

Introdução

O presente documento tem como o objetivo apresentar os resultados que conseguimos durante a implementação e os estudos realizados nos tópicos regressão linear e regressão logística que fazem parte no conteúdo da disciplina "Aprendizado de Máquinas". A primeira parte do documento faz referencia à regressão linear e todo o que tem a ver com os resultados solicitados e produzidos ao aplicar o código de programação e a segunda parte é todo o relacionado à regressão logística.

O projeto esta divido em duas partes, na Parte I nós trabalhamos com os dados do número oficial de casos do Covid-19 no Brasil, a partir de 25 de fevereiro de 2020, e se estendendo pelos 100 dias seguintes, com estes dados aplicamos a regressão linear. Já na parte II implementamos a regressão logística a imagens para o reconhecimento de dígitos manuscritos.

Durante a exposição dos resultados, iremos mencionando conteúdos teóricos das duas regressões, para dizer os fatos mais relevantes ao fazer uso delas.

Parte I

Regressão Linear

A regressão linear é uma ferramenta estatística comum para modelar a relação entre algumas variáveis explicativas e resultados com valor real. O modelo mais simples para regressão linear é o que envolve uma combinação linear das variáveis de entrada

$$h(\mathbf{x}, \theta) = \theta_0 + \theta_1 x_1 + \dots + \theta_m x_m$$

Onde $\mathbf{x} = (x_1, \dots, x_m)^T$. A maior propriedade deste modelo é que ele é uma função linear dos parâmetros $\theta_1, \dots, \theta_m$ ademais é uma função linear das variáveis de entrada x_i impondo limitações significantes ao modelo.

Além disso, é preciso definir uma função de perda para indicar se $h(\mathbf{x})$ prediz \mathbf{y} corretamente ou não, ou em outras palavras, a função de perda é para definir o quanto será penalizada a discrepância entre $h(\mathbf{x})$ e \mathbf{y} .

Uma maneira comum é usar a função de perda ao quadrado.

$$L(h(\mathbf{x}), \mathbf{y}) = (h(\mathbf{x}) - \mathbf{y})^2$$

Por outro lado, algumas tarefas de aprendizagem exigem preditores não lineares, como polinomiais, por exemplo,

$$p(\mathbf{x}) = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_n x^n$$

Para solucionar aquela regressão polinomial como um problema de regressão linear, se define o mapeamento $\psi(\mathbf{x}) = (1, x, x^2, \dots, x^n)$ para ter o polinômio como um produto interno entre o vetor de coeficientes θ y o mapeamento $\psi(\mathbf{x})$ e poder encontrar o vetor ótimo de coeficientes θ .

Agora, no problema do projeto temos como objetivo encontrar uma função que aproxime o número de casos de Covid-19 no Brasil. Para isso treinamos os dados oficiais a partir de 25 de fevereiro e por 134 dias seguintes. Na tabela 1 se observa que a variável de entrada ou atributos são os dias t , começando em 1 até 134. Na

figura 1 se tem o número de casos por dia tendo aparentemente um crescimento exponencial.

Tabela 1: Número de casos de Covid-19 no Brasil

Dia	Número de casos
1	1
20	225
40	10381
60	54209
80	205173
100	589048
120	1154255

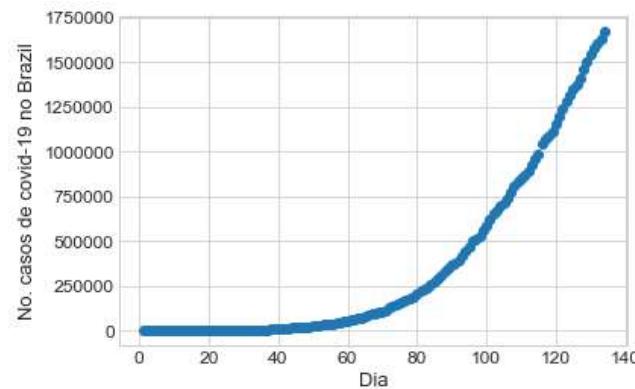


Figura 1: Número de casos Covid-19 Brasil.

A primeira hipóteses foi supor que a função é um polinômio de grau n . Para isso foi necessário usar o método de regressão multilinear. Basados na teoria, se tem que a função de hipóteses de uma multilinear de n atributos está dada por

$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n.$$

Portanto, para os m exemplos de treinamento a vetorização do problema de regressão multilinear é da forma

$$h_{\theta}(\mathbf{X}) = \mathbf{X}\theta = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \dots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(1)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \dots & x_1^{(m)} \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}. \quad (1)$$

Para o projeto em particular, cada atributo foi definimos o $x_j = t^j$, com $j = 0, 1, \dots, n$, para obter a matriz \mathbf{X} como em (1) de dimensão $m \times (n + 1)$, logo

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 4 & 8 & \dots & 2^n \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ 1 & m & m^2 & m^3 & \dots & m^n \end{bmatrix}. \quad (2)$$

