Trabalho final

SME0212
Otimização Não-Linear

Universidade de São Paulo (USP)

André Guarnier De Mitri - 11395579
Caio Assumpção Rezzadori - 11810481

1 Introdução

Os dados têm sido considerado por muitos "o novo petróleo", uma vez que, devido o aumento da capacidade de armazená-los com o surgimento da era do Big Data, modelos matemáticos preditivos desenvolvidos em cima de extensas bases de dados estão se popularizando e sendo almejados por muitos setores em todo o mundo.

Uma técnica de produção de tais modelos preditivos é pela análise de regressão, onde os parâmetros de uma função selecionada são ajustados aos dados conhecidos de forma que melhor descreva a variância deles. O problema mais clássico é a regressão linear simples, onde estima-se os coeficientes da reta que melhor explique o comportamento de uma variável contínua.

Tal modelo é exemplificado na Figura 1.

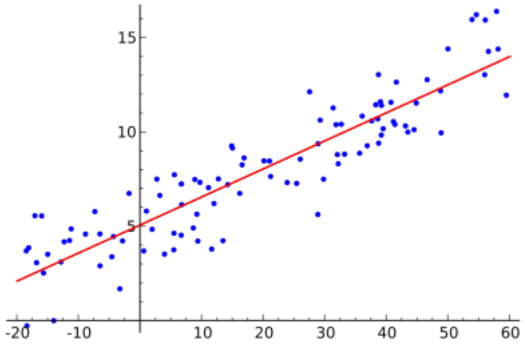

Figura 1: Regressão linear simples

Para o caso da regressão linear simples, há uma forma fechada/analítica para encontrar os coeficientes da reta, que é pelo método dos mínimos quadrados.

Todavia, existe um outro tipo de problema semelhante ao problema de regressão, chamado problema de classificação. Nele, deseja-se prever valores discretos ao invés de contínuos.

Os modelos mais simples de classificação são modelos para respostas binárias, como por exemplo respostas do tipo "Sim" ou "Não", "Masculino" ou "Feminino", entre outros; sendo que as respostas normalmente são representadas numericamente por 1 ou 0.

Isso posto, uma forma de desenvolver um modelo preditivo para respostas binárias é por meio da estimação dos parâmetros da função logística/curva sigmoide, dada por:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \boldsymbol{x}}}$$

onde $\boldsymbol{x}, \boldsymbol{\theta} \in \mathbb{R}^p$, sendo $\boldsymbol{x}$ o vetor de $p-1$ variáveis preditoras e $x_0 = 1$ a primeira componente de $\boldsymbol{x}$, e $\boldsymbol{\theta}$ o vetor dos p parâmetros que se deseja estimar.

Tal estimação fornecerá, na verdade, a função de densidade de probabilidade do evento ser a resposta 1, dadas as variáveis que se tem controle. Ou seja, será estimada $f(\boldsymbol{x}) = P(Y = 1|X = \boldsymbol{x})$, onde Y é a variável resposta (variável que será predita) e X ao vetor de variáveis preditoras (variáveis controláveis).

Para simplificar o problema e para permitir algumas visualizações, será ajustado um modelo para apenas 1 variável preditora. Em outras palavras, deseja-se estimar os parâmetros θ_0 e θ_1 para a seguinte função:

$$f_{\boldsymbol{\theta}}(x) = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}} \tag{1}$$

Cujo comportamento pode ser exemplificado na Figura 2

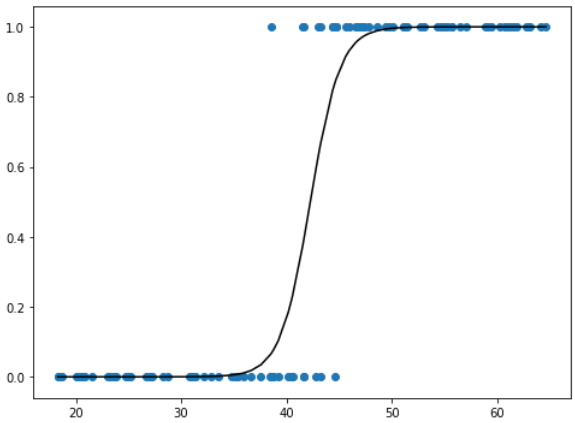

Figura 2: Regressão logística

Uma vez com a função de probabilidade estimada, será utilizado o seguinte critério de classificação para novas predições:

$$C(x) = \begin{cases} 1, \text{ se } f_{\boldsymbol{\theta}}(x) \geq 0.5 \\ 0, \text{ caso contrário} \end{cases} \tag{2}$$

Tal critério será importante quando formos avaliar a adequabilidade do modelo.

