



Universidade Estadual de Campinas  
Instituto de Matemática, Estatística e Computação Científica  
MS960 - Aprendizado de Máquinas: Aspectos Teóricos e Práticos - Prof.: João Florindo  
Alunos:  
Júlia Machado Moretto, RA: 176953  
Pedro Gabriel Martins Ono, RA: 158336

# Projeto 1

Campinas  
2020

# Sumário

<b>1</b>	<b>Introdução</b>	<b>III</b>
<b>2</b>	<b>Regressão Linear</b>	<b>III</b>
2.1	Regressão linear ajuste a função polinomial . . . . .	III
2.2	Regressão linear ajuste a função exponencial . . . . .	IV
2.3	Diferentes valores para a taxa de aprendizado . . . . .	IV
2.4	Equações normais . . . . .	IV
<b>3</b>	<b>Regressão Logística</b>	<b>V</b>
3.1	Regressão logística multi-classes . . . . .	V
3.2	Regressão logística multi-classes regularizada . . . . .	VI
<b>4</b>	<b>Conclusão</b>	<b>VI</b>
<b>5</b>	<b>Referências</b>	<b>VII</b>
<b>6</b>	<b>Apêndice</b>	<b>VII</b>

# 1 Introdução

O objetivo desse projeto consistiu em implementar regressão linear multivariável (com algumas variações descritas em suas especificidades nas próximas seções) e aplicar a dados do número de casos de covid 19 no Brasil e regressão logística multivariável e regularizada a imagens do banco MNIST para classificar os dígitos manuscritos.

Realizamos esse projeto em Python3 em um Jupyter Notebook que pode ser acessado por link disponível no fim desse documento no apêndice.

## 2 Regressão Linear

Nessa seção utilizamos os dados fornecido pela disciplina no arquivo casesBrazil.csv com dados dos números oficiais de casos de do COVID-19 no Brasil de 100 dias a partir de 25 de fevereiro de 2020.

Utilizamos para essa parte as seguintes bibliotecas: pandas, numpy, matplotlib.pyplot.

### 2.1 Regressão linear ajuste a função polinomial

Implementamos o algoritmo de regressão linear chamado de Gradiente Descendente e o aplicamos para o ajuste de polinomiais de ordem 3,5,10.

Definimos os dados de entrada (dias) como X, adicionado de uma coluna de 1's e os dados de saída (números de casos) como Y.

Inicialmente normalizamos e padronizamos os nossos dados de entrada seguindo duas fórmulas:

Normalização(scaling):

$$x = x - \min(dias) / (\max(dias) - \min(dias)) \quad (1)$$

Padronização(z-score):

$$x = x - \text{média}(dias) / \text{desvioPadrão}(dias) \quad (2)$$

Para os polinômios só elevamos a ordem relativa os dados já normalizados/padronizados.

Testamos nosso modelo com ambas possibilidades. Além disso variamos nossa taxa de aprendizado(  $\alpha$ ) e números de iterações para observar as variações no resultado.

Veja tabela 1.

Tipo	Grau polinômio	# Iterações	$\alpha$	Custo
Padr	5	5000	0.03	2.80E+07
Padr	5	5000	0.01	4.10E+07
Norm	10	5000	1	5.60E+07
Norm	10	5000	0.3	7.60E+07
Norm	10	1000	1	8.40E+07

Tabela 1: Melhores valores de custo para regressão linear ajuste a função polinomial

## 2.2 Regressão linear ajuste a função exponencial

Nessa parte tentamos aproximar( com regressão linear e gradiente descendente) os mesmos dados do item 1 mas a uma curva exponencial:

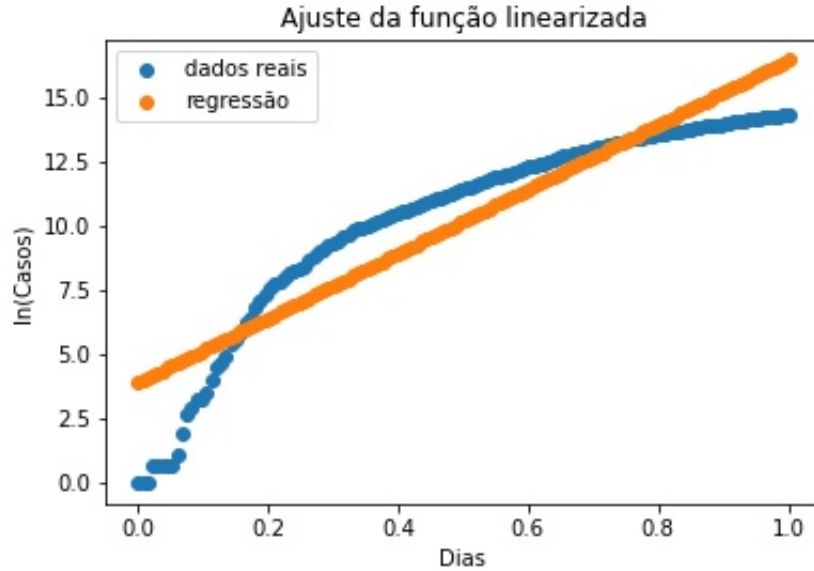
$$\theta_0 e^{\theta_1 x} = y \quad (3)$$

. Que linearizamos para aplicar gradiente descendente:

$$\ln(\theta_0) + \theta_1 x = \ln(y) \quad (4)$$

. Onde  $\ln(\theta_0)$  virou o primeiro coeficiente,  $x$  permaneceu sendo a entrada e a saída passou a ser  $\ln(y)$ . Também padronizamos e normalizamos os dados de entrada conforme o item anterior. Veja figuras 1 e 2.

Figura 1: Gráfico da Exponencial Linearizada



## 2.3 Diferentes valores para a taxa de aprendizado

Aqui variamos nossa taxa de aprendizado(  $\alpha = 10, 3, 1, 0.3, 0.1, 0.03, 0.01, 0.003, 0.001$ ) e observamos o gráfico de custo  $J$  Veja tabela 2 e figura 3

## 2.4 Equações normais

Aqui resolvemos o problema de aproximação da exponencial por equações normais, que nos resulta no melhor valor possível por ser resolvida analiticamente.

Para isso apenas resolvemos a seguinte equação matricial:

$$\theta = (X^t X)^{-1} X^t y \quad (5)$$

Figura 2: Regressão para curva exponencial



Tipo	# Iterações	$\alpha$	Custo
Norm	5000	0.3	1.202645
Padr	5000	0.03	1.202645
Padr	10000	0.1	1.202645
Padr	5000	0.1	1.202645
Padr	1000	0.1	1.202645

Tabela 2: Melhores valores de custo para regressão linear ajuste a função exponencial

. Veja figura 4.

### 3 Regressão Logística

Usando 5000 dados de imagens do MNIST escritas como um vetor de tamanho 400 (arquivo dado pela disciplina imageMNIST.csv) e um arquivo com os dígitos correspondentes(labelMNIST.csv) utilizamos o algoritmo de regressão logística para classificação desses números.

Além das bibliotecas da seção de regressão linear utilizamos: cv2 e random.

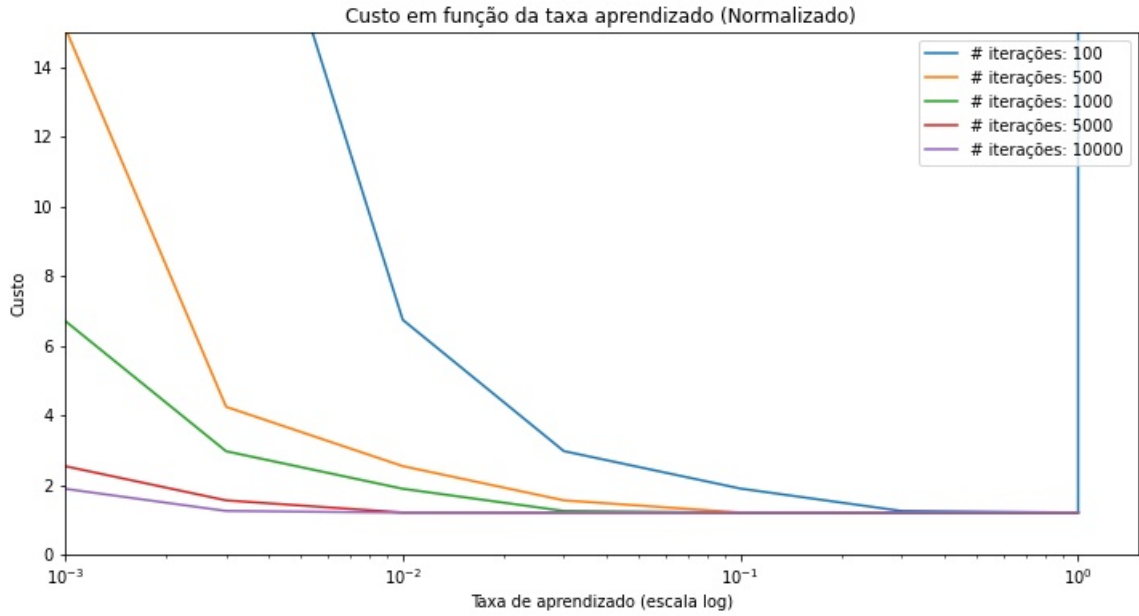
Utilizamos os seguintes parâmetros: taxa de aprendizado  $\alpha = 0.04$  e número de iterações = 3000

