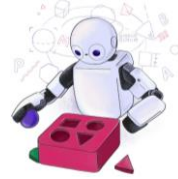


T319 - Introdução ao Aprendizado de Máquina: *Regressão Linear (Parte I)*



Inatel

Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

Motivação

- **Exemplo 1:** Estimar o preço de casas.
- **Exemplo 2:** Estimar as vendas de sorvete.



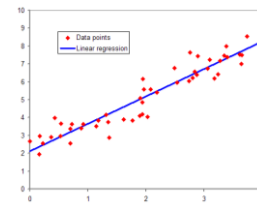
- Podemos encontrar uma relação matemática entre a área de uma casa e seu valor?
- Ou da temperatura e a quantidade de sorvetes vendidos?

Exemplos de aplicações de regressão linear:

- Modelos para predição da perda por caminho.
- Modelos para predição do preço de casas.
- Modelos para predição do salário de um empregado baseado em sua idade.

Regressão Linear

- Um dos mais, se não o mais, conhecido algoritmo de aprendizado de máquina.
- Vai nos dar vários *insights* (i.e., intuições) importantes para o entendimento de outros algoritmos mais complexos, como, classificadores e redes neurais.
- **Objetivo:** encontrar uma função, h , que mapeie, de forma ótima, os atributos de entrada x em uma variável de saída y : $y = h(x)$, de tal forma que $h(x)$ seja uma boa aproximação da **função verdadeira**, mas desconhecida, chamada de **função objetivo**, $f(x)$.
- Regressão também é conhecida como **aproximação de funções**.
- Como faríamos para encontrar uma função, $h(x)$, que **aproxime** $f(x)$ de forma ótima?



Temos x (atributos) e y (rótulos) e queremos encontrar $\hat{y} = h(x)$.

insight: percepção; intuição; compreensão.

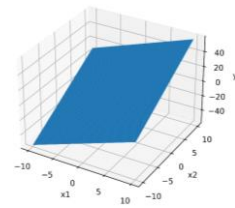
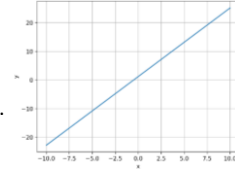
O termo "regressão" foi cunhado por Francis Galton no século 19 para descrever um fenômeno biológico. O fenômeno foi que as alturas dos descendentes de ancestrais altos tendem a **regredir** para a média (um fenômeno também é conhecido como **regressão à média**).

Os algoritmos de aprendizado de máquina são descritos como o aprendizado de uma função, h , que melhor mapeie as variáveis de entrada x em uma variável de saída y : $y = h(x)$

Regressão Linear

- **Qual forma deve ter a função $h(x)$?** Os modelos mais simples são:
 - Com apenas um atributo, x_1 , $h(x)$ é uma reta, $h(x) = a_0 + a_1 x_1$.
 - Com dois atributos, x_1 e x_2 , $h(x)$ é uma superfície 2D (ou seja, um plano), $h(x) = a_0 + a_1 x_1 + a_2 x_2$.
 - E assim por diante.
- **Modelo geral:** Equação de um Hiperplano

$$h(x) = a_0 + a_1 x_1 + \dots + a_K x_K = a_0 + \sum_{i=1}^K a_i x_i.$$
- Existem outros modelos, os quais veremos mais adiante.
- Na literatura, a função $h(x)$, é chamada de **função hipótese**, pois é uma das possíveis soluções encontradas no **espaço de hipóteses**, H .
- **Espaço de hipóteses:** é conjunto de todas as possíveis **funções hipótese**.
 - Hiperplano formado por todos possíveis valores dos parâmetros, $a_k, \forall k$.



Por quê Regressão LINEAR?

A regressão linear é chamada linear porque você modela sua variável de saída (y) como uma **combinação linear** de entradas e pesos (vamos chamá-las de x e a , respectivamente).

Linear significa "linear com relação aos parâmetros" e não às variáveis, i.e., x . Portanto os seguintes modelos também são lineares com relação aos parâmetros.