2 Modelagem do problema

Entendido o escopo do problema, que é estimar os parâmetros da função sigmoide para um certo conjunto de dados com o intuito de ajustar um modelo de classificação que seja capaz de predizer respostas binárias, analisemos como é a minimização da regressão linear simples por mínimos quadrados, primeiramente.

Deseja-se encontrar um modelo que minimize a soma dos quadrados dos erros aleatórios, o que fornece o seguinte problema de otimização:

$$min \quad Q = \sum_i \epsilon_i^2 = \sum_i (Y_i - \theta_0 - \theta_1 X_i)^2$$

Utilizando técnicas de cálculo diferencial, nós conseguimos encontrar os estimadores da reta dados por $\hat{\boldsymbol{\theta}} = \begin{pmatrix} \hat{\theta}_0 \\ \hat{\theta}_1 \end{pmatrix}$ quando calculamos $\left.\frac{\partial Q}{\partial \boldsymbol{\theta}}\right|_{\boldsymbol{\theta}=\hat{\boldsymbol{\theta}}} = \mathbf{0} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. Esse cálculo fornece os estimadores que minimizam Q pois trata-se de uma função convexa.

Quando deseja-se estimar a curva sigmoide, todavia, tem-se que a função dada por

$$Q = \sum_i \epsilon_i^2 = \sum_i \left(Y_i - \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}} \right)^2$$

não será convexa, o que significa que fornecerá um ou mais mínimos locais, máximos locais, e/ou pontos de inflexão. Dito isso, em vez de utilizar a função de perda dos quadrados dada por Q, utiliza-se a função "Log Loss" dada por:

$$\text{Log Loss}(f_{\boldsymbol{\theta}}(x), y) = \begin{cases} -\log[f_{\boldsymbol{\theta}}(x)], \text{ se } y = 1 \\ -\log[1 - f_{\boldsymbol{\theta}}(x)], \text{ se } y = 0 \end{cases}$$

Podendo ser reescrita na forma:

$$\text{Log Loss}(f_{\boldsymbol{\theta}}(x), y) = -y \log[f_{\boldsymbol{\theta}}(x)] - (1 - y) \log[1 - f_{\boldsymbol{\theta}}(x)]$$

onde $f_{\boldsymbol{\theta}}(x)$ é a função logística estimada e y a observação do dado binário.

Esta função de custo atribui uma punição ao modelo quando a função de densidade de probabilidade ($f_{\boldsymbol{\theta}}(x)$) está mais próxima do oposto da resposta esperada do dado (y). Isso posto, como vamos ajustar um modelo para um conjunto de m dados, queremos estimar os parâmetros de $f_{\boldsymbol{\theta}}(x)$ pelo seguinte problema de otimização:

$$min \quad J(\boldsymbol{\theta}) = \frac{1}{m} \sum_i^m \text{Log Loss}(f_{\boldsymbol{\theta}}(x_i), y_i)$$

$$= \frac{1}{m} \sum_i^m (-y_i \log[f_{\boldsymbol{\theta}}(x_i)] - (1 - y_i) \log[1 - f_{\boldsymbol{\theta}}(x_i)])$$

Ou, em notação matricial:

$$min \quad J(\boldsymbol{\theta}) = \frac{1}{m}(-\boldsymbol{y}^T \log[\boldsymbol{f_\theta}(\boldsymbol{x})] - (\mathbf{1} - \boldsymbol{y})^T \log[\mathbf{1} - \boldsymbol{f_\theta}(\boldsymbol{x})]) \tag{3}$$

onde:

- $\boldsymbol{f_\theta}(\boldsymbol{x})$ é o vetor de valores da função logística ajustada sobre os parâmetros $\boldsymbol{\theta}$ aplicada sobre os respectivos dados preditores representados por $\boldsymbol{x}$;
- $\boldsymbol{y}$ é o vetor de observações/dados da resposta que o modelo será ajustado sobre;
- $\mathbf{1}$ é o vetor de mesma dimensão que $\boldsymbol{y}$ e de $\boldsymbol{f_\theta}(\boldsymbol{x})$ cujas componentes são iguais a 1.

