâmetro no sentido oposto ao gradiente (por isso "descendente"):

$$\beta_j := \beta_j - \alpha \frac{\partial J}{\partial \beta_j},$$

onde α é a taxa de aprendizado (um hiperparâmetro que controla o tamanho do passo de atualização). Para a função de custo quadrático, a derivada em relação a β_j é dada por:

$$\frac{\partial J}{\partial \beta_j} = -2 \sum_{i=1}^{m} (y_i - \hat{y}_i) x_{ij},$$

(para $j = 0$, $x_{i0} = 1$ porque β_0 acompanha o intercepto). Substituindo na atualização, temos que, a cada iteração:

$$\begin{cases} \beta_0 := \beta_0 - \frac{2\alpha}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i), \\ \beta_j := \beta_j - \frac{2\alpha}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i) x_{ij}, \quad j = 1, 2, \ldots, n. \end{cases}$$

O gradiente descendente continua ajustando os parâmetros até que a função de custo alcance um mínimo (ou até que as atualizações se tornem muito pequenas). Embora o método dos mínimos quadrados com solução analítica seja exato e rápido para conjuntos de dados relativamente pequenos ou medianos, o gradiente descendente é útil em cenários de grande escala (muitos dados ou muitas variáveis) onde a inversão de $(X^T X)$ pode ser inviável computacionalmente ou quando se deseja atualizar o modelo continuamente com novos dados (aprendizado online).

3 Vantagens e Desvantagens da Regressão Linear

3.1 Vantagens

A regressão linear possui diversas qualidades que explicam sua popularidade como primeiro modelo a ser considerado em problemas de previsão:

- **Simplicidade e interpretabilidade**: É fácil compreender o funcionamento do modelo e interpretar seus resultados. A equação linear fornece uma relação direta entre entradas e saída, permitindo entender o impacto de cada variável independente no valor predito. Isso torna a regressão linear intuitiva, mesmo para iniciantes.

- **Treinamento eficiente**: Ajustar um modelo de regressão linear (especialmente via equação normal) é computacionalmente rápido para conjuntos de dados de tamanho moderado. Mesmo para grandes volumes de dados, métodos como gradiente descendente escalam bem. Assim, é um algoritmo eficiente em termos de tempo de treinamento.

- **Boa performance em relações aproximadamente lineares**: Quando a suposição de linearidade é ao menos aproximadamente válida, a regressão linear tende a fornecer previsões precisas. Em muitos problemas do mundo real, uma primeira aproximação linear já captura parte significativa da relação entre variáveis.

- **Pouca parametrização**: Diferente de alguns algoritmos de machine learning mais complexos, a regressão linear pura não exige a escolha de muitos hiperparâmetros (além, talvez, da taxa de aprendizado se usado gradiente descendente). Isso simplifica o processo de modelagem inicial.

- **Base para extensões**: Muitos modelos mais complexos se baseiam ou estendem a ideia de relações lineares. Por exemplo, redes neurais sem funções de ativação não passam de uma regressão linear múltipla; algoritmos regularizados (como veremos) adicionam penalidades à função de custo da regressão linear. Portanto, entender regressão linear ajuda na compreensão de uma série de outros modelos.

3.2 Desvantagens

Por outro lado, é importante conhecer as limitações e potenciais problemas ao aplicar regressão linear:

- **Suposição de linearidade**: O modelo presume que a relação entre as variáveis independentes e a variável dependente é linear. Caso exista um relacionamento não linear pronunciado nos dados, a regressão linear simples não conseguirá capturá-lo adequadamente, levando a erros sistemáticos (nesse caso, outras abordagens ou transformações de variáveis seriam necessárias).

- **Sensibilidade a outliers**: A minimização do erro quadrático enfatiza discrepâncias grandes (devido ao termo ao quadrado). Portanto, observações atípicas (outliers) podem influenciar fortemente os coeficientes ajustados, distorcendo a reta de regressão. É recomendável analisar e possivelmente tratar outliers antes de ajustar o modelo, ou utilizar métodos robustos quando apropriado.

- **Extrapolação limitada**: Modelos lineares tendem a não se comportar bem fora do intervalo de dados usado para treiná-los. Como a relação é fixa e linear, usar o modelo para extrapolar muito além do range observado de variáveis independentes pode produzir previsões irreais, pois não captura possíveis mudanças de tendência.