Analiticamente obtemos a matriz \mathbf{X} definida anteriormente, mas na pratica foi necessário fazer reescalamento dos dados, tomando os atributos de entrada (dias de 1 até m)para evitar problemas de overflow. Para isso, precisamos dividir cada elemento pelo maior t , obtendo um vetor de números na forma i/m , $i = 1, 2, \dots, m$. Logo, a matriz usada no algoritmo de regressão multilinear é da forma:

$$\mathbf{X} = \begin{bmatrix} \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \frac{1}{m} & \dots & \frac{1}{m} \\ \frac{1}{m} & \frac{2}{m} & \frac{4}{m} & \frac{8}{m} & \dots & \frac{2^n}{m} \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ \frac{1}{m} & 1 & 1 & 1 & \dots & 1 \end{bmatrix}.$$

Já que \mathbf{y} são os valores de saída, então a função custo para o método de regressão multilinear é definido como o erro quadrático normalizado e se escreve na forma vetorizada como

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(\mathbf{X}) - \mathbf{y})^2 \quad (3)$$

Para obter o melhor θ tal que minimize a função custo, isso é, para que o erro seja minimizado, e ademais econômico computacionalmente; usamos o método do gradiente descendente cujo algoritmo é,

Algorithm 1: Gradiente descendente

Input: X, k, m, y, α

Output: Vetor θ que aproxima a função de hipóteses aos targets y

for $i \leq k$ **do**

| $\theta \leftarrow \theta - \frac{\alpha}{m} X^T (h_\theta(X) - y)$

end

Tendo presente o anteriormente mencionado os resultados obtidos ao aplicar o código no Python, definimos os parâmetros $\alpha = 10, 3, 1, 0.3, 0.1, 0.03, 0.01, 0.003, 0.001$, número de iterações $n_i = 1500$ e polinômios de grau $n = 3, 5, 10$.

Para valores $\alpha = 3$ e $\alpha = 10$ o modelo diverge para todos os graus n do polinômio. Observamos nas figuras 2,3 e 4 que o custo é melhor para todos os casos se tomamos $\alpha = 1$, sendo o $n = 10$ o grau do polinômio que aparenta convergir mais rápido. Mas dito custo está no ordem de 10^{11} para todos os n .

Na figura 5 observamos que o polinômio grau 10 aparenta estar próximo aos dados reais, mas o erro é grande demais. Portanto, concluímos que a função polinomial não é um bom modelo que ajuste o crescimento de número de casos de Covid-19. Das figuras 6 e 7 observamos também que estão próximas aos dados de treinamento, mas com função custo que tem valores maiores.

Dado que nossa hipótese inicial do ajuste de polinômio não deu certo e usando o fato que visualmente o modelo aparenta ter um crescimento exponencial, vamos supor agora que a função hipóteses é dada por

$$h = h_\theta(x) = \theta_0 e^{\theta_1 x}.$$

Se aplicamos logaritmo natural ambos lados, obtemos

$$\begin{aligned} \ln(h) &= \ln(\theta_0) + \theta_1 x \\ \tilde{h} &= \tilde{\theta}_0 + \theta_1 x. \end{aligned} \tag{4}$$

Já que (4) é linear, então podemos usar a função custo dada em (3) com $\tilde{h}, \tilde{y} = \ln(y)$, $\tilde{\theta} = [\tilde{\theta}_0, \theta_1]^T$ e a matriz X como em (2) para $n = 1$. Logo, usando o algoritmo de gradiente descendente vamos obter o melhor parâmetro $\tilde{\theta}$ tal que tenha menor erro quadrático.