Uma vez que a função de custo $J(\boldsymbol{\theta})$ é convexa [3], como será mostrado na próxima seção, então garante-se que a minimização do problema irrestrito (3) fornecerá um ponto ótimo global, encontrando os melhores estimadores para o modelo.

Por fim, tem-se que o gradiente da função de custo é dado por:

$$\nabla J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i}^{m}\begin{pmatrix} f_{\boldsymbol{\theta}}(x_i) - y_i \\ x_i[f_{\boldsymbol{\theta}}(x_i) - y_i] \end{pmatrix} = \frac{1}{m}X^T(\boldsymbol{f_\theta(x)} - \boldsymbol{y}) \tag{4}$$

onde $X = \begin{pmatrix} | & | \\ \mathbf{1} & \boldsymbol{x} \\ | & | \end{pmatrix}$

Para uma instância da resolução deste problema, foi utilizada a base de dados "Breast Cancer Wisconsin Data Set" retiradas do site *Kaggle*, que fornece informações de uma amostra de 569 pacientes com tumores. O objetivo será ajustar o modelo de regressão logística discutido de tal forma que seja possível predizer se o tumor de um paciente (cujos dados estão na coluna "*diagnosis*") é maligno (valor categórico "M" e que será representado por 1) ou benigno (valor categórico "B" e que será representado por 0) a partir do raio médio do tumor (cujos dados estão na coluna "*radius_mean*").

3 Implementação

Foi utilizada a linguagem *Python 3.0* na plataforma web *Google Colaboratoy* para a implementação dos algoritmos iterativos para a resolução do problema irrestrito (3). As bibliotecas mais utilizadas e seus propósitos foram:

- *numpy (np)*: Manipulação de operações matriciais;
- *pandas (pd)*: Leitura e manipulação da base de dados;
- *matplotlib.pyplot (plt)*: Geração de gráficos.

3.1 Funções principais

- Função logística/sigmoide em (1):

```
def f_theta(theta, X = X):
  return 1/(1 + np.exp(-X@theta))
```

  onde "theta" é a estimação do vetor de parâmetros em uma certa iteração e "X" a variável global com os valores fixos da variável preditora;

- Função de custo (2):

```
def J(theta, X=X, y=y, m=m):
    J = -y.T@np.log(f_theta(theta, X)) - (1-y).T@np.log(1-
    f_theta(theta, X))
    J = (1/m)*J
    return J
```

  onde "y" é a variável global com os valores fixos da variável resposta e "m"o número fixo de amostras/pacientes;

- Gradiente da função de custo (4):

```
def J_grad(theta, X=X, y=y, m=m):
    dev = X.T@(f_theta(theta, X) - y)
    dev = (1/m)*dev
    return dev
```

Vale ressaltar, que o caráter "@" significa produto matricial.

Uma vez com as funções principais implementadas, mostremos os métodos de programação não-linear utilizados para a resolução do problema e suas implementações.

Foram utilizados métodos de busca linear. Isso significa que as iterações para encontrar os estimadores $\boldsymbol{\theta_k}$ que minimizam a função $J(\boldsymbol{\theta})$ são do tipo:

$$\boldsymbol{\theta_{k+1}} = \boldsymbol{\theta_k} + \alpha_k \boldsymbol{p_k}$$

Onde α_k é o tamanho do passo e $\boldsymbol{p_k}$ é a direção da iteração.

O passo inicial utilizado para os dois métodos escolhidos foi $\boldsymbol{\theta}_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$

3.2 Gradiente descendente

No caso do gradiente descendente, tem-se que $\boldsymbol{p_k} = -\nabla J(\boldsymbol{\theta_k})$. Além disso, mesmo que tenha sido mostrado recentemente que o tamanho de passo dado por $\alpha'_k = \dfrac{2m}{||X||_F}$ é o melhor para a regressão logística [3], foi utilizado um tamanho de passo menor dado por $\alpha_k = 0.01\alpha'_k$, uma vez que utilizar α'_k provocou erros numéricos, como o cálculo de $\log 0$ em algumas iterações, e também a divergência numérica dos parâmetros.

