Projeto I de MS960

Regressão Linear e Classificação Logística

Gianlucca Valente Ribeiro Tesseroli
RA 173047
g173047@dac.unicamp.br

Otávio Moreira Paiano
RA 185284
o185284@dac.unicamp.br

Instituto de Matemática, Estatística e Computação Científica
Universidade Estadual de Campinas

campinas
16 de Outubro de 2020

Sumário

I Regressão Linear

Dado o número de casos de COVID-19 por dia no Brasil por 134 dias, vamos usar Regressão Linear para ajustar a curva por polinômios e por uma exponencial. Em seguida, analisamos qual modelo descreveu melhor o avanço da pandemia no Brasil.

I.i Aproximação polinomial

Após importação dos dados, normalizamos tanto os valores de x (dias) quanto os de y (número de casos), para que, assim, tenhamos um intervalo menor para análise e cálculos.

Conforme o código, utilizamos o gradiente descendente para obter os parâmetros que aproximem a equação obtida ao número de casos. Como é possível ver pelo gráfico dos dados originais, não conseguiríamos uma boa aproximação utilizando um polinômio de grau 1, isto é, uma reta. Desse modo, calculando as potências de x, podemos estimar essa equação com polinômios de maiores graus. Com isso, considerando o grau n, com $n \in \{3, 5, 10\}$, temos os seguintes resultados:

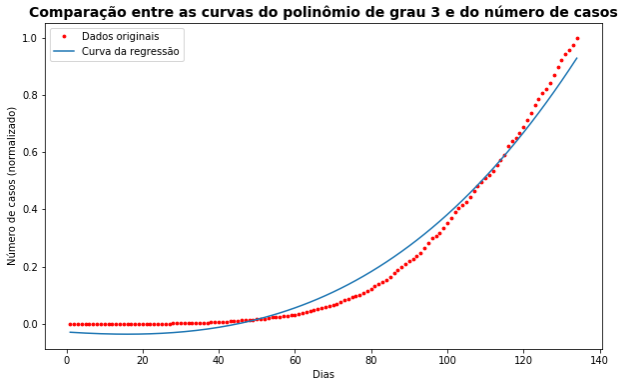

Figura 1: número de casos de COVID-19 por dia (pontos vermelhos) junto ao polinômio de grau $n = 3$ (curva azul) durante os primeiros 134 dias da pandemia.

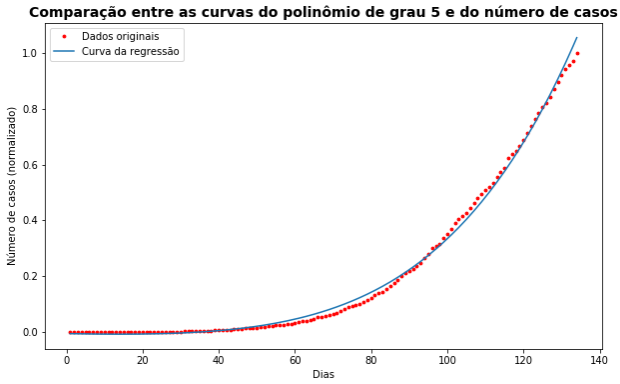

Figura 2: número de casos de COVID-19 por dia (pontos vermelhos) junto ao polinômio de grau $n = 5$ (curva azul) durante os primeiros 134 dias da pandemia.

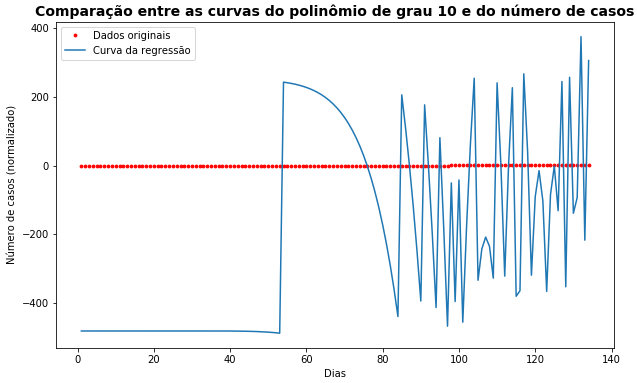

Figura 3: número de casos de COVID-19 por dia (pontos vermelhos) junto ao polinômio de grau $n = 10$ (curva azul) durante os primeiros 134 dias da pandemia.

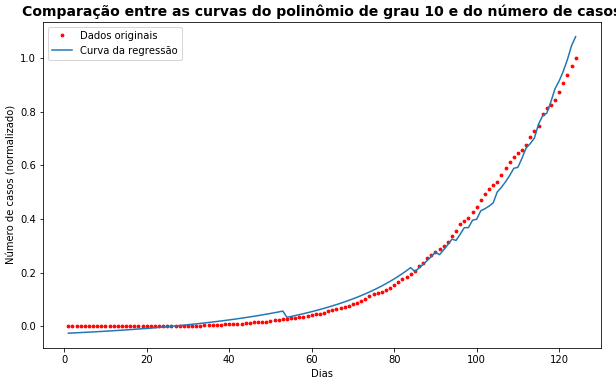

Figura 4: número de casos de COVID-19 por dia (pontos vermelhos) junto ao polinômio de grau $n = 10$, alpha $\alpha = 0.001$ e $max_iter = 10^6$ (curva azul) durante os primeiros 134 dias da pandemia.

Note que para $n = 10$ temos duas estimativas. Isso ocorre, pois com o α definido para $n \in \{3, 5\}$, não obtemos um polinômio bem comportado. Já com essa nova estimativa, temos uma melhor aproximação entre a equação obtida com a regressão linear e o conjunto de dados originais. Porém, a curva que melhor se aproxima aos dados originais, dentre as 3 construídas, é a de grau 5.