- **Requer validação de pressupostos**: Para que as inferências clássicas (intervalos de confiança dos coeficientes, testes de significância etc.) da regressão linear sejam válidas, pressupõe-se algumas condições sobre os resíduos: homocedasticidade (variância constante do erro), normalidade dos erros e independência das observações. Se esses pressupostos forem violados, a qualidade do ajuste ou a interpretação estatística pode ficar comprometida.

- **Possibilidade de sobreajuste com muitas variáveis**: Embora um modelo linear seja relativamente simples, se incluirmos variáveis demais (especialmente variáveis irrelevantes ou altamente correlacionadas entre si), o modelo pode se tornar excessivamente ajustado ao conjunto de treinamento, perdendo capacidade preditiva em novos dados. Nesses casos, técnicas de regularização ou seleção de variáveis são úteis.

- **Incapacidade de modelar relações complexas**: Problemas que envolvem interações complexas entre variáveis ou padrões altamente não lineares geralmente não são bem representados por uma única expressão linear. Modelos mais flexíveis (como árvores de decisão, florestas aleatórias, redes neurais, etc.) podem capturar melhor essas nuances, embora frequentemente às custas de menor interpretabilidade.

4 Principais Tipos e Variações de Regressão Linear

Há diversas variações e extensões do modelo de regressão linear para lidar com diferentes situações. Aqui destacamos as principais:

4.1 Regressão Linear Simples vs. Múltipla

Já apresentamos esses dois conceitos, mas para resumir: a regressão linear **simples** envolve apenas uma variável preditora, enquanto a regressão linear **múltipla** envolve duas ou mais variáveis preditoras. Conceitualmente, a diferença está no tipo de relação geométrica (reta em 2 dimensões para a simples, plano ou hiperplano em dimensões superiores para a múltipla) e na quantidade de coeficientes a estimar. Em termos matemáticos:

Simples: $y = \beta_0 + \beta_1 x_1 + \varepsilon$; Múltipla: $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$.

Os métodos de ajuste (mínimos quadrados, gradiente descendente) discutidos anteriormente se aplicam a ambos os casos, sendo a versão múltipla apenas uma generalização direta da simples.

4.2 Regressão Polinomial

Embora o termo sugira um tipo distinto de modelo, a **regressão polinomial** nada mais é do que uma regressão linear múltipla onde criamos variáveis preditoras que são potências de uma variável original. Por exemplo, para modelar uma relação quadrática entre x e y, podemos definir $x_1 = x$ e $x_2 = x^2$ e então ajustar um modelo linear múltiplo $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$. De forma geral, um modelo polinomial de grau d para uma única variável x seria:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_d x^d + \varepsilon,$$

que continua sendo linear nos parâmetros β (apesar de não ser linear em x). A regressão polinomial é útil quando a relação entre x e y é não-linear, mas pode ser bem aproximada por um polinômio. Entretanto, deve-se ter cuidado: à medida que aumentamos muito o grau do polinômio, o modelo se torna excessivamente flexível e pode levar a *sobreajuste* (overfitting), ajustando até mesmo o ruído dos dados. A escolha do grau apropriado costuma requerer validação (por exemplo, usando dados de validação ou avaliação de métricas como o erro em dados não usados no ajuste).

4.3 Regressão Ridge (Regularização L2)

A **regressão Ridge** é uma extensão do modelo linear que incorpora um termo de *regularização L2* à função de custo. Regularização é uma técnica usada para evitar sobreajuste adicionando uma penalização para coeficientes grandes. No caso da Ridge, penalizamos o quadrado dos coeficientes. A nova função de custo a ser minimizada passa a ser:

$$J_{\text{ridge}}(\beta_0, \ldots, \beta_n) = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2, \tag{7}$$

onde $\lambda \geq 0$ é um *hiperparâmetro* que controla a intensidade da penalização. Note que a soma da penalização exclui β_0 (intercepto), já que não é desejável penalizar um termo de bias. Quando $\lambda = 0$, recuperamos a regressão linear tradicional (sem penalização). Conforme λ aumenta, os coeficientes β_j são forçados a valores menores (mais próximos de zero), o que geralmente reduz a variância do modelo ao custo de introduzir um viés adicional. A Ridge tende a funcionar bem em situações com muitas variáveis correlacionadas ou quando n (número de features) é próximo de m (número de observações), situações em que a regressão linear padrão poderia ter coeficientes instáveis. Inclusive, a solução analítica da regressão Ridge tem uma forma fechada similar à equação normal:

$$\hat{\beta}^{\text{ridge}} = (X^T X + \lambda I)^{-1} X^T \mathbf{y},$$

onde I é a matriz identidade $(n + 1 \times n + 1)$ (com o primeiro elemento correspondente a β_0 geralmente não penalizado). Essa fórmula mostra que a regularização L2 equivale a adicionar um valor λ nas diagonais da matriz $X^T X$ antes de invertê-la, o que a torna sempre invertível mesmo quando $X^T X$ original não seria (resolvendo problemas de multicolinearidade perfeita, ainda que à custa de um pequeno viés).

4.4 Regressão Lasso (Regularização L1)

A **regressão Lasso** (do acrônimo em inglês *Least Absolute Shrinkage and Selection Operator*) usa um tipo diferente de regularização: a *regularização L1*, em que a penalidade adicionada à função de custo é proporcional ao valor absoluto dos coeficientes. A função de custo do Lasso é:

$$J_{\text{lasso}}(\beta_0, \ldots, \beta_n) = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} |\beta_j|, \tag{8}$$

Assim como na Ridge, λ controla a força da penalização (e não penalizamos β_0). Diferentemente da Ridge, não existe fórmula analítica simples para os coeficientes ótimos do Lasso – a solução é obtida via métodos numéricos (como algoritmos de coordenação cíclica). A principal característica do Lasso é que ele pode reduzir coeficientes exatamente a zero quando λ é suficientemente grande, efetivamente *eliminando* variáveis irrelevantes do modelo. Por isso, o Lasso realiza uma espécie de seleção de atributos automática: é um modelo útil quando se suspeita que apenas um subconjunto das variáveis explicativas tem contribuição significativa. Por outro lado, em cenários com muitas variáveis altamente correlacionadas, o Lasso pode apresentar instabilidades, pois tende a escolher arbitrariamente uma dentre as variáveis correlacionadas para atribuir coeficiente diferente de zero, ignorando as demais.

4.5 Elastic Net

O **Elastic Net** é uma técnica de regressão que combina os termos de regularização L1 e L2. Ele foi proposto para unir os benefícios de Ridge e Lasso, mitigando algumas de suas limitações individuais. A função de custo do Elastic Net adiciona ambas as penalidades:

$$J_{\text{EN}}(\beta) = \sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \left[\alpha \sum_{j=1}^{n} |\beta_j| + (1 - \alpha) \sum_{j=1}^{n} \beta_j^2 \right], \tag{9}$$

onde $\lambda \geq 0$ controla a força total da regularização e $0 \leq \alpha \leq 1$ controla o balanço entre L1 e L2. Quando $\alpha = 1$, o Elastic Net se reduz ao Lasso; quando $\alpha = 0$, se reduz ao Ridge. Valores intermediários de α produzem um efeito misto. Na prática, o Elastic Net costuma ser uma escolha eficaz quando há muitas variáveis e se deseja tanto estabilidade (oferecida pela L2)

quanto seletividade de variáveis (oferecida pela L1). A desvantagem é ter dois hiperparâmetros para ajustar (λ e α), mas isso pode ser feito via validação cruzada.

5 Exemplos de Uso em Problemas Reais

A regressão linear é aplicada em inúmeras situações do dia a dia para realizar previsões ou inferir relações entre variáveis. A seguir, descrevemos alguns exemplos típicos em que esse modelo pode ser empregado com sucesso:

- **Previsão de preços de imóveis:** Esse talvez seja um dos casos mais clássicos. A partir de dados históricos sobre casas ou apartamentos vendidos (com informações como área em metros quadrados, número de quartos, localização, idade do imóvel, etc.), pode-se ajustar um modelo de regressão linear para prever o preço de venda de um novo imóvel. Cada variável independente representa um atributo do imóvel e o valor predito (variável dependente) é o preço estimado. Por exemplo, o modelo poderia indicar que, a cada metro quadrado adicional, espera-se um aumento de certo valor no preço, mantendo os demais fatores constantes. Esse tipo de modelo auxilia imobiliárias e compradores/vendedores a estimarem o valor justo de propriedades.