A implementação do método segue abaixo.

```python
# Tamanho do passo(alpha)
a = 0.01*2*m/np.linalg.norm(X)

# Chute inicial
theta_k = np.zeros(n)

# Lista das iteracoes
J_list = []
theta_list = []

for i in range(2*10**5):
    J_val = J(theta_k) # Custo do theta da iteracao atual
    J_list.append(J_val) # Guardando custo da iteracao

    theta_k = theta_k - a*J_grad(theta_k) # Atualizando os
    parametros
    theta_list.append(list(theta_k)) # Guardando parametro da
    iteracao

    if(abs(J_grad(theta_k)[0]) <= 1e-15 and abs(J_grad(theta_k)[1])
     <= 1e-15):
        # Condicao de parada de primeira ordem
        print(f"Parou na iteracao {i+1}")
        break
```

3.3 Método de Newton

No método de newton, a direção de descida é dada por $\boldsymbol{p_k} = -(\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta_k}))^{-1} \nabla J(\boldsymbol{\theta_k})$ onde $\nabla^2_{\boldsymbol{\theta}} J$ é a Hessiana da função de custo. Seu cálculo segue abaixo.

$$\nabla^2_{\boldsymbol{\theta}} J_i(\boldsymbol{\theta}) = \frac{\partial^2 J_i(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} = \boldsymbol{x_i}^T \boldsymbol{x_i} f_{\boldsymbol{\theta}}(\boldsymbol{x})(1 - f_{\boldsymbol{\theta}}(\boldsymbol{x})) \frac{1}{m}$$

alternativamente,

$$D_{ii} = f_\theta(x_i)(1 - f_\theta(x_i))$$
$$\Rightarrow \nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \frac{1}{m} X^T D X$$

onde $X = \begin{pmatrix} | & | \\ \boldsymbol{x_0} & \boldsymbol{x_1} \\ | & | \end{pmatrix} = \begin{pmatrix} | & | \\ \mathbf{1} & \boldsymbol{x} \\ | & | \end{pmatrix}$, para este problema (1 variável preditora) e D é uma matriz diagonal.

Mostremos que tala Hessiana é semi-definida positiva para qualquer $\boldsymbol{\theta}$:

$$\begin{aligned} a^T \nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) a &= a^T X^T D X a \\ &= a^T X^T D^{1/2} D^{1/2} X a \\ &= ||D^{1/2} X a||_2^2 \\ &\geq 0 \end{aligned}$$

Isso implica que a função de custo, $J(\boldsymbol{\theta})$ é convexa para todo $\boldsymbol{\theta}$.

A implementação da Hessiana e da direção de descida foram feitas, respectivamente, da seguinte maneira:

```
def J_hessian(theta, X=X, y=y, m=m):
    hess = (1/m)*X.T @ np.diag(f_theta(theta) * (1-f_theta(theta))) @ X
    return hess
```

```
def pk(theta, X=X, y=y, m=m):
    hessiana = J_hessian(theta)
    mult = np.linalg.solve(hessiana, -J_grad(theta))
    return mult
```

Percebe-se que, ao invés de inverter a matriz Hessiana, foi calculado a solução do sistema linear $\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta_k}) \boldsymbol{p_k} = -\nabla J(\boldsymbol{\theta_k})$, uma vez que calcular inversas de matrizes tem um custo computacional maior.

Para o método de Newton, utilizou-se um tamanho de passo fixo dado por $\alpha_k = 1$.

Sua implementação é:

```
# Tamanho do passo(alpha)
a = 1

# Chute inicial
theta_k = np.zeros(n)

# Lista das iteracoes
J_list = []
theta_list = []

for i in range(100):
    J_val = J(theta_k) # Custo do theta da iteracao atual
    J_list.append(J_val) # Guardando custo da iteracao

    theta_k = theta_k + a*pk(theta_k) # Atualizando os parametros
    theta_list.append(list(theta_k)) # Guardando parametro da iteracao

    if(abs(J_grad(theta_k)[0]) <= 1e-15 and abs(J_grad(theta_k)[1]) <= 1e-15):
        # Condicao de parada de primeira ordem
        print(f"Parou na iteracao {i+1}")
        break
```

4 Resultados numéricos

4.1 Gradiente descendente (Convergindo)

Tabela 1: Resultados principais

Iterações	θ_0	θ_1	$J(\boldsymbol{\theta})$	$\nabla J(\boldsymbol{\theta})$
200.000	-15.238	1.033	0.289	[7.698e-06 -5.381e-07]