I.ii Aproximação exponencial

Para aproximar esses dados por uma curva exponencial do tipo $h(x) = \theta_0 e^{\theta_1 x}$ precisamos encontrar os parâmetros θ_0 e θ_1. Para isso, aproximaremos $log(y)$ por $log(h_\theta)$. Feito isso, obtivemos o seguinte resultado:

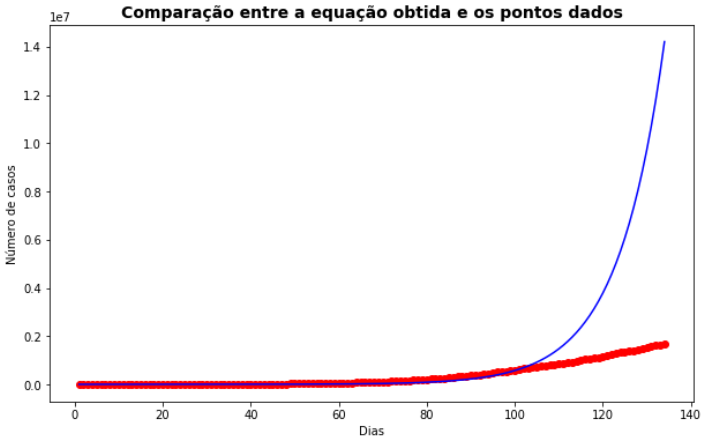

Figura 5: número de casos de COVID-19 por dia (pontos vermelhos) junto à curva exponencial (curva azul) durante os primeiros 134 dias da pandemia.

Com base na equação exponencial descrita no item anterior, alternaremos a taxa de aprendizado α e observaremos o comportamento da função de custo J, em função dessa alteração. Considerando $\alpha \in \{10, 3, 1, 0.3, 0.1, 0.03, 0.01, 0.003, 0.001\}$, temos:

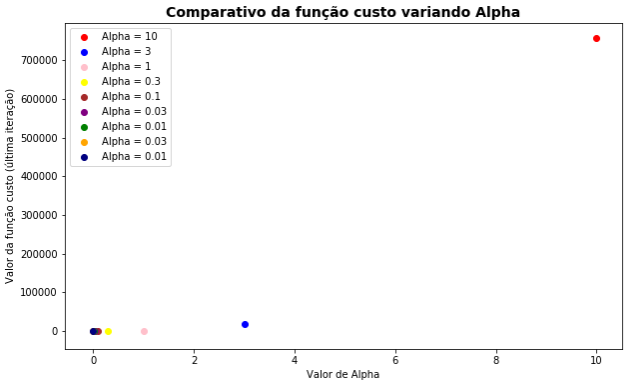

Figura 6: valor da função custo para valores de α ao final das iterações.

Neste caso, a taxa de aprendizado para $\alpha = 10$ é tão grande, que não conseguimos observar a diferença de comportamento dos demais valores $J(\alpha)$. Isso ocorre, pois com uma alta taxa de aprendizado, os parâmetros podem ficar distorcidos e, com isso, acabamos saindo do domínio em que a função de custo tem um bom comportamento. Excluindo o caso de Alpha igual à 10, temos:

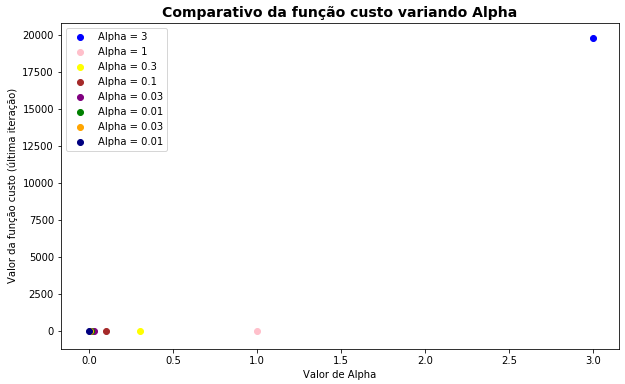

Figura 7: valor da função custo para valores de α ao final das iterações.

Novamente, $J(\alpha = 3)$ possui valor muito acima dos demais. Como feito com o caso anterior, excluiremos esse ponto do conjunto de dados.

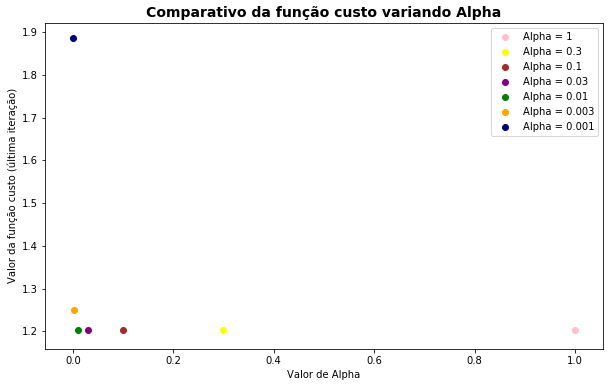

Figura 8: valor da função custo para valores de α ao final das iterações.

Agora, com os valores de J muito menores e condizentes com o que desejamos, isto é, com J tendendo a zero, percebemos que os valores da função de custo, para $\alpha \in \{1, 0.3, 0.1, 0.03, 0.01\}$ estão próximos uns dos outros. No entanto, para os menores valores de Alpha, não foi possível obter um valor tão pequeno para a função de custo. Isto é justificado pela diminuição no passo dado, o que faz com que ela demore mais iterações para chegar até o menor custo. Além disso, podemos observar se, ao menos, a função de custo está diminuindo a cada iteração, conforme esperado. É possível fazer essa análise com um gráfico da função de custo pelas iterações. Tendo isso em vista:

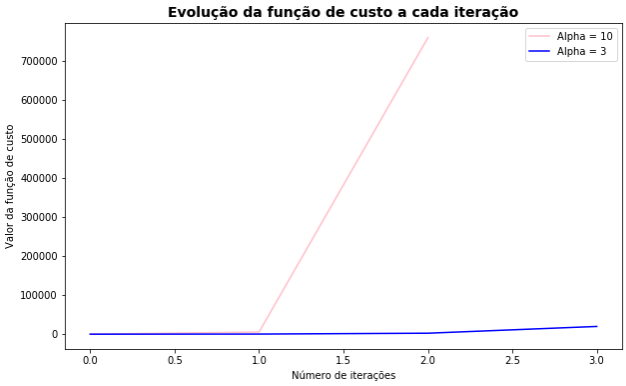

Figura 9: valor da função custo em função das iterações para valores de $\alpha = 3$ e $\alpha = 10$. Nota-se que o algorítmo diverge para esses casos.