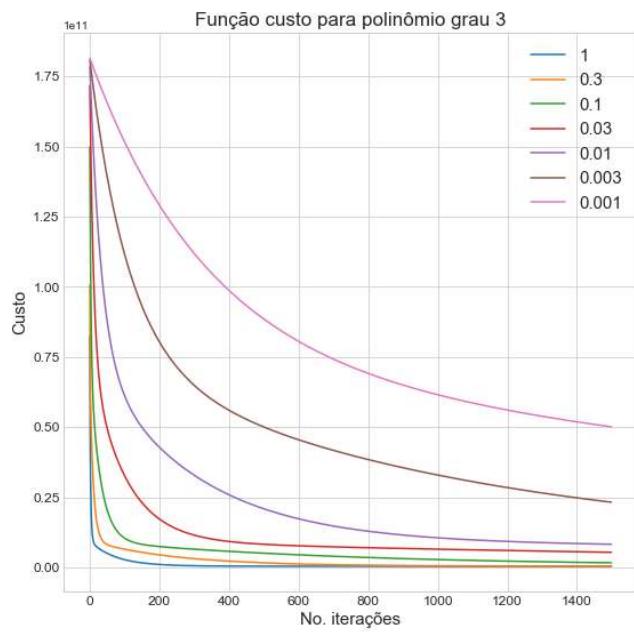


Figura 2: Função de custo com diferentes valores α para polinômio grau 3

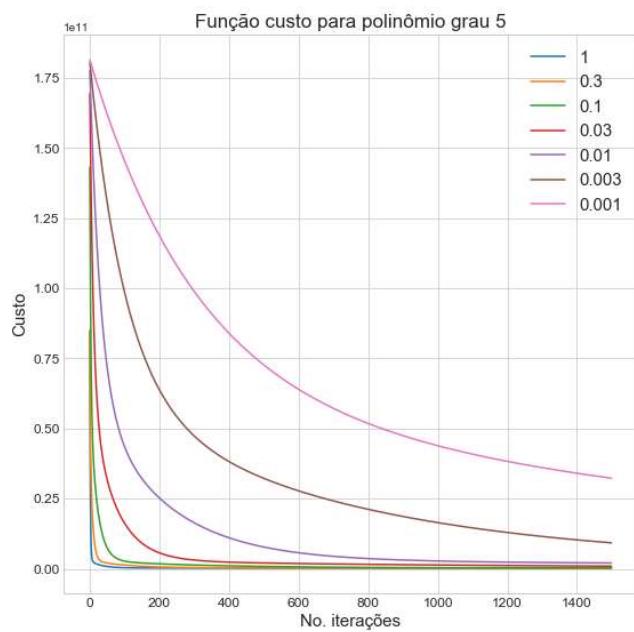


Figura 3: Função de custo com diferentes valores α para polinômio grau 5

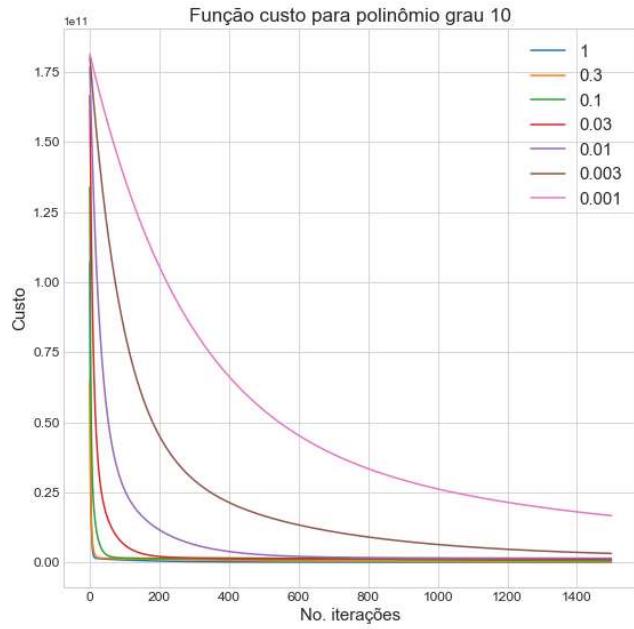


Figura 4: Função de custo com diferentes valores α para polinômio grau 10

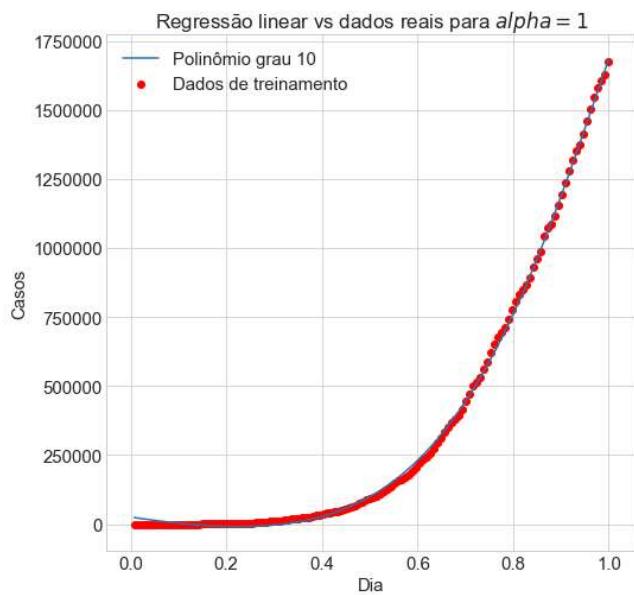


Figura 5: Polinômio grau 10 e $\alpha = 1$.

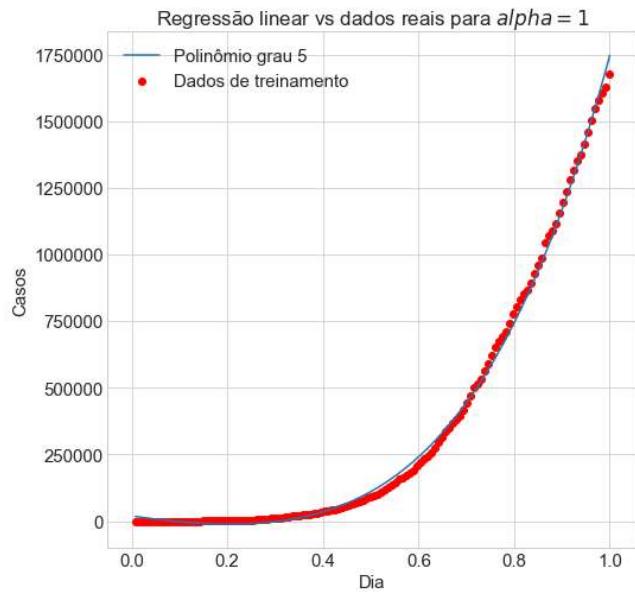


Figura 6: Polinômio grau 5 e $\alpha = 1$.