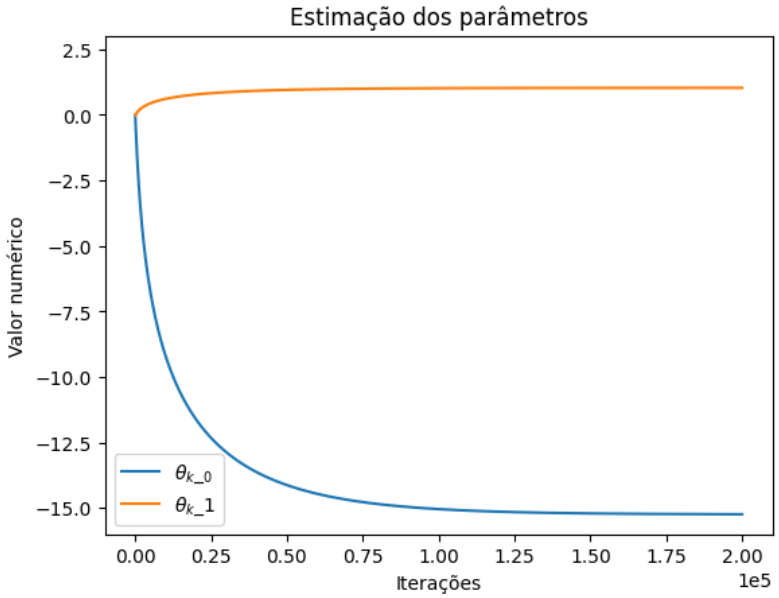

Figura 3: Iterações dos parâmetros

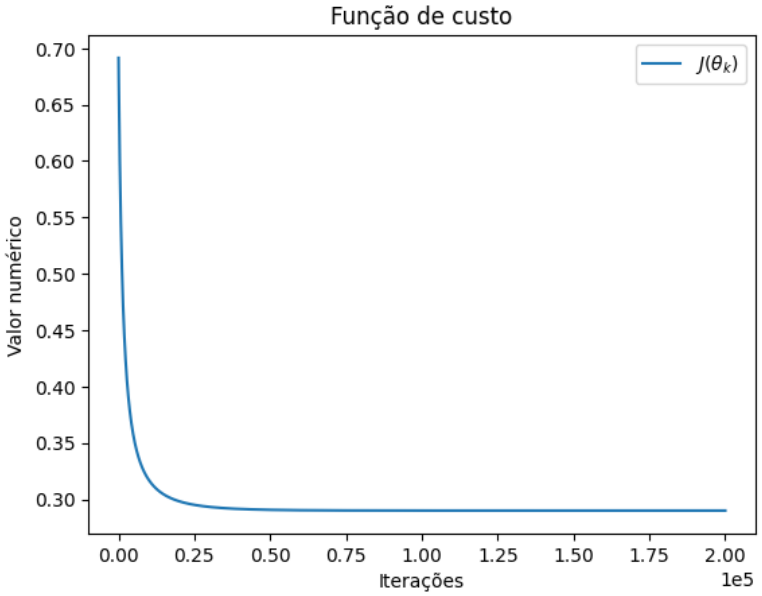

Figura 4: Iterações da função de custo

Observa-se que a partir de uma certa quantidade de iterações, a função de custo não apresenta muitas mudanças, indicando que ela atingiu seu mínimo.

O comportamento da regressão logística ajustada aos dados pode ser vista abaixo.