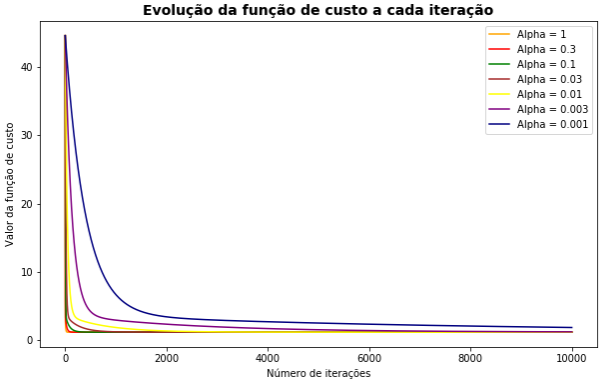

Figura 10: valor da função custo em função das iterações para valores de α que fizeram o algorítmo convergir.

Portanto, para $\alpha \in \{10, 3\}$ temos um aumento da função de custo com o passar das iterações e, por essa distorção, temos valores bem maiores do que quando comparamos com as demais taxas de aprendizado. Excluindo esses dois casos, a função de custo comporta-se exatamente como esperado.

I.iii Minimização pela equação normal

A equação normal realizará uma minimização analítica da função de custo. Nela, temos a vantagem de não precisar inicializar a taxa de aprendizado α, nem o máximo de iterações. Além disso, a resolução pela Equação Normal tem um número conhecido de etapas. Assim, a performance do método é medida em complexidade de tempo, ao invés de taxa de convergência, como ocorre em métodos iterativos como o método do gradiente.

Utilizando este método, obtivemos a seguinte aproximação:

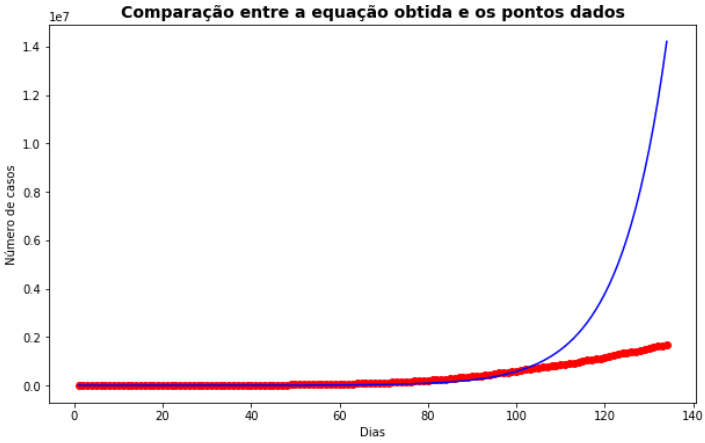

Figura 11: número de casos de COVID-19 por dia (pontos vermelhos) junto à curva exponencial, obtida pela equação normal, (curva azul) durante os primeiros 134 dias da pandemia.

Percebemos que a curva obtida se aproxima dos dados originais até, por volta, do 100° dia.

$$h_\theta(x) = \theta_0 \exp(\theta_1 x)$$

$$\log(n) = \log(\theta_0) + \theta_1 x$$

$$p \equiv \log(h_\theta); \;\; b \equiv \log(\theta_0); \;\; a \equiv \theta_1$$

É possível observar que ambas as curvas exponenciais não se adequam aos pontos originais com grande precisão. Isso acontece, pois quando linearizamos a equação expoencial, obtendo p, conforme demonstrado acima, econtramos parâmetros que minimizam a diferença entre p e y, sendo y o vetor com os resultados originais. No entanto, ao retornar p para a forma exponencial, isto é, h_θ, o erro ganha proporções exponenciais e não minimiza mais a diferença entre h_θ e y, fazendo com que a curva aproximada fique acima dos pontos dados.

II Classificação Logística

Dada a base de imagens MNIST, que contém dígitos manuscritos, e sabendo que cada imagem possui dimensão $20x20$, utilizaremos a Regressão Logística para classificar cada um dos exemplos de maneira mais asssertiva possível.

Em primeiro lugar, foi necessário alterar a base labelMNIST.csv, pois as figuras com número 0 foram dadas como 10. Com isso, fizemos a substituição de tudo que era 10 na label, por 0. Dito isso, seguimos para a construção do algoritmo.

Normalizamos os valores de x e y, isto é, do vetor com a imagem linearizada e das saídas esperadas para cada imagem. Como estamos trabalhando em uma Regressão Logística Multi-classes, executamos a regressão 10 vezes, sendo cada uma delas para obter os parâmetros da equação para aproximação da imagem de cada dígito. Como temos esse conjunto com 10 equações, construímos uma matriz de dimensão 10 x m, na qual m representa o número de treinamentos que temos, em que, a cada linha i, temos as posições do i-ésimo dígito no nosso vetor de labels. Definimos, também, uma matriz que armazena os valores dos parâmetros, tendo dimensão 10 x $(n + 1)$, com n sendo o número de atributos, e com preenchimento semelhante ao explicitado para a matriz anterior. Desse modo, pudemos obter as 10 equações, em que cada uma representa a aproximação um dígito. Para avaliar a força preditiva de nosso modelo, construímos uma função que recebe a imagem vetorizada e um conjunto de parâmetros, para mensurar o valor da sigmoide com essa combinação. Após o cálculo para os 10 conjuntos, tomamos o índice do conjunto de parâmetros que obteve o maior valor obtido, isto é, o mais perto de 1 (indicativo de veracidade na classificação), pois, como a matriz está ordenada de forma que a linha 0 representa possui os parâmetros para o dígito 0, linha 1 possui parâmetros para previsão do dígito 1, e assim por diante, então o índice i representará o dígito i. Temos 4.999 exemplos de treinamento. Para aferir a precisão do algoritmo, executamos o teste para cada exemplo.