#### 3.1 Regressão logística multi-classes

Implementamos a regressão logística multi-classes utilizando a função sigmóide e adaptando o algoritmo do gradiente descendente utilizado no caso linear.

Nessa regressão obtivemos uma porcentagem de acertos 90.2 %

Figura 3: Custo por taxa de aprendizado



Alguns dos dígitos identificados errado estão nesse relatório, para ver todos acesse o repositório do github informado no apêndice. Veja figura 5.

### 3.2 Regressão logística multi-classes regularizada

Nessa parte implementamos a regressão logística multi-classes com regularização. A regularização funciona como uma função de penalidade. Segue abaixo nosso resultado com nosso fator de regularização  $\lambda = 0.5$  Nessa regressão obtivemos uma porcentagem de acertos de 89.7 % Veja figura 6.

## 4 Conclusão

Para o primeiro problema percebemos que a regressão polinomial foi mais adequada aos dados do que a exponencial.

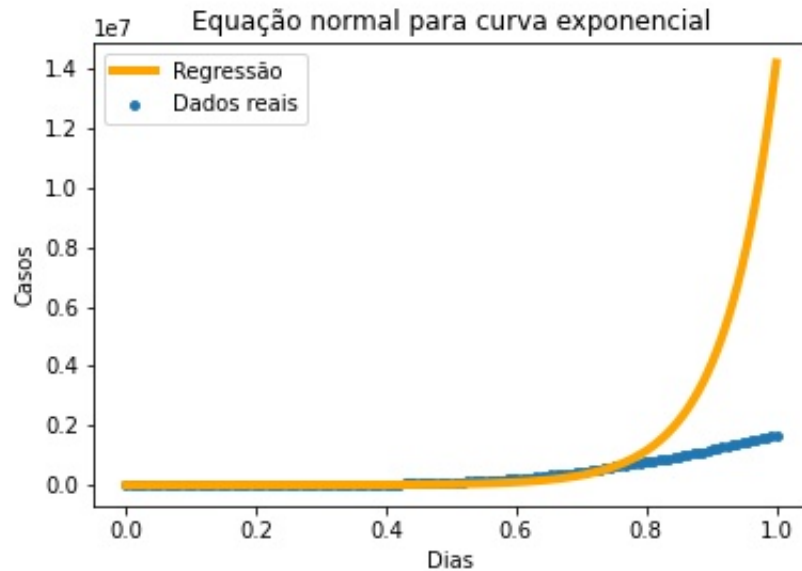
Podemos perceber também que mesmo com um baixo número de iterações obtivemos um resultado próximo ao analítico (equação normal).

Além disso notamos que para um  $\alpha > 1$  a solução divergiu, logo valores menores são mais adequados a esse caso.

Para o segundo problema os resultados com e sem regularização foram semelhante. Logo foi inconclusivo, neste caso, qual seria o melhor algoritmo.

Obtivemos bons resultados mesmo com a simplicidade dos métodos.

Figura 4: Equação normal para curva exponencial



## 5 Referências

- <https://www.ime.unicamp.br/~friedlan/livro.pdf>
- <https://pandas.pydata.org/docs>
- <https://matplotlib.org/3.3.2/contents.html>
- <http://yann.lecun.com/exdb/mnist/>

Acessos em 10/2020.

## 6 Apêndice

Link para o código do projeto: <https://github.com/Pedro0no/ms960>

Sim, o cachorro Ziko codou a maior parte desse projeto. Veja figura 7.