- **Estimativa de salários:** Em recursos humanos e economia do trabalho, costuma-se usar regressão para entender e prever salários com base em características dos empregados. As variáveis independentes podem incluir anos de experiência, nível de escolaridade (por exemplo, graduação, mestrado), área de atuação, idade, entre outros fatores. O modelo de regressão linear forneceria um salário estimado (variável dependente) para um perfil específico. Por exemplo, poderia-se concluir que cada ano adicional de experiência acrescenta em média um certo valor ao salário, ou que determinadas certificações têm impacto positivo mensurável. Esse tipo de análise é útil tanto para profissionais avaliarem expectativas salariais quanto para empresas definirem faixas salariais competitivas.

- **Projeção de consumo de energia:** Empresas de fornecimento de energia elétrica utilizam regressão linear para prever a demanda futura de energia com base em variáveis como temperatura ambiente, hora do

dia, dia da semana, estação do ano e consumo passado. O objetivo é conseguir antecipar picos de consumo e planejar a geração ou compra de energia de forma eficiente. Por exemplo, em dias de temperatura muito alta (ou muito baixa, dependendo da região), o consumo tende a subir devido ao uso de ar-condicionado ou aquecimento, e o modelo linear pode quantificar essa relação para melhorar as previsões. Ainda que fenômenos de consumo possam ser complexos, modelos lineares servem como um ponto de partida simples e muitas vezes eficaz para previsões de curto prazo nesse setor.

Esses são apenas alguns cenários. A regressão linear também é empregada em áreas como finanças (por exemplo, para relacionar indicadores econômicos e prever vendas ou lucros), marketing (estimando o impacto de investimentos em publicidade nas vendas), engenharia (modelando relações entre variáveis de projeto e desempenho de um sistema), entre muitos outros. Sempre que houver uma suspeita de relação aproximadamente linear entre uma variável alvo e um conjunto de fatores explicativos, a regressão linear é uma ferramenta a ser considerada.

6 Implementações Práticas em Python

Para consolidar os conceitos, vamos construir um pequeno exemplo prático usando Python. Demonstramos como realizar uma regressão linear simples tanto "manualmente" (usando NumPy para cálculos matriciais) quanto utilizando a biblioteca scikit-learn, e por fim visualizamos o resultado com um gráfico.

6.1 Exemplo: Ajustando um Modelo Linear Simples

No exemplo a seguir, geraremos um conjunto de dados sintético no qual há uma relação linear clara entre x (variável independente) e y (variável dependente), com algum ruído aleatório. Em seguida, calcularemos os coeficientes de regressão linear de duas formas: primeiro usando a fórmula analítica (Equação 6) via operações do NumPy, e depois usando a classe `LinearRegression` do scikit-learn. Por fim, exibiremos os coeficientes encontrados e um gráfico com os pontos de dados e a reta de regressão ajustada.

```
import numpy as np
```

```
# 1. Gerando dados sint ticos (y = 5 + 2*x + ru do)
np.random.seed(42)
m = 50  # n mero de pontos
X = 10 * np.random.rand(m)         # x aleat rio entre 0
    e 10
y = 5 + 2 * X + np.random.normal(scale=3, size=m)  # y =
    5 + 2x + ru do gaussiano

# 2. Ajuste manual via f rmula normal (beta = (X^T X)^-1
     X^T y)
# Para usar a f rmula matricial, montamos X design
    matrix com coluna de 1s
X_design = np.vstack([np.ones(m), X]).T  # matriz m x 2
    (1 para intercepto, e X)
# Calcula beta_chapeu = (X^T X)^{-1} X^T y
beta_hat = np.linalg.inv(X_design.T.dot(X_design)).dot(
    X_design.T).dot(y)
beta0_manual, beta1_manual = beta_hat[0], beta_hat[1]

# 3. Ajuste usando scikit-learn
from sklearn.linear_model import LinearRegression
model = LinearRegression(fit_intercept=True)
X_reshaped = X.reshape(-1, 1)  # reshape para matriz m x
    1
model.fit(X_reshaped, y)
beta0_sklearn = model.intercept_
beta1_sklearn = model.coef_[0]

# 4. Exibindo coeficientes obtidos
print(f"Coeficiente␣beta0␣(intercepto)␣-␣manual:␣{
    beta0_manual:.3f}")
print(f"Coeficiente␣beta1␣(inclina  o)␣-␣manual:␣{
    beta1_manual:.3f}")
print(f"Coeficiente␣beta0␣-␣scikit-learn:␣{beta0_sklearn
    :.3f}")
print(f"Coeficiente␣beta1␣-␣scikit-learn:␣{beta1_sklearn
    :.3f}")

# 5. Visualiza  o dos dados e do modelo ajustado
```

```
import matplotlib.pyplot as plt
plt.scatter(X, y, color='blue', label='Dados␣(x,y)')
# reta ajustada usando coeficientes do modelo:
x_vals = np.linspace(0, 10, 100)
y_vals = beta0_sklearn + beta1_sklearn * x_vals
plt.plot(x_vals, y_vals, color='red', label='Reta␣
   ajustada')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Regress o␣Linear␣Simples␣-␣Dados␣Sint ticos'
   )
plt.legend()
plt.show()
```