II.i Regressão Logística Multi-classes

Tendo em vista tudo o que apresentamos anteriomente e utilizando o algoritmo padrão de Regressão Logística Multi-Clases, temos os seguintes resultados:

```
O números de previsões incorretas é: 1119
Logo, o percentual de acertos é: 77.62 %
```

Figura 12: percentual de acertos do algoritmo.

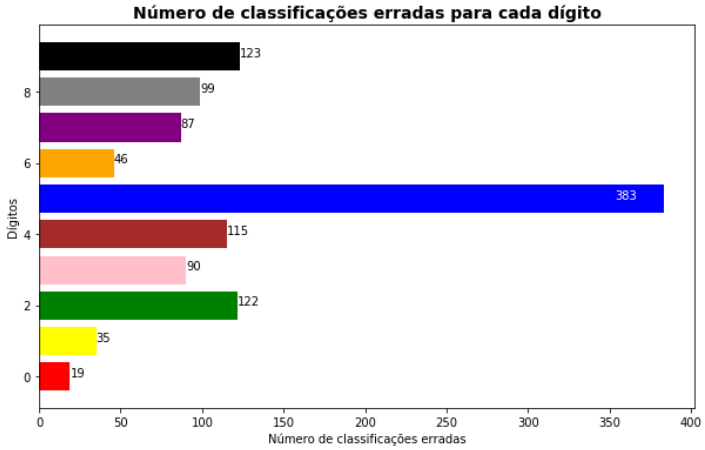

Figura 13: classificações erradas para cada dígito.

Com o gráfico acima, podemos perceber a maioria das classificações erradas foram para o dígito 5. Como temos muitas imagens com equívoco de classificações, colocaremos apenas alguns exemplos. No código, temos todas as imagens printadas, com a classificação correta e qual a saída dada pele programa.

```
O dígito é: 0
O dígito previsto foi: 6
```

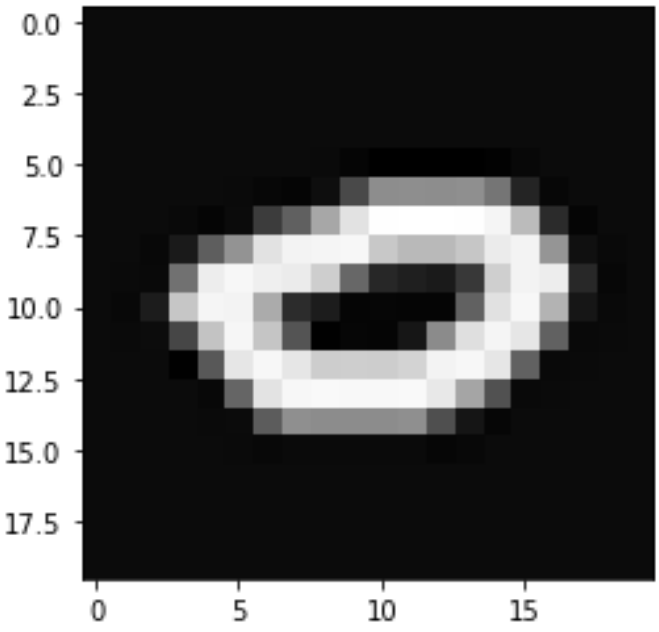

Figura 14: dígito 0 classificado como 6.

```
O dígito é: 1
O dígito previsto foi: 8
```

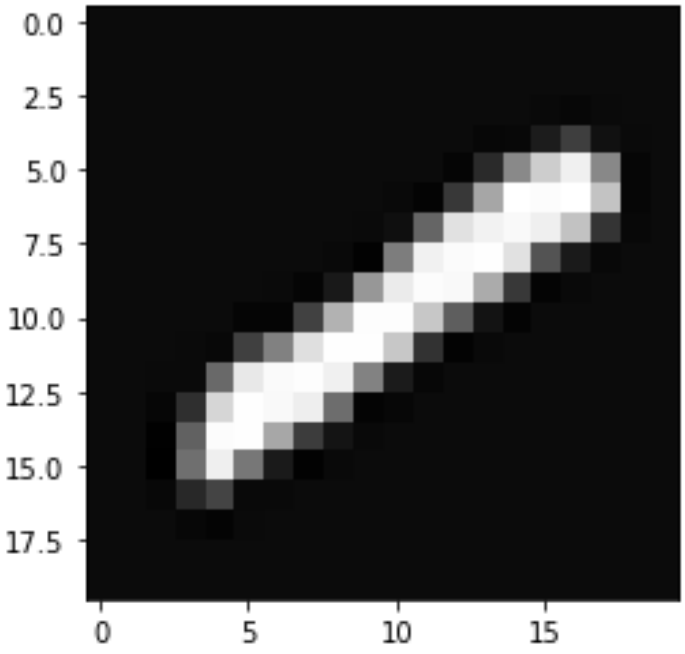

Figura 15: dígito 1 classificado como 8.

```
O dígito é: 2
O dígito previsto foi: 7
```