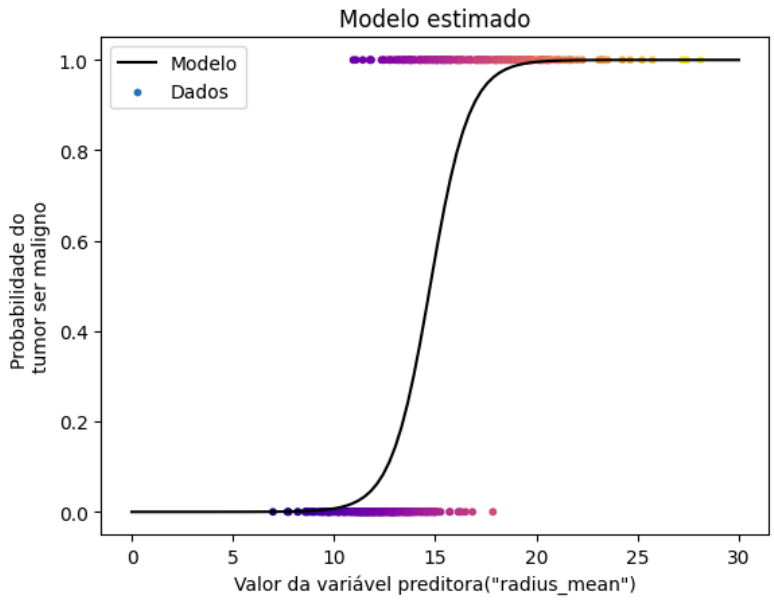

Figura 5: Regressão logística ajustada

4.2 Gradiente descendente (Divergindo)

Quando utilizou-se o tamanho de passo α'_k, os seguintes resultados foram obtidos:

Tabela 2: Resultados principais

Iterações	θ_0	θ_1	$J(\boldsymbol{\theta})$	$\nabla J(\boldsymbol{\theta})$
200.000	-461.745	35.237	nan	[0.160 2.323]

Onde "nan" significa que não foi possível calcular $J(\boldsymbol{\theta})$. Percebe-se que os parâmetros divergiram muito do que foi calculado anteriormente. Além disso, o valor das componentes do gradiente da função são relativamente altos, indicando que o método não convergiu.

O comportamento dos parâmetros e da função de custo seguem abaixo.

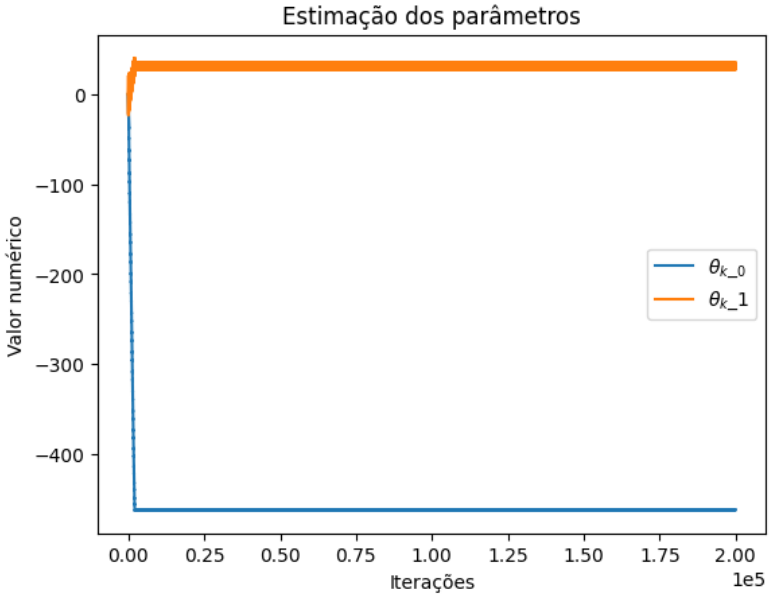

Figura 6: Iterações dos parâmetros

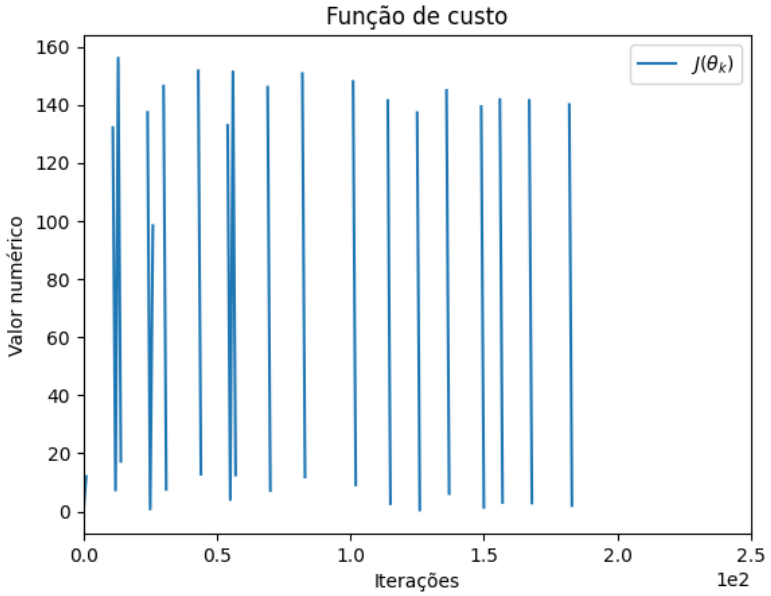

Figura 7: Iterações da função de custo

Para este caso, o comportamento da regressão logística ajustada aos dados pode ser vista abaixo.

