Aula 3

1 O que é regressão linear

A regressão linear é um dos algoritmos mais clássicos de aprendizado de máquina. Sua função é encontrar um modelo que é uma combinação linear das features do exemplo de entrada, com o objetivo de encontrar o valor provável para um novo exemplo.

1.0.1 Como funciona

Dada uma coleção de exemplos rotulados $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, onde N é o tamanho da coleção, $\mathbf{x}_i$ é o vetor de características D-dimensional do exemplo $i = 1, \ldots, N$, y_i é o valor relacionado ao vetor de características e cada característica $x_i^{(j)}$, $j = 1, \ldots, D$, também é um número real.

Queremos construir um modelo $f_{w,b}(\mathbf{x})$ como uma combinação linear das características do exemplo $\mathbf{x}$:

$$f_{w,b}(\mathbf{x}) = \mathbf{w}\mathbf{x} + b,$$

onde w é um vetor de parâmetros com D dimensões e b é um número real. Portanto, o modelo é parametrizado por dois valores: w e b.

Para um dado valor y novo, o modelo retornará um valor. Para dois modelos com w e b diferentes, a predição muda. O objetivo é encontrar os valores ótimos para w e b, que podemos denotar como w* e b*. Com esses valores, é esperada uma maior acurácia de predição.

2 Exemplo: Relação entre Tamanho da Casa e Preço de Venda

Suponha que uma imobiliária deseja modelar a relação entre o **tamanho do imóvel** (em m²) e seu **preço de venda** (em R$ mil). Foram coletados os seguintes dados de 6 propriedades:

Tabela 1: Dados de Tamanho e Preço de Imóveis

Casa	**Tamanho (m²)** (X)	**Preço (R$ mil)** (Y)
1	50	300
2	70	380
3	80	420
4	90	470
5	100	530
6	120	600

2.1 Modelo de Regressão Linear

O modelo de regressão linear simples assume a relação:

$$Y = aX + b + \epsilon \tag{1}$$

Onde:

- a = coeficiente angular (quanto Y aumenta para cada unidade de X)
- b = intercepto (valor de Y quando X=0)
- ϵ = erro aleatório

2.2 Ajuste do Modelo

Achando os parâmetros da regressão lineares teremos:

$$\hat{Y} = 5X + 50 \tag{2}$$

Interpretação:

- Cada metro quadrado adicional aumenta o preço em R$ 5 mil
- O valor base (para X=0) é R$ 50 mil

2.3 Previsão

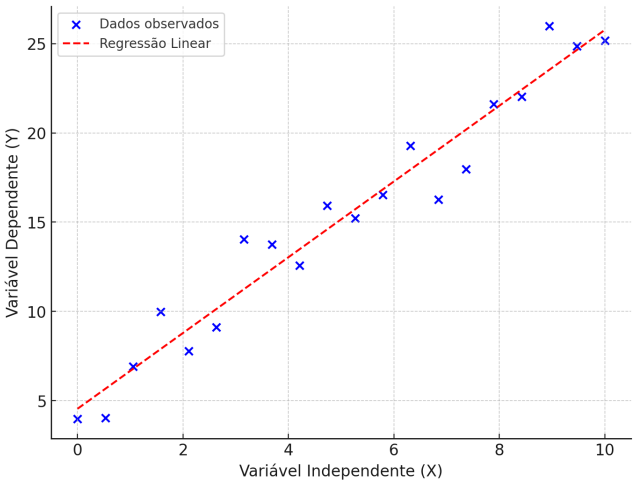
Para uma casa com 110 m²:

$$\hat{Y} = 5 \times 110 + 50 = 600 \text{ mil reais} \tag{3}$$

2.4 Diferença para Classificação

Suponha que somos uma imobiliária e queremos prever o preço de venda de uma casa com base no seu tamanho em metros quadrados (m²). Coletamos dados de 6 casas vendidas recentemente:

Você pode ter notado que a forma desse modelo linear é bem parecida com a forma do classificador linear. A principal diferença está no papel do hiperplano que o modelo descreve. Enquanto que no classificador linear queremos separar dois grupos de exemplos, o objetivo do hiperplano na regressão linear é estar o mais próximo possível dos exemplos de treinamento. Um exemplo de regressão linear está disposto na figura a seguir:



Neste exemplo, temos uma regressão linear para exemplos de dimensão 1. Para exemplos de dimensão 2, teríamos um plano ao invés de uma linha, e para exemplos em que D > 2, teríamos um hiperplano.