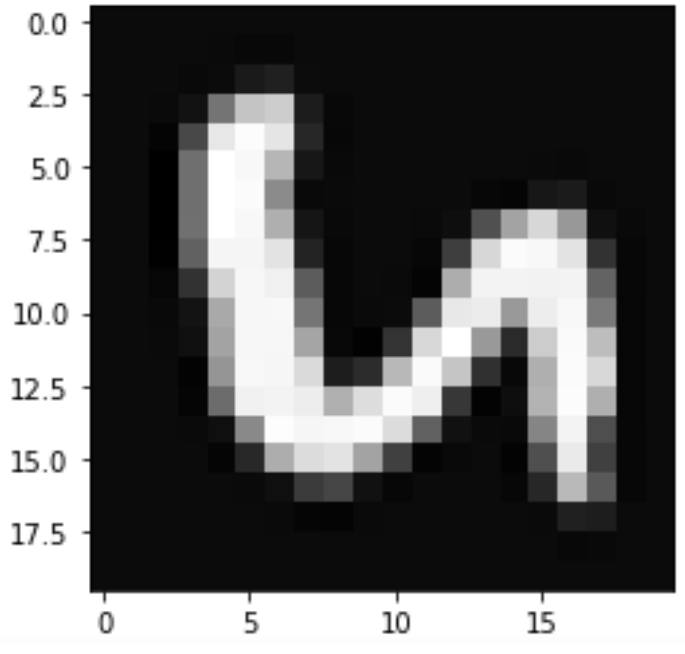

Figura 16: dígito 2 classificado como 7.

```
O dígito é: 3
O dígito previsto foi: 0
```

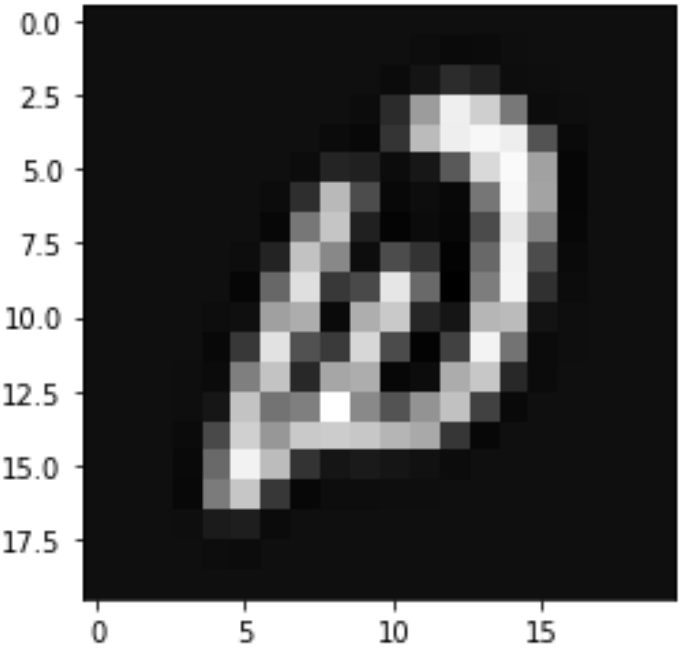

Figura 17: dígito 3 classificado como 0.

```
O dígito é: 4
O dígito previsto foi: 9
```

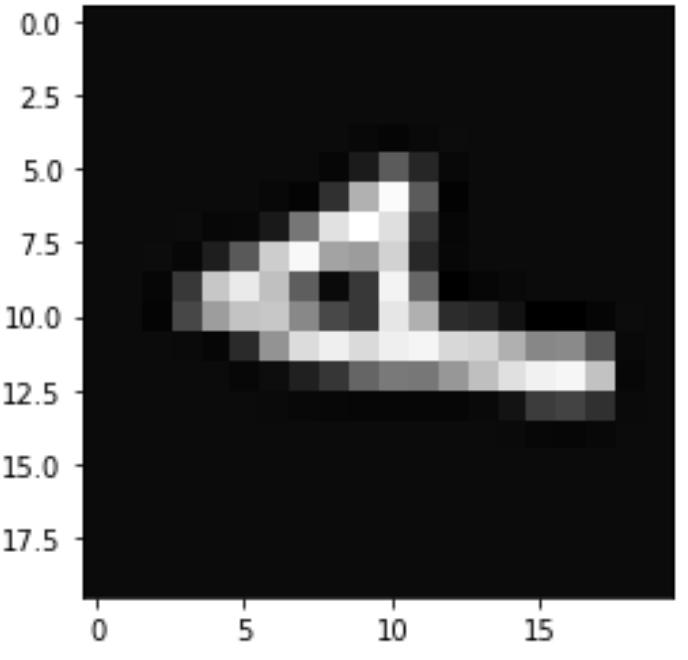

Figura 18: dígito 4 classificado como 9.

II.ii Regressão Logística Multi-classes Regularizada

Agora, implementando o algoritmo de Regressão Logística Multi-classes Regularizada, tivemos uma melhora na previsão. Definimos o lambda, parâmetro de regularização, como $\lambda = 10$.

```
O números de previsões incorretas é: 447
Logo, o percentual de acerto é de: 91.06 %
```

Figura 19: percentual de acertos do algoritmo, utilizando regularização.

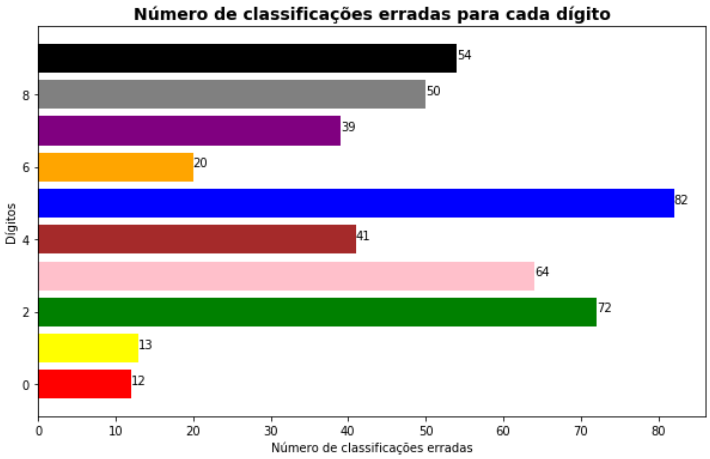

Figura 20: classificações erradas para cada dígito, utilizando regularização.