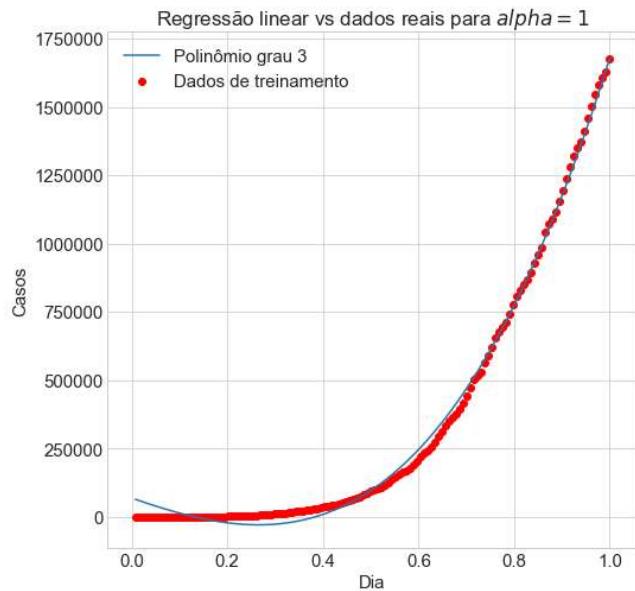


Figura 7: Polinômio grau 3 e $\alpha = 1$.

Uma vez obtido $\tilde{\theta}$, retornamos a nossa função original h_θ , que em termos de $\tilde{\theta}$ temos

$$h_\theta(x) = \exp \left\{ \tilde{\theta}_0 + \theta_1 x \right\}.$$

Tendo em conta o raciocino anterior e usando os valores de $\alpha = 10, 3, 1, 0.3, 0.1, 0.03, 0.01, 0.003, 0.001$, obtemos para $n_i = 1500$ iterações o valor da função custo na figura 8. Para valores de $\alpha = 3$ e $\alpha = 10$ o problema diverge. Observamos também que para $\alpha = 1$ a função custo converge rapidamente. Para o caso particular de $\alpha = 1$ fazemos comparação da função exponencial com os dados reais na figura 9 e podemos concluir que o modelo não se ajusta perfeitamente a uma função exponencial. Nos primeiros 100 dias parece se ajustar muito bem, mas não seria um bom modelo para fazer predição de número de casos para um certo dia maior a 100.

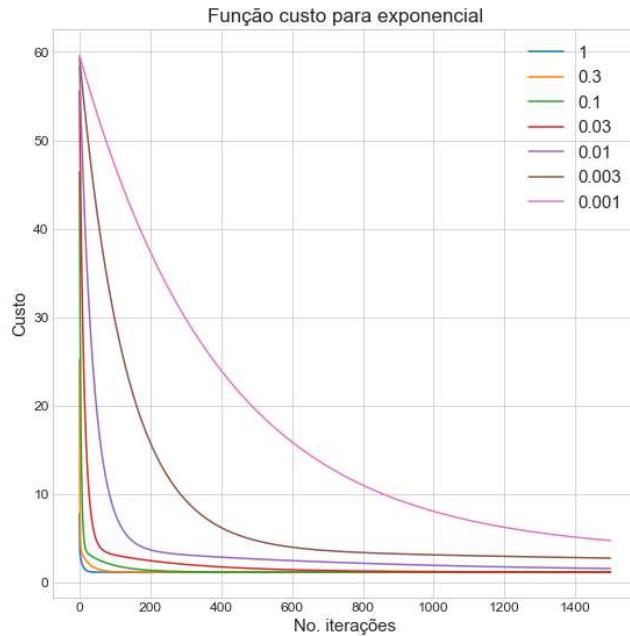


Figura 8: Função de custo para diferentes valores de α .

Dado que temos n relativamente pequeno, poderíamos resolver o problema (4) usando a equação normal, onde temos

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

No algoritmo é recomendado usar pseudoinversa, para evitar matrizes quase singulares. E nosso caso, obtivemos a função exponencial

$$h(x) = \exp \{3.80883983 + 0.09448341x\}. \quad (5)$$

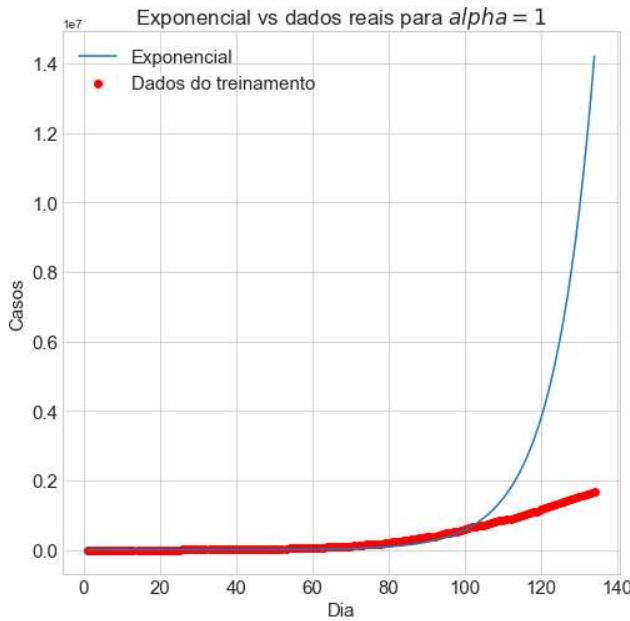


Figura 9: Função exponencial para $\alpha = 1$.