3 Empirical Risk

Até o momento, com o objetivo de quantificar quão errada era uma predição, utilizamos uma função de perda. Porém, e se quiséssemos quantificar a perda em relação a todo o conjunto de treinamento do modelo? Para esse tipo de tarefa, usaremos uma função semelhante a loss function: o empirical risk. O empirical risk, dada uma amostra de treinamento de n dados, um modelo $f(x_i)$ e uma loss $L(f(x_i), y_i)$, é definido como:

$$R_n(f(x_i)) = \frac{1}{n}\sum_{i=1}^{n} L(f(x_i), y_i)).$$

Note que temos, em suma, o cálculo da média aritmética das perdas. A partir disso, podemos, então, utilizar uma função de perda e, então, calcular o empirical risk. Dentre as funções de perda mais comuns a serem usadas, temos o erro quadrático médio (mean squared error ou MSE), que será tema do próximo capítulo.

4 Mean Squared Error

O erro quadrático é uma das principais loss functions que podem ser utilizadas. Ela pode ser escrita da forma:

$$L(f(x_i, \theta), y_i) = \frac{(y_i - f(x_i))^2}{2} = \frac{(y_i - \theta x_i)^2}{2}.$$

Observe que essa loss tem como característica intensificar o erro encontrado, seja para menos ou para mais. Para valores pequenos de loss, ao calcularmos o quadrado, teremos como resultado um número muito próximo da loss ou, até mesmo, menor. Já para valores grandes de loss, o resultado da operação será maior do que o valor base. Com isso, priorizamos ainda mais a ideia de minimizar a perda.

A partir do MSE, podemos escrever, então, a função empirical risk como:

$$R_n(\theta) = \frac{1}{n}\sum_{i=1}^{n} \frac{(y^t - \theta x^t)^2}{2}.$$

5 Algoritmo de regressão linear com Gradiente e regularização

Assim como na aula passada, podemos utilizar a ideia do gradiente para minimizar a nossa perda. Dado um único exemplo aleatório do conjunto de treinamento, calculamos o gradiente de sua perda da seguinte forma:

$$\nabla L(f(x_i, y_i)) = \frac{\partial L(f(x_i, y_i))}{\partial \theta} = \frac{\frac{\partial (y_i - \theta x_i)^2}{2}}{\partial \theta} = -x_i(y_i - \theta x_i).$$

Como devemos, a cada observação de um dado, "andar"na direção contrária do gradiente, atualizaremos sempre com o termo $+x_i(y_i - \theta x_i)$. A partir disso, com uma taxa de aprendizado η, temos o seguinte algoritmo:

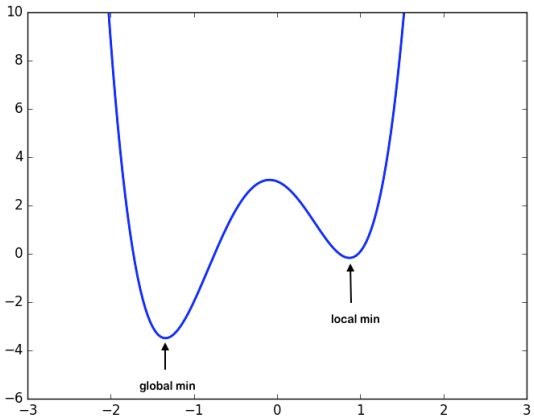
1. Inicializar $\theta = 0$
2. Tomar randomicamente um $t = 1, ..., n$ (4)
3. $\theta = \theta + \eta x^t(y^t - \theta x^t)$

6 Closed Form Solution

6.1 Convexidade de Funções

Em Machine Learning, como o nosso objetivo básico é encontrar o ponto ótimo de minimização para nossas funções de Loss, vale a pena perguntar: não há em alguns casos alguma fórmula que possamos usar? Por quê sempre devemos "adivinhar"onde esse ponto ótimo está com métodos iterativos (tal qual a descida de Gradiente), ao invés de simplesmente jogar numa fórmula? Bem, a resposta é que as funções de Loss raramente tomam um formato em que possamos encontrar uma solução analítica simples que resolva.

Mesmo caso nós tentemos usar a descida de Gradiente para treinar nossos modelos, muitas vezes ela tem dificuldade de achar o mínimo global da nossa função de Loss. Para um exemplo simples, imagine que o algoritmo fique "preso"no menor pico dessa função, ao invés de explorar mais e atingir o mínimo global:



Todavia, há uma classe de funções relativamente bem comportadas para as quais a descida de gradiente sempre funciona: as funções convexas.