Novamente, o dígito que possui maior número de classificações equivocadas é o 5, porém, com menor diferença para os demais do que no caso anterior. A seguir, demonstraremos alguns exemplos das imagens erroneamente classificadas.

```
O dígito é: 0
O dígito previsto foi: 8
```

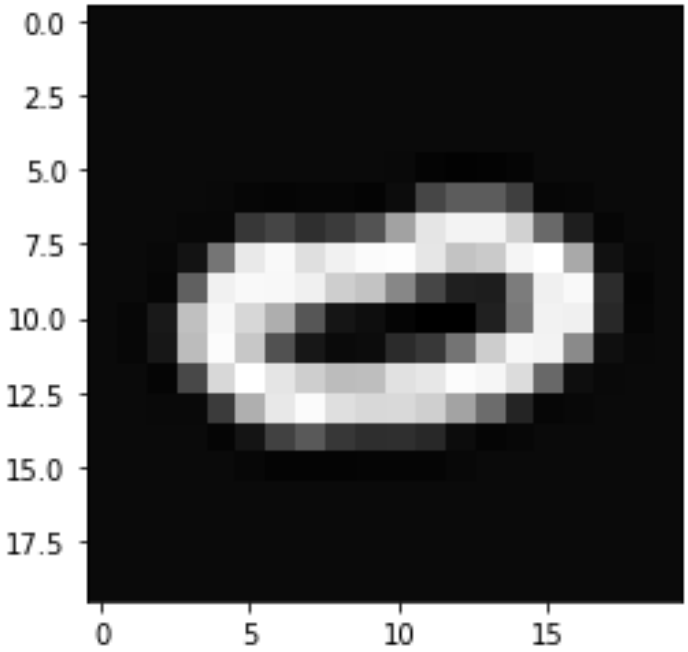

Figura 21: dígito 0 classificado como 8.

```
O dígito é: 2
O dígito previsto foi: 0
```

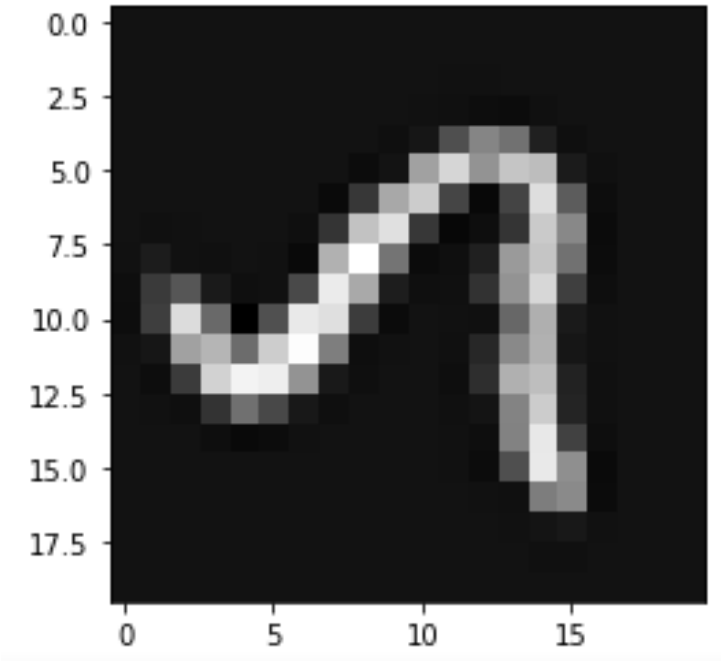

Figura 22: dígito 2 classificado como 0.

```
O dígito é: 6
O dígito previsto foi: 5
```

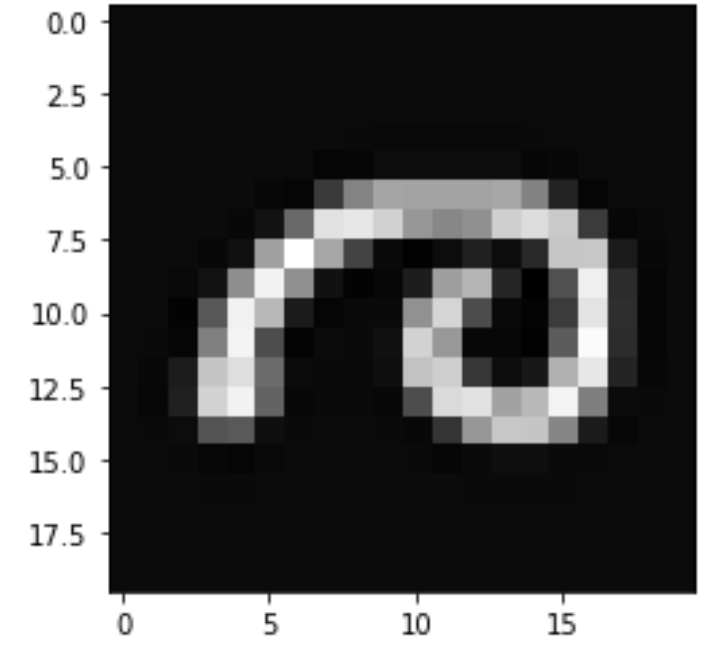

Figura 23: dígito 6 classificado como 5.

```
O dígito é: 7
O dígito previsto foi: 1
```