Na figura 10 graficamos a função de hipóteses (5) e comparamos com os dados reais, para assim confirmar que a função exponencial não modela o número de casos de Covid-19 no Brasil.

Os resultados obtidos nesse projeto abre o caminho a explorar outras possíveis funções que modelem o comportamento dos número de casos de Covid-19 no país, fazendo testes com funções exponenciais e polinomiais por exemplo ou como os resultados obtidos em [1], onde concluem que a função de crescimento logístico se ajusta bem para os casos na China.

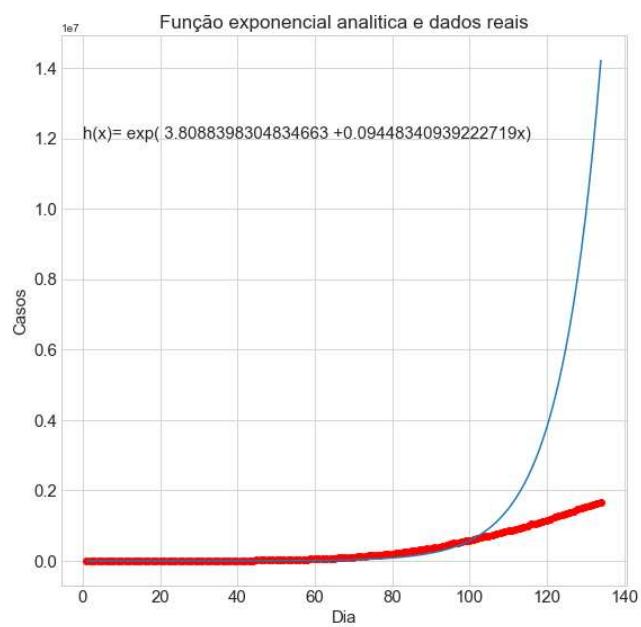


Figura 10: Função exponencial para $\alpha = 1$.

Parte II

Regressão logística

Quando temos classificação binária, podemos usar regressão logística que permite estimar a probabilidade de que um dado pertença a certa classe. Esse método usa a função logística dada por

$$g(z) = \frac{1}{1 + e^{-z}}. \quad (6)$$

Se temos atributos X , parâmetro θ e variáveis de saída $y \in \{0, 1\}$, então a função de hipóteses é definida por

$$\begin{aligned} h_{\theta}(X) &= g(\theta^T X) = \frac{1}{1 + e^{-\theta^T X}} \\ &= \mathbb{P}(y = 1|X; \theta). \end{aligned} \quad (7)$$

Nesse tipo de regressão quisermos encontrar uma fronteira de decisão tal que separe os dados para cada valor de y . Um exemplo dessa fronteira de decisão pode ser $g(z) = 0.5$. Portanto, a escolha do valor θ é importante. Para isso usamos o método de máxima verossimilhança.

A verossimilhança de θ é dada por

$$L(\theta) = \mathbb{P}(y|X; \theta) = (h_{\theta}(X))^y (1 - h_{\theta}(X))^{1-y}.$$

Vamos supor que temos m exemplos de treinamento independentes entre si. Por tanto temos

$$L(\theta) = \prod_{i=1}^m \mathbb{P}(y^{(i)}|X^{(i)}; \theta).$$

O θ ideal é aquele que maximiza $L(\theta)$, por tanto, definimos como função custo

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)}))].$$

Similar no caso de regressão linear, vamos usar o gradiente descendente como algoritmo iterativo para obter o melhor valor do θ para classificar os dados. Usamos o algoritmo 1, mas com a função de custo h_θ dada em (7).

Quando temos mais de dois classes usamos o método um contra todos. Portanto, se temos k classes, vamos ter k classificadores binários.

Uma aplicação da classificação das classes é o reconhecimento de dígitos manuscritos. Para isso vamos a usar a base de imagens MNIST, onde cada dígito é linearizando cada imagem original de tamanho 20×20 para um vetor de comprimento 400 que está complementado com o arquivo *labelMNIST* que contém o dígito correspondente a cada linha do arquivo *imageMNIST*.

Na tabela 11 temos os dados relacionados com o arquivo das imagens MNIST.

0	1	2	3	4	5	6	7	8	9	...	390	391	392	393	394	395	396	397	398	399
0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 400 columns

Figura 11: Tabela dos dados Imagens MNIST.

No algoritmo implementado usamos a função de hipótese e a função de custo anteriormente mencionada na Parte II do documento. Já para, achar a solução (o θ ótimo) executamos o algoritmo do gradiente descendente. Os parâmetros estabelecidos foram $\alpha = 2$ e 1500 como o número de iterações. Ademais, como cada imagem tem escrito a mão um número do 0 até o 9 portanto temos um problema multi-classes pois são 10 classes no total para classificar.

Na figura 12 podemos observar que o valor do custo é bem pequeno conforme as iterações vão aumentando e chegando ao limite definido.