Formalmente, definimos funções convexas como funções em que a segunda derivada, chamada de Hessiana para dimensões maiores que 2, possui auto-vetores maiores que 0, isto é:

$$H(f) \rightarrow \text{autovetores}\left(\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}\right) \geq 0 \quad (5)$$

Não é necessário entender nada disso: apenas imagine que essa condição determina que nossa função vai sempre ter um mínimo global que podemos atingir, tal qual um "barranco"que escorregamos por meio da gradient descent. Desse modo, funções convexas sempre são ótimas para aprendizado de máquina.

Calculando a Hessiana da nossa Loss Function (MSE), é possível provar matematicamente que seus autovetores são maiores que 0, indicando sua convexidade, o que implica um cenário ideal para descida de gradiente (como descrevemos). Isso fica muito óbvio quando analisamos a natureza quadrática dela:

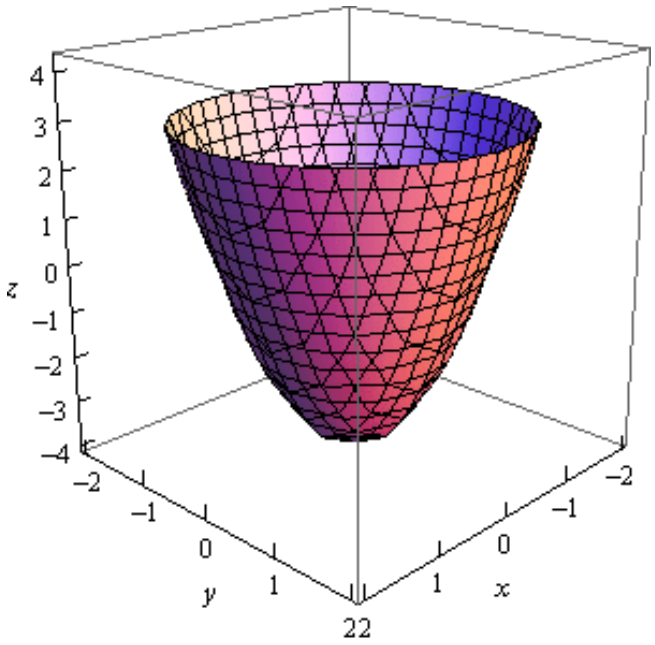


Figura 1: Representação Didática da Loss Function $L(\theta) = \|Y - X\theta\|^2$. Observe que há um mínimo global apenas, e em qualquer ponto do gráfico o gradiente aponta para cima; invertê-lo leva em direção a esse mínimo

É importante ter essa noção sobre a convexidade de funções para podermos ajustar nossos projetos de machine learning conforme necessário: sua Loss ser ou não convexa deve ajustar suas expectativas sobre o quão bem o gradient descent pode funcionar. No caso da regressão linear, por exemplo, podemos ter certeza de que ele funcionará perfeitamente, graças a esse atributo.

6.2 Solução Fechada

Voltando ao primeiro parágrafo dessa seção (sobre soluções analíticas para nossos problemas de otimização), o caso da regressão linear é um dos raros em que existe uma fórmula para calcular os coeficientes perfeitos para o curve fitting. Vamos deduzi-la agora.

Utilizaremos o método clássico que se ensina em cálculo I para esse fim: igualar a segunda derivada a 0. Isso funcionará, pois a convexidade garante a existência de um único ponto de mínima em nossa função. Para esse fim, vamos utilizar ferramentas de cálculo matricial (não se assuste! Apenas aceite que é possível derivar vetores e matrizes em relação a outros vetores e matrizes).

Seja Y o vetor de labels e X nossa matriz de treinamento, cujas linhas são os exemplos associados às labels, e as colunas, as features. Desse modo, como já estabelecido, temos:

$$L(\theta) = \|Y - X\theta\|^2 \tag{6}$$

Derivando em relação a θ, usaremos uma identidade famosa em cálculo matricial, de modo que:

$$\frac{\partial L}{\partial \theta} = \frac{\partial((Y - X\theta)^T(Y - X\theta))}{\partial \theta} \tag{7}$$

$$= 2(Y - X\theta)^T \frac{\partial(Y - X\theta)}{\partial \theta} \tag{8}$$

$$= 2(Y - X\theta)^T(-X) \tag{9}$$

Igualamos a expressão a 0, multiplicando por -1 e dividindo por 2. Também trocamos a ordem da multiplicação, o que requer também trocar as transposições. Daí, fazendo a distributiva:

$$0 = X^T(Y - X\theta) \tag{10}$$

$$= X^TY - X^TX\theta \tag{11}$$

A partir daqui basta isolar o θ:

$$X^TX\theta = X^TY \tag{12}$$

$$\theta = (X^TX)^{-1}X^TY \tag{13}$$

E obtemos uma fórmulazinha mágica para calcular nosso vetor de parâmetros ideal! "Fácil"assim.