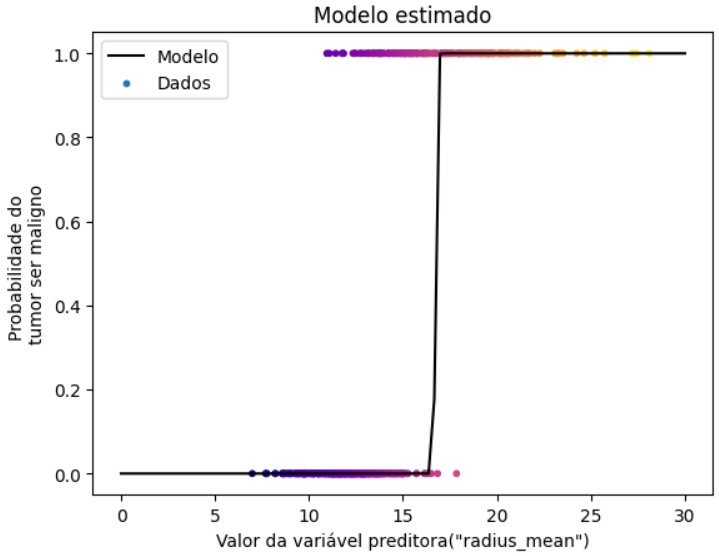

Figura 8: Regressão logística ajustada

Uma possível interpretação de tal resultado, é que o modelo "decorou" os dados, ao invés de se ajustar a eles. Tal acontecimento é chamado de"*Overfitting*".

4.3 Newton

Tabela 3: Resultados principais

Iterações	θ_0	θ_1	$J(\boldsymbol{\theta})$	$\nabla J(\boldsymbol{\theta})$
8	-15.245	1.033	0.289	[-6.170e-17 -8.433-16]

