Gradiente Descendente

Alunos: Anabelle Elizabeth Araujo de Souza, Gabriel Luiz dos Santos Silva e Jules Severo Barcos

Prof. Me. Alexandre Garcia de Oliveira

Santos, 08 de Maio de 2023

CONTROLE DE VERSÃO			
Autor	**Versão**	**Data**	**Descrição**
Anabelle Souza	1.0	06/05/2023	Criação do documento
Jules Severo	2.0	06/05/2023	Criação do documento
Gabriel L. S. Silva	3.0	06/05/2023	Criação do documento

Sumário

1 Definição de Gradiente Descendente

Gradiente descendente é um algoritmo de otimização usado em aprendizado de máquina e em outros campos relacionados, como a otimização de funções matemáticas. O objetivo do gradiente descendente é encontrar o mínimo global de uma função de custo, ajustando os valores dos parâmetros de um modelo de acordo com a direção do gradiente descendente, ou seja, a direção em que a função está decrescendo mais rapidamente. O algoritmo funciona iterativamente, atualizando os valores dos parâmetros em pequenos passos, de acordo com a inclinação da curva de custo em relação a cada parâmetro. Esses passos são determinados pelo tamanho do passo (também conhecido como taxa de aprendizado) e pela magnitude do gradiente em cada ponto. O processo continua até que a função de custo atinja um mínimo global ou até que um critério de parada seja atingido. O gradiente descendente é um dos algoritmos mais usados em aprendizado de máquina devido à sua simplicidade e eficácia em muitos tipos de problemas de otimização.

2 Parábola com gradiente descendente

Esse código é um exemplo de como encontrar o mínimo de uma função de três variáveis usando o método do gradiente descendente. Ele utiliza uma função matemática chamada parábola, definida como:

$$f(x, y, z) = (x - 2)^2 + (y + 3)^2 + (z - 1)^2$$

O método do gradiente descendente é um algoritmo de otimização que usa o gradiente da função (derivada parcial em cada dimensão) para encontrar o mínimo local. O algoritmo atualiza iterativamente as variáveis de entrada usando a fórmula:

$$x' = x - \gamma \cdot \nabla f(x)$$

Onde x representa um ponto na função, $\nabla f(x)$ é o gradiente da função em x, γ e é uma constante que determina o tamanho do passo, e x' é o novo ponto na função.

O algoritmo continua atualizando as variáveis de entrada até que a diferença entre o valor da função atual e o valor anterior esteja abaixo de um valor de tolerância pré-determinado.

O programa solicita que o usuário insira um valor inicial para as variáveis x0, y0 e z0, bem como o Learning rate. Em seguida, ele executa o algoritmo do gradiente descendente com essas entradas, imprimindo o ponto de mínimo, o valor mínimo e o número de iterações necessárias para alcançar o mínimo.

```haskell
type R = Double

f :: R -> R -> R -> R
f x y z = (x-2)**2 + (y+3)**2 + (z-1)**2

grad :: R -> R -> R -> (R, R, R)
grad x y z = (2*(x-2), 2*(y+3), 2*(z-1))

descent :: R -> R -> R -> R -> R -> Int -> R -> (R, R, R, Int)
descent lr x y z err i tol
  | err < tol = (x, y, z, i)
  | otherwise = descent lr x' y' z' err' (i+1) tol
  where
    (dx, dy, dz) = grad x y z
    x' = x - lr*dx
    y' = y - lr*dy
    z' = z - lr*dz
    err' = f x' y' z'

main :: IO ()
main = do
  putStr "Digite o valor de x0: "
  x0 <- readLn
  putStr "Digite o valor de y0: "
  y0 <- readLn
  putStr "Digite o valor de z0: "
  z0 <- readLn
  let tol = 1e-16
  putStr "Digite o valor do learning rate: "
  lr <- readLn
  let (xf, yf, zf, n) = descent lr x0 y0 z0 (f x0 y0 z0) 0 tol
  putStrLn $ "Ponto de mínimo: (" ++ show xf ++ ", " ++ show yf ++ ", " ++ show zf ++ ")"
  putStrLn $ "Valor mínimo: " ++ show (f xf yf zf)
  putStrLn $ "Número de iterações: " ++ show n

```

Figura 1: Haskell - Código.