A matriz de confusão M é uma ferramenta que ajuda a identificar quanto efetivo foi a classificação dos dados. A posição M_{ij} contem o número de elementos que foram classificados na classe j e são realmente o número i . Portanto o traço da matriz é igual ao número de elementos classificados corretamente. No caso testado temos que a porcentagem de dados bem previstos é de 94,06%, que pode ser conferido na figura 14. Na figura 13 temos um exemplo, onde os manuscritos desenhados conformam o número oito.

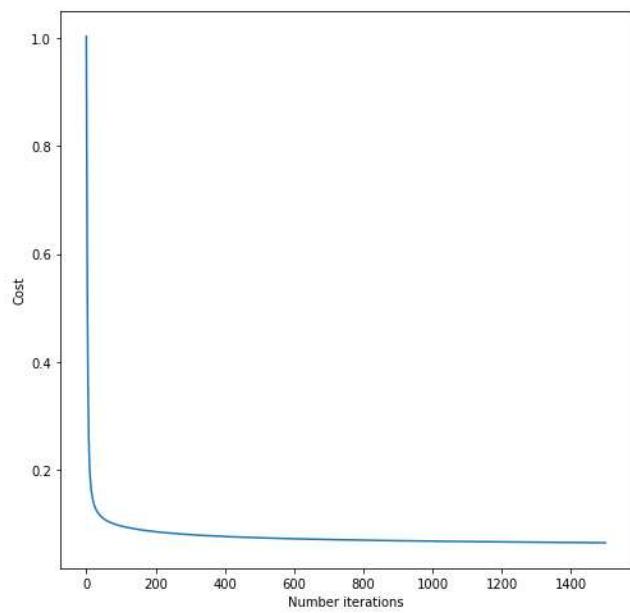


Figura 12: Função de custo sem regularização

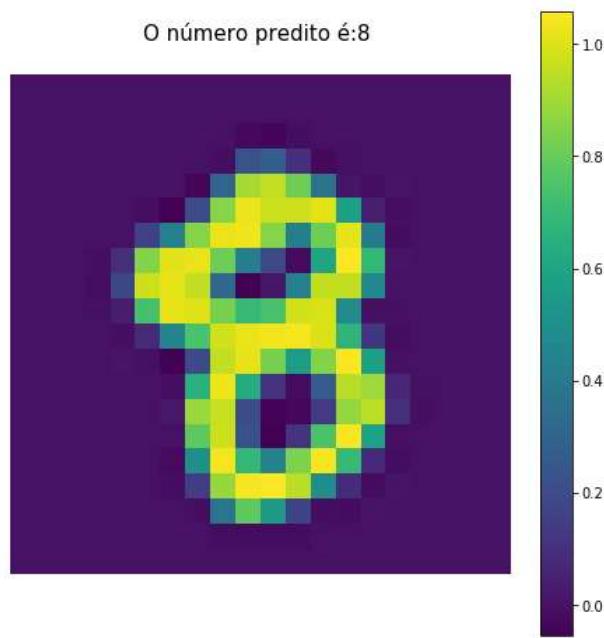


Figura 13: Número previsto dos manuscritos MNIST

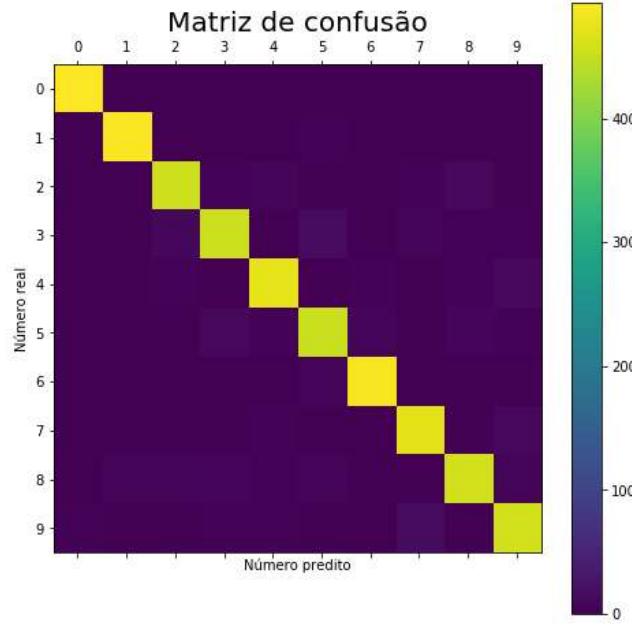


Figura 14: Matriz de confusão.

Por outro lado, também conseguimos predizer as imagens que classificou incorretamente das quais foram 297 (ver anexo 0.1). Ao ver que o número de imagens não classificadas corretamente se viu a necessidade de acrescentar ao algoritmo da regressão logística multi-classes uma regularização. A nova função custo é

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)})(1 - h_\theta(x^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2,$$

e o algoritmo do gradiente descendente

Algorithm 2: Gradiente descendente regularizado

Input: X, k, m, y, α

Output: Vetor θ que aproxima a função de hipóteses aos targets y

```

for  $i \leq k$  do
    if  $j = 0$ 
         $\theta_0 \leftarrow \theta_0 - \frac{\alpha}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$ 
    else:
         $\theta_j \leftarrow \theta_j - \frac{\alpha}{m} \sum_{i=1}^m (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \lambda \theta_j$ 
    end

```

Para este caso consideramos o mesmo número de iterações, o mesmo valor α e aqui

aparece outro parâmetro λ . Como sabemos que se este fosse muito pequeno ele não faria nada pois a função custo ficaria quase igual e tivéramos um underfitting. Decidimos definir como $\lambda = 10$, mas desafortunadamente o resultado não foi esperado pois a porcentagem de dados bem previsto diminuiu a 91,34%, este resultado pode ser visto na figura 15. O valor da função de custo é bem menor que a classificação sem regularização (ver figura 17), mesmo tendo uma porcentagem de predição correta menor.