Figura 5: Erros - não regularizado

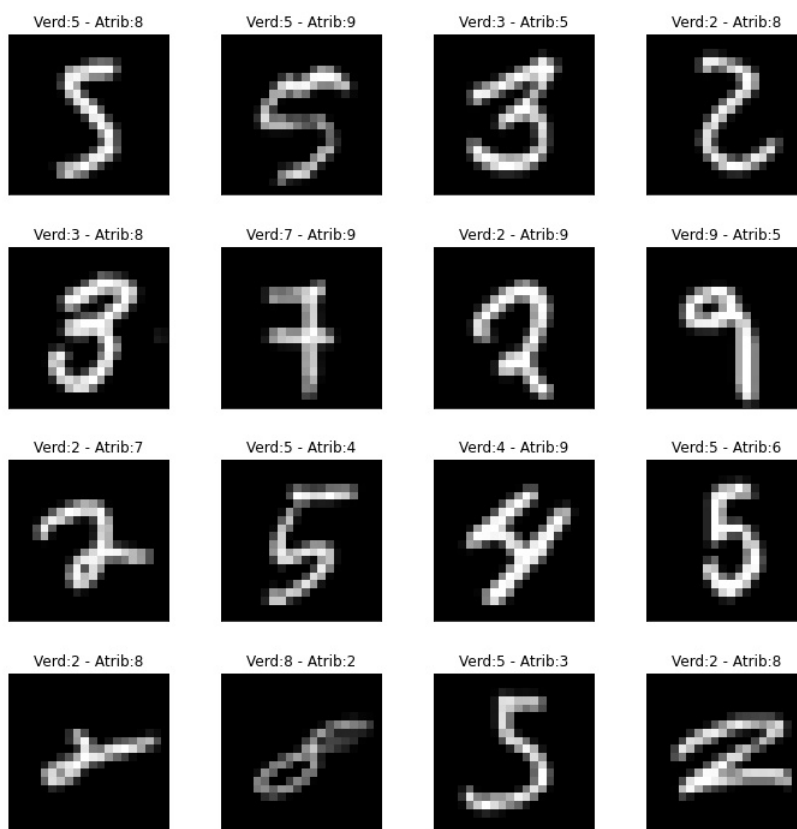




Figura 6: Erros - regularizado

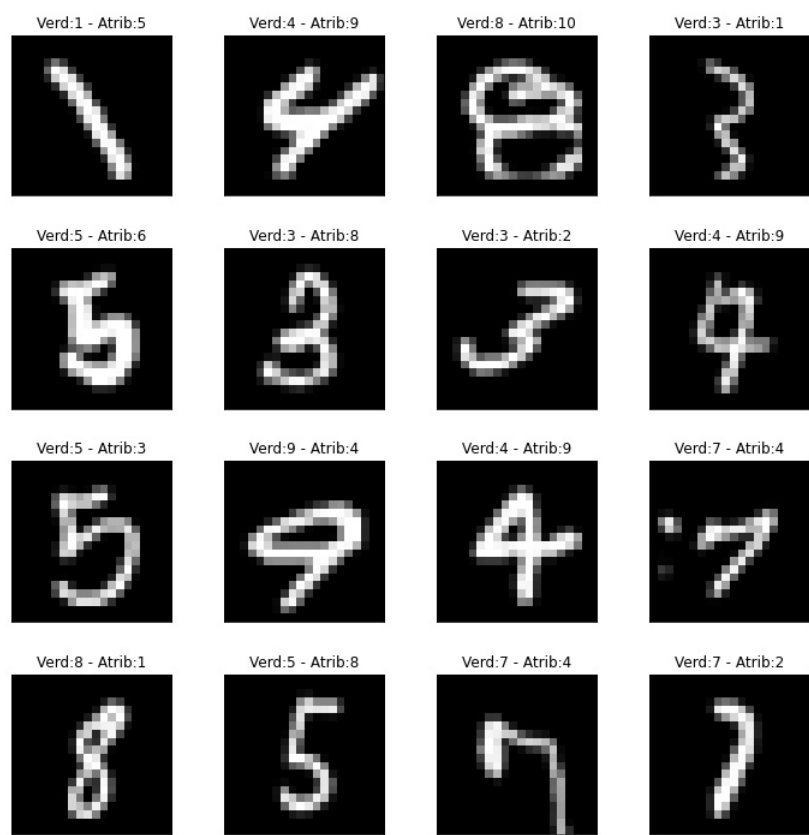


Figura 7: Foto de equipe