Nos exemplos abaixo, o ponto mínimo de uma função é o ponto em que a função atinge o menor valor possível, ou seja, é o ponto onde a função atinge o seu mínimo absoluto. Já o valor mínimo é o valor numérico que a função atinge no ponto mínimo, ou seja, é o menor valor que a função pode assumir.

```
Digite o valor de x0: 999
Digite o valor de y0: 999
Digite o valor de z0: 197999
Digite o valor do learning rate: 0.1
Ponto de mínimo: (2.0000000000421805, -2.999999999957608, 1.0000000083768166)
Valor mínimo: 7.01746321733726e-17
Número de iterações: 138
```

Figura 2: Teste1.

```
Digite o valor de x0: -1
Digite o valor de y0: 2
Digite o valor de z0: 499
Digite o valor do learning rate: 0.1
Ponto de mínimo: (1.999999999947506, -2.99999999991251, 1.0000000087140068)
Valor mínimo: 7.59443254956859e-17
Número de iterações: 111
```

Figura 3: Teste2.

```
Digite o valor de x0: 44
Digite o valor de y0: 19
Digite o valor de z0: 13
Digite o valor do learning rate: 0.1
Ponto de mínimo: (2.000000008555551, -2.999999995518521, 1.0000000024444433)
Valor mínimo: 9.925641061834706e-17
Número de iterações: 100
```

Figura 4: Teste3.

3 Método dos Mínimos Quadrados

Dado o conjunto de treinamento. Qual é a melhor parábola que aprova o conjunto de treinamento de modo a minimizar o erro?

$$[(1, 4), (-1, 5), (2, 7), (-2, 8), (3, 12), (-3, 13), (4, 19), (-4, 20)]$$

Definição: Este método é utilizado para encontrar a melhor parábola que se ajusta a um conjunto de dados de treinamento e minimiza o erro. Para aplicar o método dos mínimos quadrados, é necessário formar uma matriz de covariância e uma matriz de resposta com base nos pontos do conjunto de treinamento.

Para encontrar a melhor parábola que se ajusta aos pontos do conjunto de treinamento e minimiza o erro, devemos seguir a equação geral de uma parábola, dada por:

$$\hat{y} = w_1 x^2 + w_2 x + b$$

Para encontrar os coeficientes da equação da parábola que melhor se ajusta aos pontos do conjunto de treinamento, a seguinte matriz representa o sistema de equações que deve ser resolvido:

$$\begin{bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i^2 y_i \end{bmatrix}$$

Essa matriz representa o sistema de equações que deve ser resolvido para encontrar os coeficientes da parábola de melhor ajuste para um conjunto de dados de treinamento.

1. **A primeira matriz à esquerda** contém os coeficientes das equações da parábola
2. **A matriz à direita** contém os valores resultantes da combinação linear desses coeficientes com as variáveis independentes.

A solução desse sistema de equações nos dará os valores dos coeficientes w_1, w_2 e b que minimizam a soma dos quadrados dos erros entre os valores preditos pela parábola e os valores observados nos dados de treinamento.

Seguindo os passos, utilizamos o Python para axilar nos cálculos, e logo, temos o seguinte algoritmo.

```python
def parabolic_regression(X, Y):

  # Somas e Processos com X e Y
  n = len(X)
  X_sum = sum(X)
  X2_sum = sum([x**2 for x in X])
  X3_sum = sum([x**3 for x in X])
  X4_sum = sum([x**4 for x in X])

  Y_sum = sum(Y)
  XY_sum = sum([x*y for x, y in zip(X,Y)])
  X2Y_sum = sum([x**2*y for x, y in zip(X,Y)])

  # Matriz de coeficientes
  cov_matrix = pd.DataFrame(np.array([[n, X_sum, X2_sum],
                                      [X_sum, X2_sum, X3_sum],
                                      [X2_sum, X3_sum, X4_sum]]))
  print('Matriz de Coeficientes:')
  display(cov_matrix)

  # Matriz de resposta
  res_matrix = pd.DataFrame(np.array([Y_sum, XY_sum, X2Y_sum]))
  print('Matriz de Resposta:')
  display(res_matrix)

  # Coeficientes da parábola
  coef = np.flip(np.linalg.solve(cov_matrix, res_matrix))
  return coef

parabolic_regression(X, Y)
```

Figura 5: Função de Regressão Parabolica.