No código acima, detalhamos cada etapa com comentários:

1. Geramos $m = 50$ pontos (x, y) seguindo aproximadamente a relação $y = 5 + 2x +$ ruído. Em particular, $\beta_0 = 5$ e $\beta_1 = 2$ são os valores "verdadeiros" usados na geração dos dados, e adicionamos um ruído aleatório (distribuição normal com desvio padrão 3) para tornar o problema mais realista.

2. Montamos a matriz de design X_{design} com uma coluna de 1's e uma coluna com os valores de x. Usamos a fórmula de mínimos quadrados $\hat{\beta} = (X^T X)^{-1} X^T y$ para calcular β_0 e β_1 (armazenando-os em `beta0_manual`, `beta1_manual`). Esse é o cálculo "manual" usando operações matriciais do NumPy.

3. Utilizamos `LinearRegression` do *scikit-learn* para ajustar o mesmo modelo. Após treinar (`.fit`), extraímos `model.intercept_` como β_0 e `model.coef_`[0] como β_1. Esses valores devem ser muito próximos aos calculados manualmente.

4. Em seguida, imprimimos os coeficientes obtidos por ambos os métodos para verificar se coincidem (ou são bem próximos, sujeitando-se a pequenas diferenças numéricas).

5. Finalmente, usamos `matplotlib` para plotar os pontos de dados em um gráfico de dispersão (scatter plot) e desenhar a reta de regressão resultante em vermelho. A função `plt.show()` exibirá o gráfico.

Se executarmos o código acima, obteremos coeficientes estimados próximos de $\beta_0 \approx 5.3$ e $\beta_1 \approx 1.93$, comparáveis aos valores gerados (5 e 2). A pequena diferença se deve ao efeito do ruído nos dados. A saída gráfica exibirá algo como o ilustrado na figura a seguir (pontos azuis representando os dados simulados e a linha vermelha representando a reta ajustada pelo modelo linear).

7 Conclusão

Neste artigo, exploramos em detalhes o conceito de regressão linear aplicada a problemas de aprendizado de máquina. Iniciamos pelos fundamentos, entendendo a forma básica do modelo (simples e múltiplo) e como seus coeficientes são determinados pelo método de mínimos quadrados. Discutimos as vantagens desse modelo – em especial sua simplicidade, interpretabilidade e eficiência – bem como suas limitações e suposições subjacentes. Apresentamos também variações importantes, incluindo técnicas de regularização (Ridge, Lasso e Elastic Net) que estendem a regressão linear para lidar melhor com sobreajuste e seleção de atributos. Por meio de exemplos práticos, vimos como a regressão linear pode ser aplicada em diversos domínios para realizar previsões (de preços, salários, consumo, etc.), tornando-se uma ferramenta extremamente versátil.

Também demonstramos uma implementação prática em Python, mostrando que ferramentas como NumPy e scikit-learn facilitam tanto o cálculo manual dos coeficientes quanto o uso de algoritmos já otimizados, além de possibilitar rápida visualização dos resultados.

Em suma, a regressão linear serve tanto como um ponto de partida acessível para iniciantes quanto como um componente fundamental em soluções de machine learning mais complexas. Entender bem seus mecanismos, vantagens e limitações é essencial para qualquer pessoa que se inicia na ciência de dados ou modelagem preditiva. A partir dessa base, o leitor está apto a aprofundar-se em tópicos correlatos, como análise de resíduos para verificação de pressupostos, regressões não-lineares e outros algoritmos de regressão mais avançados, sabendo que muitos dos conceitos aprendidos aqui servirão de alicerce.