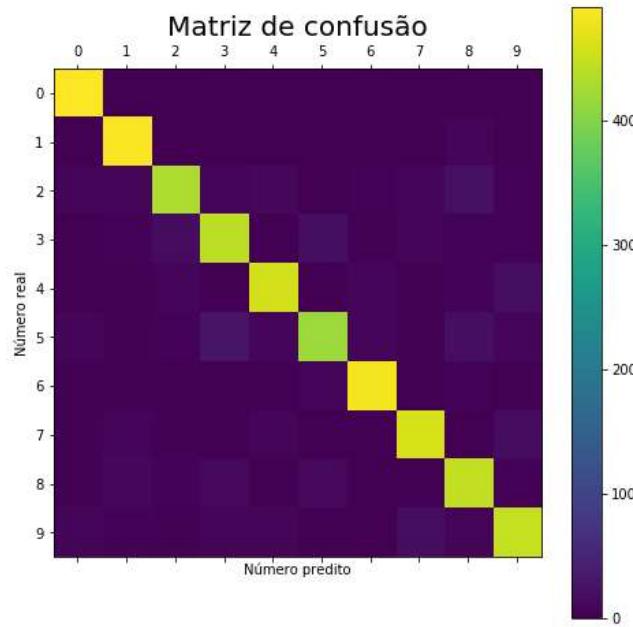


Figura 15: Matriz de confusão para o problema regularizado.

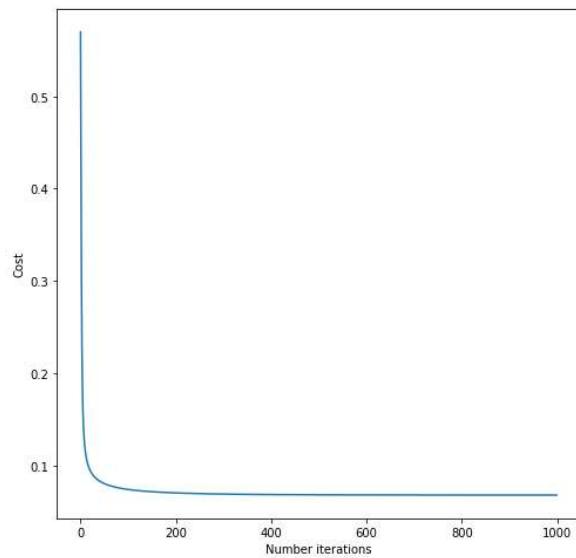


Figura 16: Função de custo sem regularização

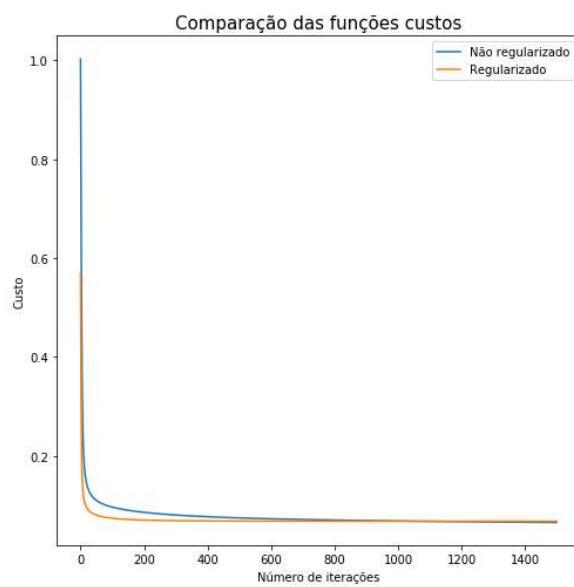


Figura 17: Função de custo sem regularização

0.1 Anexo

A seguinte tabela contem os dados mal classificados para o caso $\alpha = 2$ e 1500 iterações. O índice é o correspondente número na base de dados. (Número real 10 se refere ao 0)

Índice	Número real	Número predito
101	10	5
103	10	5
112	10	3
142	10	8
265	10	4
286	10	6
392	10	6
521	1	5
531	1	5
561	1	5
569	1	8
650	1	8
675	1	8
765	1	2
844	1	5
952	1	5
1006	2	6
1025	2	5
1026	2	7
1041	2	3
1045	2	1
1057	2	5
1062	2	8
1081	2	4
1087	2	9
1090	2	3
1097	2	8
1103	2	7
1112	2	8
1125	2	1
1130	2	9