Matriz de Coeficientes:

	0	1	2
0	8	0	60
1	0	60	0
2	60	0	708

Matriz de Resp

	0
0	88
1	-10
2	918

```
array([[ 1.        ],
       [-0.16666667],
       [ 3.5       ]])
```

Figura 6: Resultados da Função

Logo, com os resultados obtidos, chegamos a seguinte equação.

$$\begin{bmatrix} 8 & 0 & 60 \\ 0 & 60 & 0 \\ 60 & 0 & 708 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ b \end{bmatrix} = \begin{bmatrix} 88 \\ -10 \\ 918 \end{bmatrix}$$

Resolvendo a matriz em que resulta num sistema de equações lineares, temos que:

$$\begin{cases} 8w_1 + 0w_2 + 60b = 88 \\ 0w_1 + 60w_2 + 0b = -10 \\ 60w_1 + 0w_2 + 708b = 918 \end{cases}$$

Utilizando o numpy com a função **np.linalg.solve()**, a mesma utilizada em nossa função em python, conseguimos resolver o sistema linear de 3 equações, encontrando assim, os valores de w_1, w_2 e b

$$\hat{y} = x^2 + (-0.166\ldots)x + 3.5$$

E realizando a seguinte visualização da parábola e dos pontos do conjunto de treinamento, chegamos a conclusão que encontramos a melhor função.