- $y = a_0 + a_1 \log(x_1) + a_2 \cos(x_2)$
- $y = a_0 + a_1 e^{x_1}$
- $y = a_0 + a_1 x_1^2$

Regressão também é conhecida como **aproximação de funções**.

Basicamente, regressão significa encontrar a melhor curva para seus dados numéricos, ou seja, uma aproximação funcional dos dados.

O **espaço de hipóteses** é o conjunto de todas as possíveis **funções hipótese**: $h(x) = a_0 + a_1 x_1 + a_2 x_2 + \dots + a_k x_k$. Esse conjunto de todas as possíveis funções forma um hiperplano.

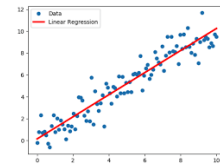
A escolha do espaço de hipóteses é importantíssima para a complexidade da tarefa de encontrar uma boa hipótese para a função desconhecida, f . Um espaço de hipóteses muito grande faz com que se leve muito tempo para se encontrar h .

Em geometria, um hiperplano é um subespaço cuja dimensão é um a menos que a de seu espaço ambiente. Se um espaço é tridimensional, então seus hiperplanos são os planos bidimensionais, enquanto se o espaço é bidimensional, seus hiperplanos são as linhas unidimensionais.

Regressão Linear

- **Objetivo:** Encontrar os **parâmetros**, também chamados de **pesos**, a_0, a_1, \dots, a_k de tal forma que $h(\mathbf{x})$ seja uma ótima aproximação de $f(\mathbf{x})$.
- **Aprendizado supervisionado:** atributos/exemplos (i.e., \mathbf{x}) mais rótulos/objetivos (i.e., y).
- A **regressão** é chamada de **linear** porque a variável de saída, y , é modelada como uma **combinação linear** dos atributos, \mathbf{x} .
 - **OBS.:** **Linear** significa “*linear com relação aos pesos*” e não com relação aos atributos, i.e., \mathbf{x} . Desta forma, os seguintes modelos também são lineares com relação aos pesos:
 - $y = a_0 + a_1 \log x_1 + a_2 \cos x_2$
 - $y = a_0 + a_1 e^{x_1}$
 - $y = a_0 + a_1 x_1^2$

Definição do Problema



O problema de **regressão** pode ser enunciado da seguinte forma:

- **Dados disponíveis:**

- Conjunto de N observações (conjunto de pares de treinamento) : $\{x(i), y(i)\}$, $i = 0, \dots, N - 1$, onde
 - $x(i) \in \mathbb{R}^K$: i -ésimo vetor de entrada com dimensão K , ou sejam K atributos (ou features).
 - $y(i) \in \mathbb{R}$: i -ésimo valor esperado de saída referente ao vetor de entrada $x(i)$.

- **Modelo:**

$$h(x(i)) = a_0 + a_1 x_1(i) + \dots + a_K x_K(i) = \mathbf{a}^T \Phi(i),$$

onde $\mathbf{a} = [a_0, \dots, a_K]^T$ e $\Phi(i) = [1, x_1(i), \dots, x_K(i)]^T$.

- \mathbf{a} é o vetor $(K + 1 \times 1)$ contendo os parâmetros/pesos que definem a **função hipótese**, ou seja, o mapeamento $h: x(i) \rightarrow \hat{y}(i)$ e $\Phi(i)$ é um vetor $(K + 1 \times 1)$ contendo os i -ésimos valores dos atributos.
- a_0 é o **coeficiente linear**, ou seja, é o valor de $h(x)$ para o ponto em que o hiperplano intercepta o eixo y , a_0 é conhecido também como **intercept**.
- Como a_0 não tem um atributo relacionado, para facilitar o modelamento matemático, supomos um atributo constante sempre unitário, $x_0 = 1$.
- **Objetivo do modelo:** encontrar o vetor de pesos \mathbf{a} que minimize o **erro**, dado por uma **função de erro**, $J_e(\mathbf{a})$, entre a aproximação $\hat{y}(i)$ e o valor desejado $y(i)$ para todo i .

$$\min_{\mathbf{a}} J_e(\mathbf{a})$$

Ou seja, **o treinamento do modelo envolve a minimização de uma função de erro.**

a_0 é o coeficiente linear, ou seja, é o valor de $h(x)$ para o ponto em que a reta intercepta o eixo y .