Índice	Número real	Número predito
1145	2	4
1153	2	9
1169	2	4
1187	2	4
1189	2	8
1207	2	8
1213	2	0
1231	2	8
101	10	5
1271	2	3
1292	2	4
1311	2	8
1362	2	8
1363	2	4
1383	2	8
1399	2	8
1408	2	6
1410	2	8
1412	2	4
1413	2	4
1421	2	8
1423	2	8
1440	2	1
1476	2	7
1484	2	3
1488	2	7
1494	2	0
1498	2	3
1524	3	5
1527	3	7
1550	3	7
1556	3	5
1563	3	2
1564	3	2
1588	3	6
1593	3	9
1599	3	2
1602	3	5

Índice	Número real	Número predito
1607	3	5
1611	3	8
1617	3	9
1626	3	2
1628	3	2
1629	3	2
1630	3	5
1700	3	5
1701	3	7
1707	3	1
1719	3	9
1725	3	1
1732	3	8
1757	3	5
1760	3	5
1763	3	2
1764	3	7
1766	3	5
1776	3	5
1791	3	8
1798	3	5
1801	3	9
1876	3	2
1879	3	2
1905	3	5
1923	3	9
1945	3	8
1959	3	1
1963	3	5
1970	3	7
1972	3	8
1976	3	5
1979	3	7
1981	3	2
2078	4	9
2087	4	6
2093	4	8
2104	4	9

Índice	Número real	Número predito
2110	4	2
2112	4	1
2118	4	9
2166	4	1
2170	4	9
2171	4	9
2187	4	9
2194	4	8
2195	4	2
2201	4	9
2217	4	2
2238	4	9
2263	4	2
2270	4	9
2277	4	8
2304	4	6
2361	4	6
2384	4	9
2393	4	9
2452	4	8
2467	4	6
2498	4	9
2506	5	3
2516	5	8
2520	5	6
2521	5	3
2533	5	9
2549	5	8
2556	5	2
2570	5	7
2574	5	3
2578	5	8
2583	5	6
2593	5	3
2598	5	4
2599	5	9
2601	5	6
2606	5	3

Índice	Número real	Número predito
2607	5	8
2610	5	6
2611	5	6
2616	5	8
2618	5	3
2619	5	8
2620	5	9
2680	5	2
2697	5	6
2698	5	6
2704	5	3
2718	5	3
2739	5	2
2752	5	3
2788	5	0
2790	5	3
2796	5	1
2840	5	8
2850	5	3
2851	5	3
2857	5	9
2907	5	0
2908	5	6
2920	5	0
2937	5	8
2943	5	4
2947	5	4
2952	5	4
2958	5	9
2995	5	3
3043	6	0
3060	6	9
3078	6	5
3081	6	1
3147	6	0
3174	6	5
3204	6	1
3236	6	5

Índice	Número real	Número predito
3328	6	8
3341	6	5
3351	6	5
3382	6	0
3436	6	2
3480	6	5
3515	7	4
3521	7	1
3537	7	1
3563	7	9
3570	7	1
3616	7	4
3619	7	4
3627	7	4
3629	7	9
3634	7	0
3660	7	9
3664	7	5
3668	7	9
3722	7	9
3725	7	0
3732	7	2
3767	7	9
3795	7	2
3823	7	9
3830	7	8
3838	7	9
3853	7	9
3877	7	9
3895	7	8
3908	7	6
3909	7	9
3955	7	9
3972	7	9
4032	8	4
4051	8	6
4057	8	1
4069	8	2

Índice	Número real	Número predito
4076	8	4
4100	8	1
4110	8	7
4125	8	5
4140	8	3
4168	8	9
4177	8	1
4183	8	1
4197	8	9
4231	8	0
4237	8	5
4248	8	9
4250	8	3
4251	8	9
4252	8	9
4254	8	5
4256	8	5
4265	8	4
4294	8	2
4306	8	1
4307	8	1
4319	8	3
4325	8	4
4343	8	2
4362	8	5
4373	8	5
4384	8	2
4393	8	9
4395	8	6
4422	8	3
4428	8	5
4442	8	9
4454	8	2
4455	8	2
4460	8	3
4471	8	5
4477	8	3
4503	9	7

Índice	Número real	Número predito
4506	9	5
4509	9	8
4519	9	7
4526	9	4
4531	9	7
4553	9	4
4568	9	7
4576	9	7
4582	9	2
4590	9	7
4593	9	0
4619	9	4
4627	9	8
4636	9	3
4639	9	3
4656	9	7
4672	9	0
4736	9	5
4752	9	7
4774	9	2
4781	9	2
4783	9	7
4793	9	4
4799	9	5
4801	9	7
4810	9	3
4833	9	3
4844	9	1
4853	9	7
4860	9	7
4863	9	7
4865	9	8
4902	9	7
4906	9	7
4930	9	4
4936	9	0
4944	9	0
4990	9	3
4999	9	7

Bibliografia

- [1] C. Y. Shen, Logistic growth modelling of covid-19 proliferation in china and its international implications, International Journal of Infectious Diseases 96 (2020) 582 – 589. doi:<https://doi.org/10.1016/j.ijid.2020.04.085>.
URL <http://www.sciencedirect.com/science/article/pii/S1201971220303039>
- [2] C. M. Bishop, Pattern recognition and machine learning, springer, 2006.
- [3] J. Friedman, T. Hastie, R. Tibshirani, The elements of statistical learning, Vol. 1, Springer series in statistics New York, 2001.