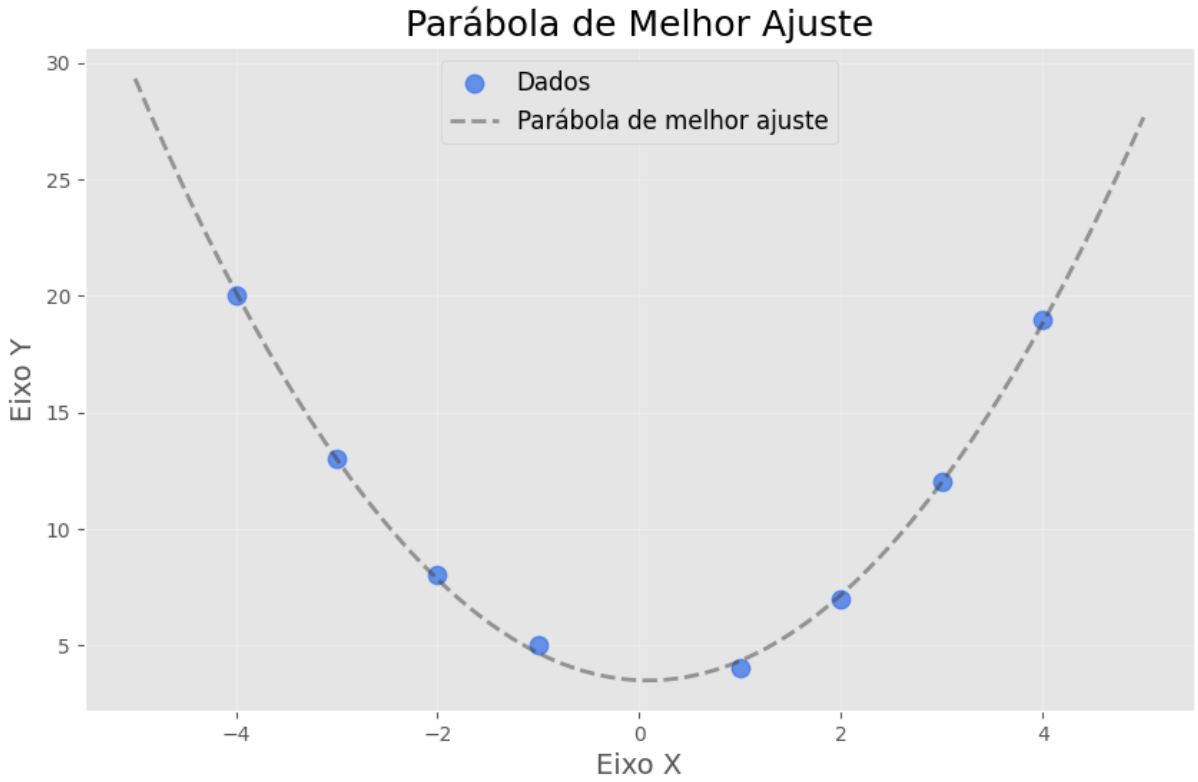

Figura 7: Grafico dos C.T e sua Parabola

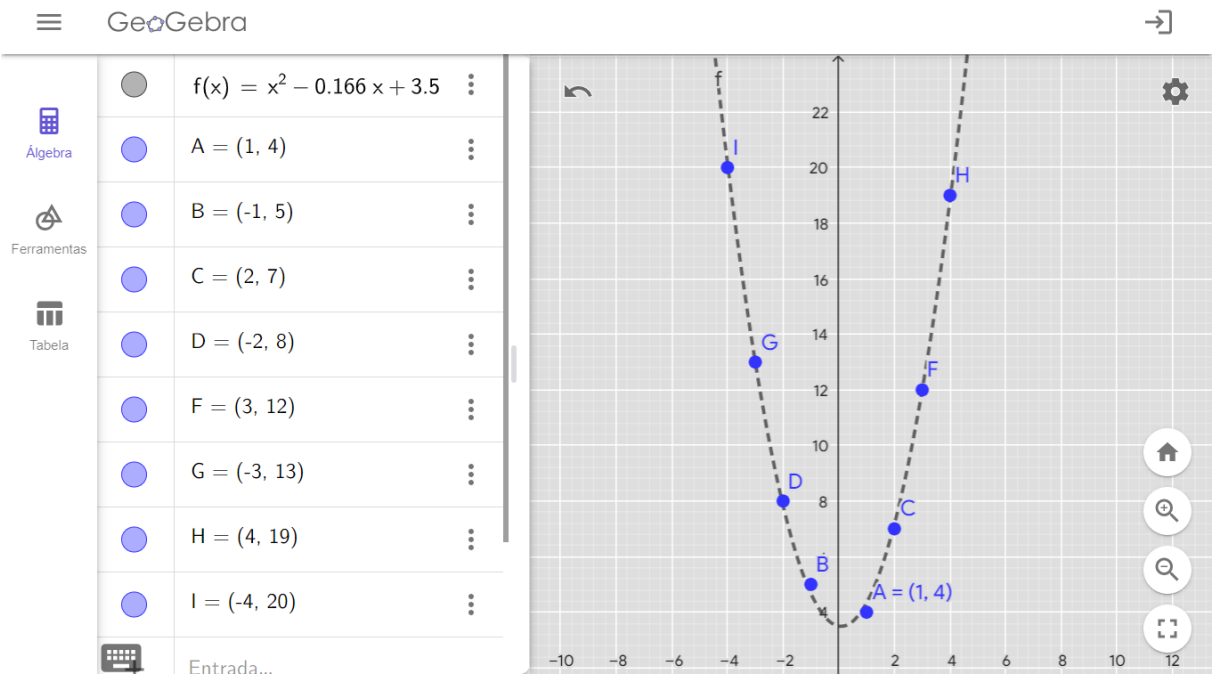

Figura 8: Grafico dos C.T no GeoGebra

4 Github

Link do repositório no Github: AnabelleSouza/Haskell
Link Mínimos Quadrados: gabrielluizone/FirstCode