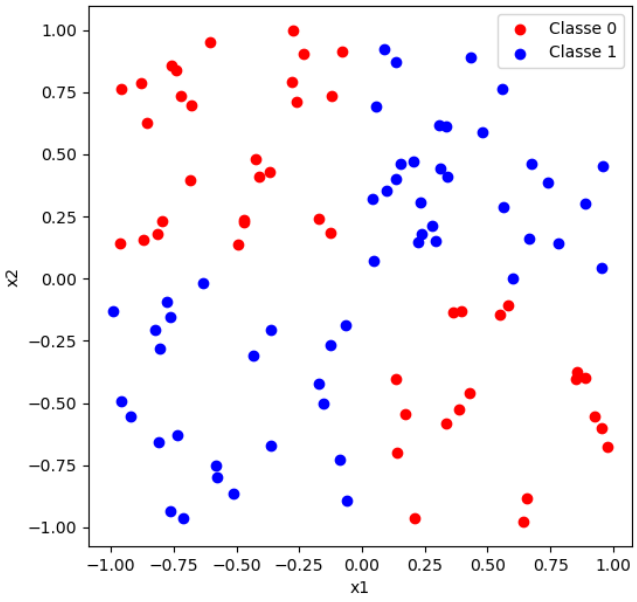
7 Feature Transformation

Feature transformation nada mais é do que uma transformação matemática que nós aplicamos no nosso vetor de features, podendo ser de vários tipos com diversos intuitos. Vamos ver um exemplo:

Assumindo que eu tenho o vetor de features $\vec{x}$ tal que:

$$\vec{x} = [x_1, x_2]$$

Vamos também que nós temos esse conjunto de dados:



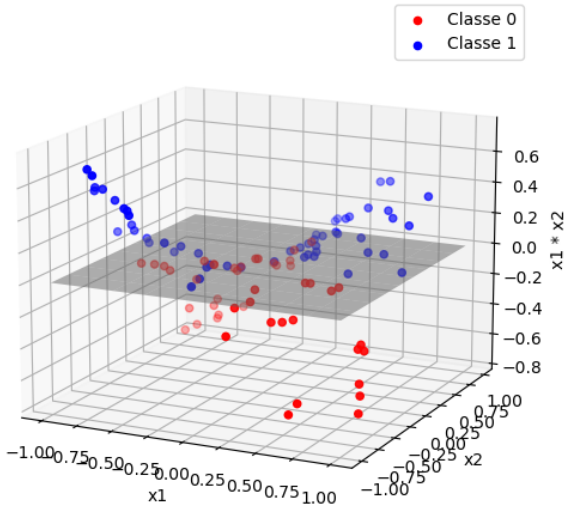
É fácil ver que tal dataset não é linearmente separável. É aí que entra a feature transformation, pois, com os dados que nós já temos a disposição podemos modificar e adicionar elementos ao nosso vetor de features $\vec{x}$.

Nesse caso, qual seria a feature tranformation que devemos aplicar? Para decidir isso precisamos analisar os nossos dados.

Se olharmos com cuidado, podemos ver que os pontos azuis são aqueles que estão ou no 1º quadrante ou no 3º quadrante (têm o mesmo sinal), e os pontos vermelhos são aqueles que estão ou no 2º quadrante ou no 4º quadrante (têm sinais diferentes). Logo, se adicionarmos o elemento $x_1 * x_2$ no nosso vetor de features $\vec{x}$ poderemos saber quais pontos tem sinais iguais e quais tem sinais diferentes, da forma que o nosso novo vetor de features $\vec{x'}$ será da forma:

$$\vec{x'} = [x_1, x_2, x_1 * x_2]$$

Se mapearmos o nossos pontos com essa nova informação teremos a seguinte representação:



Nesse novo espaço podemos ver que os pontos são facilmente linearmente separáveis.

Com esse exemplo podemos ver o poder das features transformations para solucionar problemas que antes, apenas com as ferramentas que vimos, não poderiam ser solucionados.