Função de Erro

- **Função de erro:** existem várias possibilidades para se definir a **função de erro** a ser minimizada, porém, geralmente, utiliza-se a medida do **erro quadrático médio**

$$J_e(\mathbf{a}) = \frac{1}{N} \sum_{i=0}^{N-1} (y(i) - \hat{y}(i))^2 = \frac{1}{N} \sum_{i=0}^{N-1} (y(i) - h(\mathbf{x}(i), \mathbf{a}))^2,$$

que nada mais é do que a média do somatório dos erros ao quadrado.

- Nós veremos mais adiante a razão pela qual o **erro quadrático médio** é utilizado.
- A **função de erro** pode ser reescrita em forma matricial como

$$J_e(\mathbf{a}) = \frac{1}{N} \|\mathbf{y} - \Phi \mathbf{a}\|^2,$$

onde $\mathbf{y} = [y(0), \dots, y(N-1)]^T$ é um vetor $(N \times 1)$, $\Phi = [\Phi(0), \dots, \Phi(N-1)]^T$ é uma matriz $(N \times K+1)$ e N é o número de amostras, exemplos ou observações.

- Então, para encontrar o vetor de parâmetros \mathbf{a} devemos minimizar

$$\min_{\mathbf{a} \in \mathbb{R}} \|\mathbf{y} - \Phi \mathbf{a}\|^2.$$

O uso do **erro quadrático médio** remonta a Gauss, que demonstrou que quando os valores de $y(i)$ tem ruído com distribuição normal, então os valores mais prováveis para o vetor de parâmetros/pesos \mathbf{a} são obtidos através da minimização da soma dos quadrados dos erros.

Minimizando a Função de Erro

Como encontramos o mínimo da função de erro em relação aos pesos?

- Da disciplina de cálculo, sabemos que derivando $\|y - \Phi a\|^2$ com relação a a e igualando a 0 nós encontramos o ponto onde a inclinação da função de erro é nula:

$$\frac{\partial \|y - \Phi a\|^2}{\partial a} = 2a^T \Phi^T \Phi - 2y^T \Phi = 0.$$

- Portanto, voltando à equação da derivada primeira igual a 0, temos

$$a^T \Phi^T \Phi = y^T \Phi.$$

- Após aplicarmos o transposto a ambos os lados e isolando a temos

$$a = (\Phi^T \Phi)^{-1} \Phi^T y.$$

- Essa equação é conhecida como a **equação normal** e nos dá a **solução ótima** em relação a minimização o erro quadrático médio para esse sistema de equações.

O mínimo é encontrado através do método dos **mínimos quadrados**.

Uma derivada basicamente encontra a inclinação de uma função. A inclinação é nula nos mínimos e máximos de uma função. Se a inclinação é nula e a segunda derivada é menor do que zero, então este é um máximo local, por outro lado, se a derivada segunda for positiva, então, este é um mínimo local.

Posto de uma matriz: corresponde ao número de linhas ou colunas linearmente independentes da matriz.

Se A é uma matriz quadrada (ou seja, $m = n$), então A é invertível se e somente se A tiver posto igual a n (ou seja, A tiver posto completo).

O método dos mínimos quadrados só funciona para sistemas sobredeterminados, ou seja, conjuntos de equações nas quais há mais equações (pares x e y) que incógnitas (valores do vetor a).

Minimizando a Função de Erro

Observações:

1. O método encontra uma ***solução única*** se e somente se a matriz quadrada $\Phi^T \Phi$ for ***invertível***.
2. O método só funciona para sistemas ***sobredeterminados***, ou seja, mais equações (i.e., pares de exemplos x e y) do que incógnitas (i.e., pesos), $N \geq K + 1$.
3. Para sistemas ***subdeterminados***, ou seja, que têm menos equações do que incógnitas, a matriz $\Phi^T \Phi$ é ***singular***. Neste caso, não existe solução ou ela não é única.