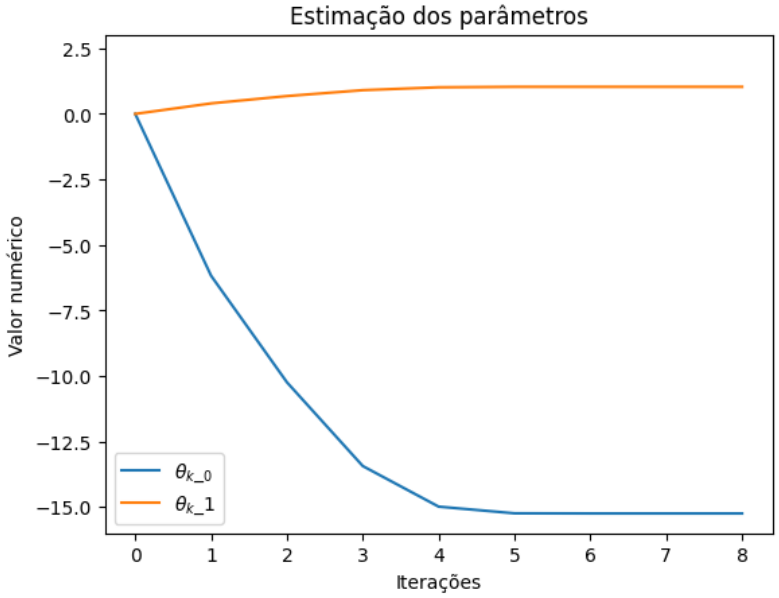

Figura 9: Iterações dos parâmetros

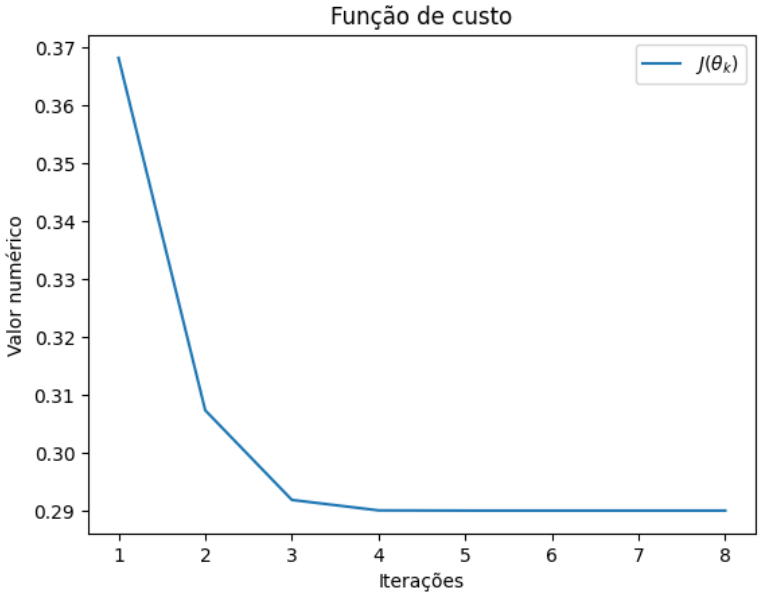

Figura 10: Iterações da função de custo

O gráfico do modelo ajustado, exibido abaixo, é praticamente idêntico ao do modelo ajustado pelo método dos gradientes descendentes, uma vez que os parâmetros estimados foram muito próximos.

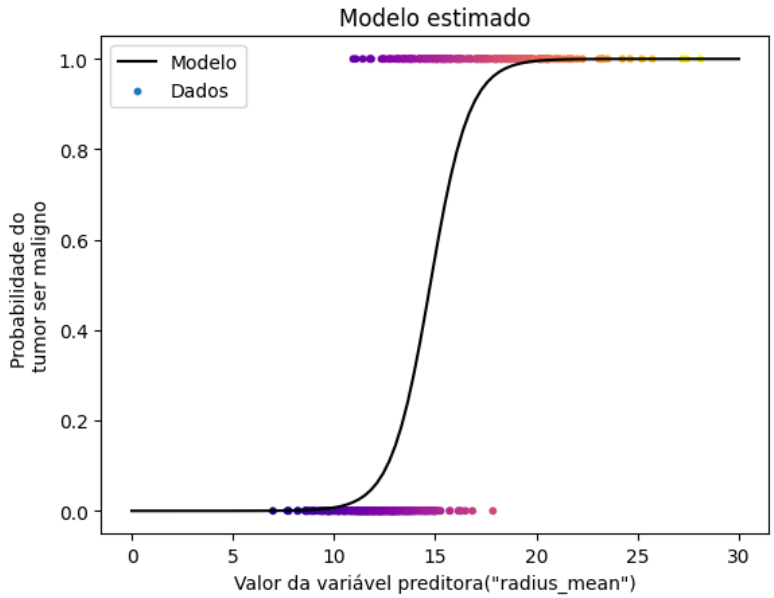

Figura 11: Regressão logística ajustada

4.4 Precisão dos modelos ajustados

Utilizando o critério de classificação (2), discutido na introdução, a precisão do modelo foi feita da forma:

$$\text{Precisão} = \frac{\text{Casos corretos previstos com o modelo}}{\text{Quantidade total de dados}}$$

Isso posto, a precisão dos modelos tiveram os seguintes resultados:

- **Gradientes descendentes** (tamanho de passo α_k): 87.87%
- **Gradientes descendentes** (tamanho de passo α'_k): 83.30%
- **Newton**: 87.87%

Em outras palavras, os modelos que convergiram conseguiram predizer 87.87% dos tipos dos tumores utilizando apenas o tamanho do raio médio do tumor.

5 Conclusão

Pode-se observar que ambos os métodos convergiram (exceto o de tamanho de passo α'_k) para aproximadamente os mesmos resultados: $\boldsymbol{\theta}^* \approx \begin{pmatrix} -15.24 \\ 1.033 \end{pmatrix}$ e $J(\boldsymbol{\theta}) = 0.289$. Consequentemente, os modelos ajustados tiveram a mesma acurácia de classificação: 87.87% .

Observa-se também uma grande discrepância no número de iterações entre os métodos, uma vez que no de Newton a resposta satisfatória foi obtida com apenas 8 iterações, enquanto no gradiente descendente foram necessárias 200.00.

Esse número grande de operações era esperado, uma vez que esse método frequentemente leva demasiado tempo para convergir.

Referências

[1] https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-ii-d20a239cde11

[2] https://developers.google.com/machine-learning/crash-course/logistic-regression/model-training

[3] https://towardsdatascience.com/binary-cross-entropy-and-logistic-regression-bf7098e75559

[4] https://www.kaggle.com/datasets/uciml/breast-cancer-wisconsin-data