O ponto principal que devemos nos atentar é que, ao mudarmos o nosso vetor de features $\vec{x}$, estamos mudando o espaço em que os nossos dados estão sendo representados. Nesse exemplo nos mudamos de um espaço 2D para um espaço 3D, contudo, features transformations também podem nos manter em um espaço com o mesmo número de dimenções, como no caso em que apenas aplicamos uma função f em $\vec{x}$, da forma que:

$$\vec{x} = [x_1, x_2] \text{ então,} \tag{14}$$

$$\vec{x'} = [f(x_1), f(x_2)] \tag{15}$$

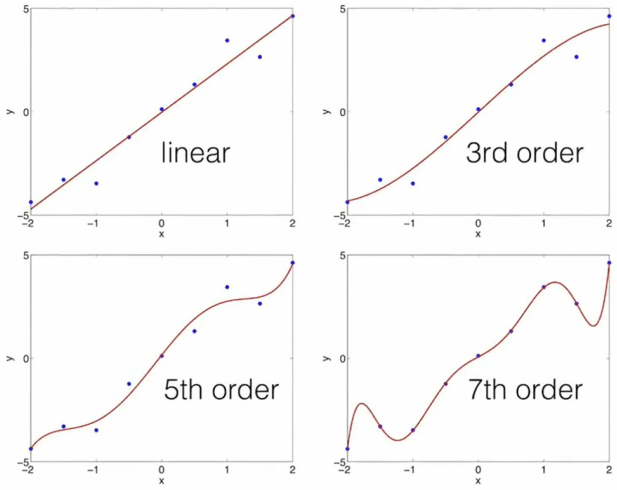
Por isso, devemos ter em mente que essas transformações que estamos realizando são apenas uma forma de movermos os nossos dados para um espaço no qual seja mais fácil trabalhar com eles. Dessa forma, devemos ter cuidado ao adicionarmos muita informação ao nosso vetor de features, pois podemos estar apenas adicionando dados inúteis que podem prejudicaram a performace do nosso modelo.

$$(x_1, x_2) \rightarrow (x_1, x_2, x_1 * x_2)$$

8 Transformação Polinomiais

No caso anterior para resolvermos o problema dos dados não serem linearmente separáveis nós aplicamos uma transformação de segunda ordem. Porém nem todos os problemas serão resolvidos somente com a segunda ordem. Em alguns deles será necessário fazer transformações de ordens ainda maiores.

A transformação polinomial é um tipo de feature transformation que consiste em mapear os atributos originais para um espaço de maior dimensão, incluindo potências e combinações dos atributos. Por exemplo, para um atributo único x, podemos definir transformações de diferentes ordens:



8.1 Primeira Ordem (Linear)

$$\phi(x) = [1, x]$$

Essa transformação é equivalente a não fazer nada nos dados, pois eles vão continuar na mesma dimensão que estavam anteriormente.

Em especial, esse 1 serve para mantermos o θ_0 dentro do próprio vetor θ. Caso você não irá trabalhar com o θ_0 dentro do vetor θ não existe a necessidade do 1.

8.2 Segunda Ordem (Quadrática)

$$\phi(x) = [1, x, x^2]$$

Para um vetor $x = [x_1, x_2]$:

$$\phi_2(x) = \left[1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\right]$$

8.3 Terceira Ordem (Cúbica)

$$\phi(x) = [1, x, x^2, x^3]$$

Para um vetor $x = [x_1, x_2]$:

$$\phi_3(x) = \left[1, x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3, x_2^3, x_1^2x_2, x_1x_2^2\right]$$

8.4 Quinta Ordem

$$\phi(x) = [1, x, x^2, x^3, x^4, x^5]$$

8.5 Sétima Ordem

$$\phi(x) = [1, x, x^2, x^3, x^4, x^5, x^6, x^7]$$

Essas transformações são muito poderosas, conseguindo expressar várias funções. Como se pode ser visto na última imagem, conforme as transformações de ordens superiores foram feitas a função começou a se encaixar perfeitamente nos dados, porém isso pode levar a overfitting, logo, a escolha da transformação que será utilizada é essencial.

Outra coisa que deve ser levado em consideração é que conforme as dimensões vão ficando muito alta será muito custoso fazer os cálculos desses vetores as vezes ficando computacionalmente inviável.

Na próxima aula entraremos no assuno de Funções Kernels que resolvem este problema da complexidade computacional.

9 Regressão Polinomial

A regressão polinomial será exatamente igual a regressão linear com a diferença que o nosso vetor de características será o vetor ϕ resultando da feature transformation e o vetor de parâmetros terá a mesma dimensão de ϕ:

1. Inicializar $\theta = 0$
2. Tomar randomicamente um $t = 1, ..., n$ (16)
3. $\theta = \theta + \eta\phi^t(y^t - \theta x^t)$