Sistemas subdeterminados (número de exemplos menor do que o número de parâmetros) não tem solução ou não tem solução única.

Regressão Linear em Python

```
# Import all the necessary libraries.
import numpy as np

# Generate input/output (features/labels) values.
N = 100; # Number of observations.

x = 2 * np.random.rand(N, 1)
y = 4 + 3 * x
y_noisy = y + np.random.randn(N, 1)

# Solve by applying the least-Squares method.
# We use the inv() function from NumPy's Linear Algebra module (np.linalg) to
compute the inverse of a matrix.
# We use dot() method for matrix multiplication.
X_b = np.c_[np.ones((N, 1)), x] # add a0 = 1 to each instance
a_optimum = np.linalg.inv(X_b.T.dot(X_b)).dot(X_b.T).dot(y_noisy)

# Print best solution.
print('a0: %1.4f' % (a_optimum[0][0]))
print('a1: %1.4f' % (a_optimum[1][0]))


a0: 4.0763
a1: 2.9413
```

The equivalent solution using the Scikit-Learn library is given below
Import the linear regression module from the library.
from sklearn.linear_model import LinearRegression

```
lin_reg = LinearRegression() # instantiate it.
lin_reg.fit(x, y_noisy)

print('a0: %1.4f' % (lin_reg.intercept_[0])) # Value that crosses the y-axis when all features
are equal to 0.
print('a1: %1.4f' % (lin_reg.coef_[0][0])) # parameters associated with the features.

a0: 4.0763
a1: 2.9413
```

 <https://scikit-learn.org>

Exemplo: normal_equation_example1.ipynb

- Percebam que com apenas 100 exemplos, os valores obtidos são bem próximos dos exatos, porém, o ruído torna impossível recuperar os parâmetros exatos da função original.
- Se aumentarmos o número de exemplos conseguimos melhorar a estimação, porém, o ruído limitará essa melhoria.
- O argumento x passado para o método `fit` da classe **LinearRegression** é uma matriz com $N \times K$. Porém, lembrem-se que se existe um peso a_0 , e portanto x deveria ter dimensão $N \times K+1$, entretanto, por padrão, a classe **LinearRegression** já faz isso pra vocês automaticamente. Caso sua função hipótese não considere o peso a_0 , então, durante a instanciação da classe vocês devem configurar o parâmetro `fit_intercept=False`.

Exemplo:

https://mybinder.org/v2/gh/zz4fap/t319_aprendizado_de_maquina/main?filepath=notebooks%2Fregression%2Fnormal_equation_example1.ipynb

`numpy.c_` : concatena vetores.

Perceba que com apenas 100 observações (ou exemplos), os valores obtidos são bem próximos dos exatos, porém, o ruído torna impossível recuperar os parâmetros exatos da função original.

Se aumentarmos o número de observações conseguimos melhorar a estimação.

O argumento x passado para o método `fit` da classe **LinearRegression** é uma matriz com $N \times K$, ou seja, uma matriz com N exemplos como linhas e K atributos como colunas. Porém, lembre-se que se existe um peso a_0 , e portanto x deveria ter dimensão $N \times K+1$, entretanto, por padrão a classe **LinearRegression** já faz isso pra você automaticamente. Caso sua função hipótese não considere o peso a_0 , então, durante a instanciação da classe você deve configurar o parâmetro **fit_intercept=False**, pois por default o método `fit` tenta encontrar o valor de a_0 .