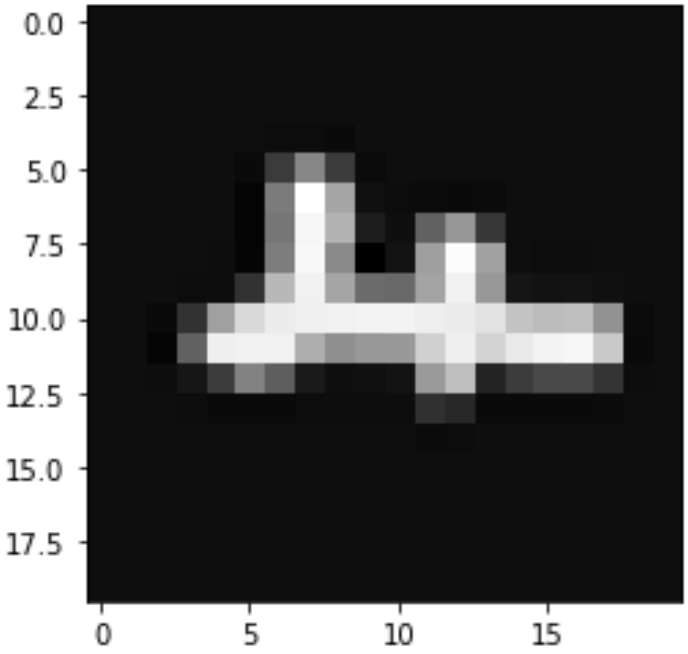

Figura 24: dígito 7 classificado como 1.

```
O dígito é: 8
O dígito previsto foi: 1
```

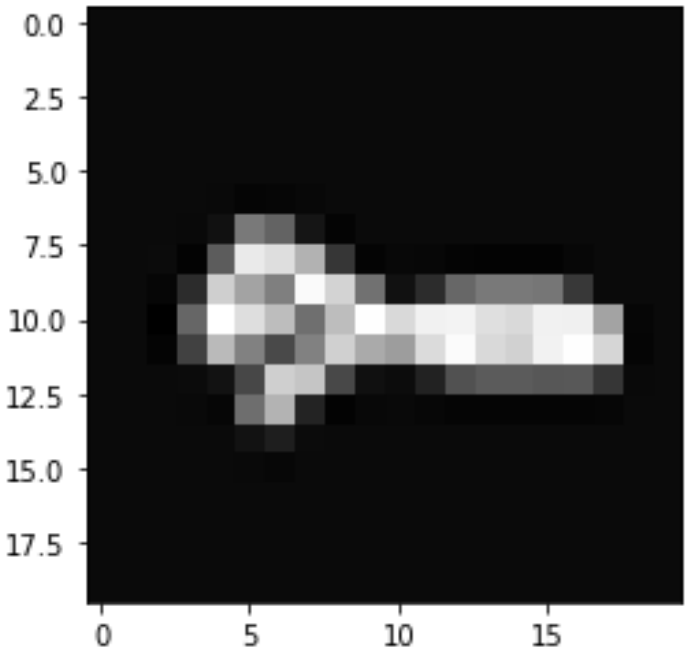

Figura 25: dígito 8 classificado como 1.

III Conclusão

Com base no que foi obtido na primeira parte deste projeto, concluímos que a curva dos casos de COVID-19, no Brasil, apesar de aparentar ser uma variação da curva exponencial, é melhor aproximada por um polinômio de grau $n = 5$. Além disso, ao comparamos os dois modelos de regressão linear, um que utiliza o gradiente descendente e outro, a equação normal, percebemos que obtemos resultados extremamente semelhantes entre eles. Desse modo, a utilização da equação normal pode ser preferida, por não haver necessidade de inicialização de algumas variáveis, economizando espaço computacional.

Já na Regressão Logística, houve mudança signficativa entre os resultados, sendo o melhor deles com a regressão regularizada. No entanto, apesar de possuir alta acuracidade, o algoritmo regularizado demonstra-se mais lento. Para tornar a comparação mais justa, seria necessário definir o máximo de iterações, no gradiente descendente, de forma a conseguir uma performance semelhante ao do método da equação normal. Com isso, é necessária avaliação cuidadosa do caso tratado, para melhor escolha do algoritmo. Além disso, notamos classificações, do próprio arquivo dado *labelMNIST.csv*, estranhas, conforme figuras abaixo:

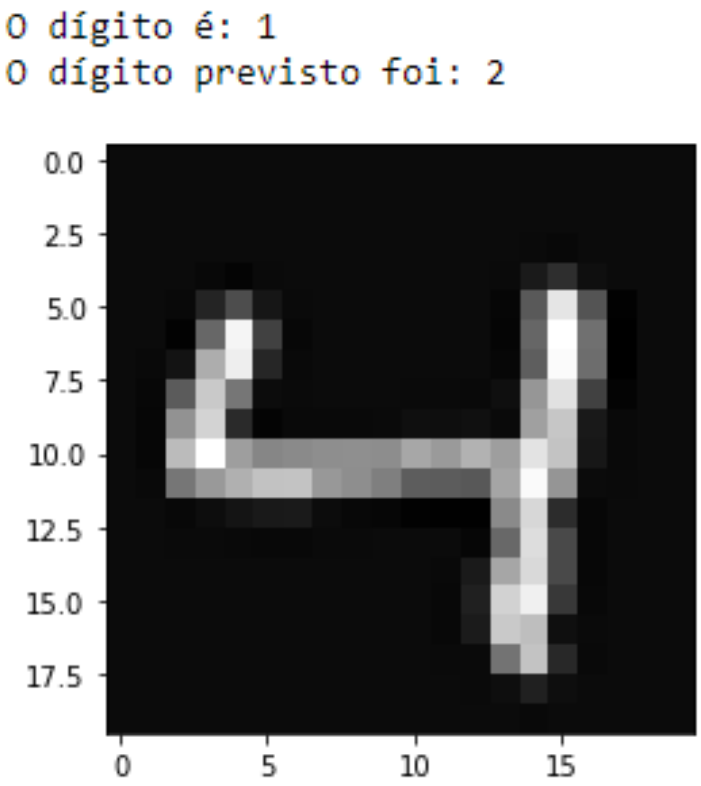

Figura 26: dígito 1 semelhante a um 4.