Superfície de Erro

- E se plotarmos a função de erro, $J_e(\mathbf{a})$? Que forma vocês acham que ela teria?
- $J_e(\mathbf{a})$ faz o mapeamento entre cada possível valor dos parâmetros do modelo e o erro correspondente:
 - $J_e(\mathbf{a}): \mathbb{R}^{K+1} \rightarrow \mathbb{R}$. Esse mapeamento define a **superfície de erro**.
- $J_e(\mathbf{a})$ assume uma forma **quadrática** com respeito ao **vetor de pesos, \mathbf{a}** .

$$J_e(\mathbf{a}) = \|\mathbf{y} - \Phi \mathbf{a}\|^2 = \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \Phi \mathbf{a}^T - \mathbf{a}^T \Phi^T \mathbf{y} + \mathbf{a}^T \Phi^T \Phi \mathbf{a}.$$
- Consequentemente, a superfície é **convexa** (ou seja, tem forma de tigela), e portanto possui um único **mínimo global**, que pode ser encontrado, por exemplo, pela **equação normal**.
- Isso é provado mostrando-se que $\frac{\partial^2 \|J_e(\mathbf{a})\|^2}{\partial^2 \mathbf{a}} = 2\Phi^T \Phi$ é uma **matriz positiva semi-definida**, e portanto, $J_e(\mathbf{a})$ sempre será **convexa** com relação ao vetor de pesos, \mathbf{a} .

No problema de regressão linear, a superfície de erro para o critério de quadrados mínimos é dada pela expressão: $J_e(\mathbf{a}) = \mathbf{y}^T \mathbf{y} - \mathbf{y}^T \Phi \mathbf{a}^T - \mathbf{a}^T \Phi^T \mathbf{y} + \mathbf{a}^T \Phi^T \Phi \mathbf{a}$.

O mínimo global pode ser encontrado pelo método dos mínimos quadrados (ou também conhecido como equação normal).

Observe que a função de erro é convexa, com um único mínimo global.

Convexo: curvo ou curvado para fora como o exterior de uma tigela ou esfera ou círculo.

Uma função de valor real definida em um intervalo n-dimensional é denominada **convexa** se um segmento de linha entre dois pontos no gráfico da função estiver acima ou no gráfico.

Prova de que formas quadráticas são convexas:

- <https://math.stackexchange.com/questions/526657/show-convexity-of-the-quadratic-function>
- <https://math.stackexchange.com/questions/483339/proof-of-convexity-of-linear-least-squares>

A figura da direita mostra um gráfico com linhas de contorno. Uma linha de contorno ou isoline de uma função de duas variáveis é uma curva ao longo da qual a função tem um valor constante. Ou seja, no nosso caso, cada uma das linhas indica curvas que tem o mesmo erro.

Superfície de Erro: Exemplo #1

- A figura ao lado mostra a **superfície de erro** para a seguinte **função observável**:

$$y_{\text{noisy}}(n) = y(n) + w(n),$$

onde $w(n) \sim N(0,1)$ e $y(n)$ é a **função objetivo**.

- Neste exemplo, a **função objetivo** (ou **modelo gerador**) é dada por:

$$y(n) = x_1(n),$$

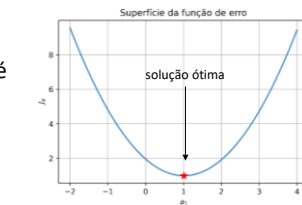
onde $x_1(n) \sim N(0,1)$.

- A **função hipótese**, $h(x)$, é dada por

$$h(x) = \hat{y} = a_1 x_1(n).$$

- O erro **erro quadrático médio** é calculado como

$$J_e(a_1) = \frac{1}{N} \sum_{n=0}^{N-1} (y(n) - \hat{y}(n))^2 = \frac{1}{N} \sum_{n=0}^{N-1} (y(n) - a_1 x_1(n))^2.$$



O erro é calculado variando-se a_1 na equação do EQM.

Exemplo:

https://mybinder.org/v2/gh/zz4fap/t319_aprendizado_de_maquina/main?filepath=notebooks%2Fregression%2Ferror_surface_example1.ipynb

Superfície de Erro: Exemplo #2

[Exemplo: error_surface_example2.ipynb](#)

- A figura ao lado mostra a **superfície de erro** para a seguinte **função observável**:

$$y_{\text{noisy}}(n) = y(n) + w(n),$$

onde $w(n) \sim N(0,1)$ e $y(n)$ é a **função objetivo**.

- Neste exemplo, a **função objetivo** (ou **modelo gerador**) é dada por:

$$y(n) = x_1(n) + x_2(n),$$

onde $x_1(n)$ e $x_2(n) \sim N(0,1)$

- A **função hipótese**, $h(x)$, é dada por

$$h(x) = \hat{y} = a_1 x_1(n) + a_2 x_2(n).$$

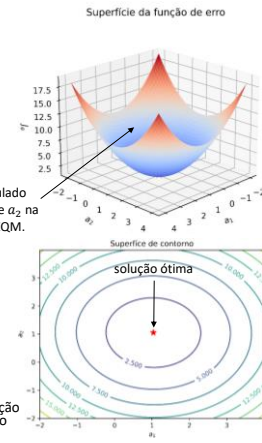
- O erro **erro quadrático médio** é calculado como

$$J_e(a_1, a_2) = \frac{1}{N} \sum_{n=0}^{N-1} (y_{\text{noisy}}(n) - \hat{y}(n))^2$$

$$= \frac{1}{N} \sum_{n=0}^{N-1} (y_{\text{noisy}}(n) - (a_1 x_1(n) + a_2 x_2(n)))^2$$

- A segunda figura mostra a **superfície de contorno**.

- Uma linha de contorno de uma função de duas variáveis é uma curva ao longo da qual a função tem um valor constante. Ou seja, no nosso caso, cada uma das linhas indica curvas que têm o mesmo erro

**Exemplo:**

https://mybinder.org/v2/gh/zz4fap/t319_aprendizado_de_maquina/main?filepath=notebooks%2Fregression%2Ferror_surface_example2.ipynb

Desvantagens da forma fechada (Eq. Normal)

- **Alta complexidade computacional:** a solução da **equação normal** envolve o cálculo da inversa de $\Phi^T \Phi$, o qual tem complexidade computacional que varia de $O(n^{2.4})$ a $O(n^3)$.
 - Ex.: Se o número de **atributos** dobrar, o tempo para cálculo aumenta de $2^{2.4} = 5.3$ a $2^3 = 8$ vezes.
- Dependendo do número de **exemplos**, N , e **atributos**, x , a matriz Φ pode consumir muita memória.
- **Portanto, essa abordagem não é escalonável!**
- Adicionalmente, para irmos além dos modelos lineares (i.e., regressão não-linear) precisamos lidar com o fato de que não existem formas fechadas como a **equação normal**.
- **Solução:** abordagens iterativas
 - Métodos iterativos de busca que façam a atualização dos parâmetros, \mathbf{a} , à medida que os dados forem sendo apresentados ao modelo.
 - Por exemplo, o algoritmo do **gradiente descendente**, o qual veremos a seguir.

- A solução da **equação normal** envolve o cálculo da inversa de $\Phi^T \Phi$, o qual tem complexidade computacional variando entre $O(n^{2.4})$ e $O(n^3)$, dependendo da implementação utilizada.
- Um algoritmo de complexidade $O(n^2)$ tem crescimento quadrático e $O(n^3)$ cúbico.
- $\Phi^T \Phi$ é uma matriz $(K+1) \times (K+1)$
- Pesquisa em tomografia sísmica envolve um número muito grande de features, da ordem de 10000 features.

Tarefas

- **Quiz:** “T319 - Quiz - Regressão: Parte I (1S2021)” que se encontra no MS Teams.
- **Exercício Prático:** [Laboratório #2](#).
 - Pode ser baixado do MS Teams ou do GitHub.
 - Pode ser respondido através do link acima (na nuvem) ou localmente.
 - [Instruções para resolução e entrega dos laboratórios](#).

Laboratório #2:

https://mybinder.org/v2/gh/zz4fap/t319_aprendizado_de_maquina/main?filepath=labs%2FLaboratorio2.ipynb

Obrigado!