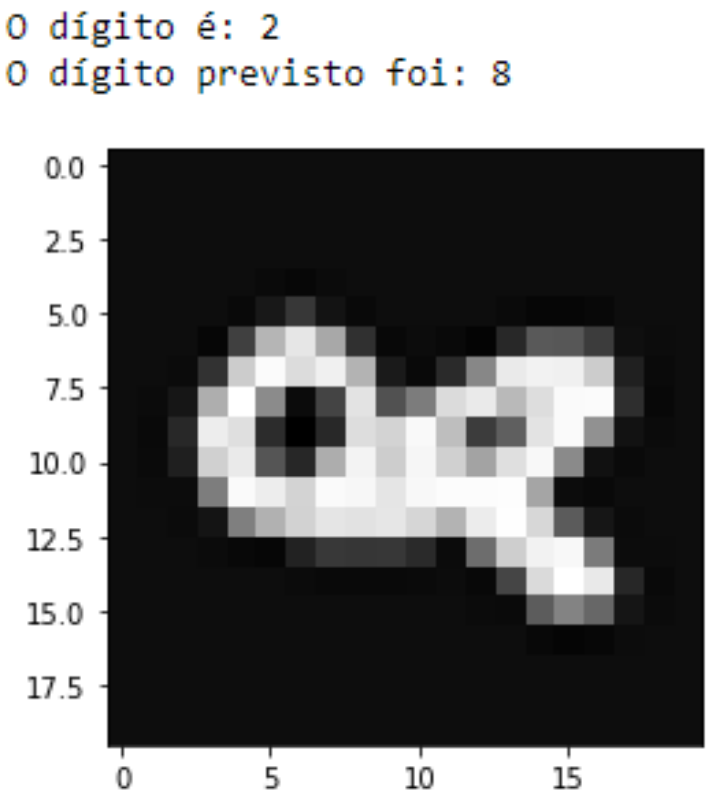

Figura 27: dígito 2 semelhante a um 8.

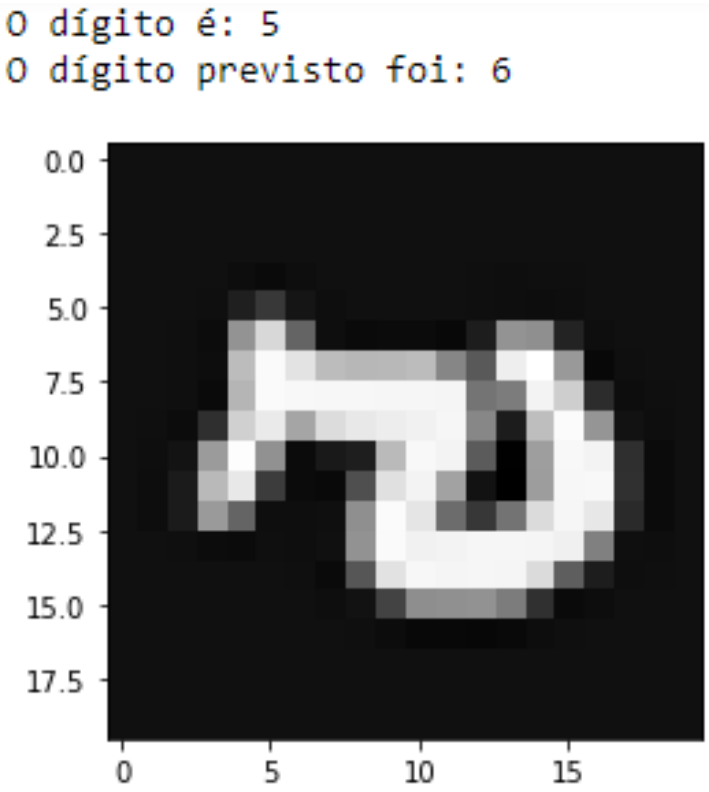

Figura 28: dígito 5 semelhante a um 6.

```
O dígito é: 2
O dígito previsto foi: 1
```

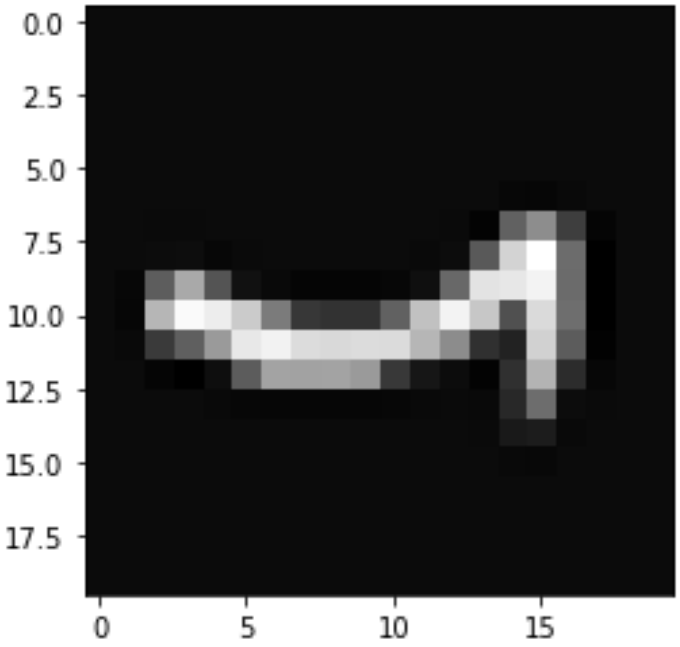

Figura 29: dígito 2 semelhante a um 1.

Como podemos observar, dígitos como os demostrados podem causar problemas, pois, como são usados no treinamento, isto é, na obtenção dos parâmetro para cada equação, pode haver distorção de tais parâmetros e, por consequência, equívocos durante a classificação.

Portanto, apesar dos pontos trazidos como possíveis problemas, todos os métodos funcionaram como esperado, obtendo bons resultados.

Referências

1. T. Hastie, R. Tibshirani and J. H. Friedman, The Elements of Statistical Learning; Data Mining, Inference, and Prediction. Springer, 2009.

2. S. Shalev-Shwartz and S. Ben-David, Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014.

3. C. Bishop, Pattern Recognition and Machine Learning. Springer, 2006.