Cálculo Diferencial e Integral
Lista 4

Guilherme Pereira Amorim
Julianna Lerner Naslauski
Natália Correia Freitas

Novembro 2022

1 Respostas

Exercício 1

$$\begin{array}{rccc} f(x,y) = & x^4 + y^3 = 4x^3y^3 & 3y^2 + x^4 \\ g(x,y) = & x + y = 1 & 1 \end{array}$$

$$\begin{cases} 4x^3\, y^3 = 3y^2\, x^4 \\ 4x^3\, y = 3x^4 \end{cases} \implies \qquad y = \frac{3}{4}x$$

Vamos substituir a equação 3:

$$\begin{cases} 4x^3\, y^3 = \lambda(1) \\ 3y^2\, x^4 = \lambda(2) \\ x + y = \lambda(3) \end{cases}$$

$$x + \frac{3}{4}x - 1 = 0$$

$$x + \frac{3}{4}x = 1$$

$$\frac{x}{1} + \frac{3x}{4} = \frac{4x+3x}{4}$$

$$\frac{4x+3x}{4} = 1$$

$$\frac{7x}{4} = 1$$

$$\frac{7x}{4} = 1$$

$$x = \frac{7}{4}$$

Vamos substituir a equação 1:

$$x + y - 1 = 0$$

$$\frac{3}{7} + y = 1$$

$$y = 1 - \frac{3}{7} \implies \frac{7-4}{7} = \frac{3}{7} = y$$

$$\boldsymbol{f}\left(\frac{4}{7}, \frac{3}{7}\right) = \left(\frac{4}{7}^4 \cdot \frac{3}{7}^3\right) = \frac{4^4 \cdot 3^3}{7^7}$$

Portanto P $= \left(\frac{4}{7}, \frac{3}{7}\right)$ ***é a função no p*** $= \frac{4^4 \cdot 3^3}{7^7}$ ***do ponto crítico maximizado da função.***

Exercício 2

$$\begin{aligned} f(x,y) &= \quad x^2 + y^2 + 3xy - x + y \\ \nabla f(x,y) &= \quad (2x + 3y - 1, 2y + 3x + 1) \\ \nabla f(x_0, y_0) &= \quad (0,0) \\ (2x_0 + 3y_0 - 1, 2y_0 + 3x_0 + 1) &= \quad (0,0) \end{aligned}$$

$$\begin{cases} 2x_0 + 3y_0 - 1 = 0(*3) \implies & 6x_0 + 9y_0 - 3 = 0 \\ 3x_0 + 2y_0 + 1 = 0(-2) \implies & -6x_0 - 4y_0 - 2 = 0 \end{cases}$$

$$\begin{array}{l} +9y_0 - 3 = 0 \\ -4y_0 - 2 = 0 \\ \hline 5y_0 - 5 = 0 \implies y_0 = 1 \end{array}$$

Vamos substituir yo na equação:

$$3x_0 + 2y_0 + 1 = 0 \implies 3x_0 + 2 + 1 = 0 \implies \quad x_0 = \frac{-3}{3} = -1$$

$$H(x,y) = \begin{vmatrix} fxx & fxy \\ fyx & fyy \end{vmatrix} = \begin{vmatrix} 2 & 3 \\ 3 & 2 \end{vmatrix} = 4 - 9 = -5$$

- 5 é menor que 0, então (-1,1) é ponto de sela.

Exercício 3

1º passo: função objetiva

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

2º passo: MLG

$$\nabla f(x,y,z) = \lambda \cdot \nabla g(x,y,z)$$
$$\nabla f(x,y,z) = (2x-2;2y-2;2z-2)$$
$$\nabla g(x,y,z) = (3,1,-1)$$
$$(2x-2;2y-2;2z-2) = \lambda \cdot (3,1,-1)$$

$$\begin{cases} 2x-2 = 3\lambda \implies & x = \dfrac{3\lambda+2}{2} \\ 2y-2 = \lambda \implies & y = \dfrac{\lambda+2}{2} \\ 2z-2 = -\lambda \implies & z = \dfrac{-\lambda+2}{2} \end{cases}$$

3º passo: Substituir $3x + y - z = 1$

$$3\left(\frac{3\lambda+2}{2}\right) + \left(\frac{\lambda+2}{2}\right) - \left(\frac{-\lambda+2}{2}\right) = 1$$
$$\frac{9\lambda+6}{2} + \frac{\lambda+2}{2} - \frac{\lambda+2}{2} = 1 \implies \frac{11\lambda+6}{2} = 1$$
$$11\lambda = -4 \implies \lambda = \frac{-4}{11}$$

4º passo: Substituir o λ nos pontos:

$$x = 3\left(\frac{(-4/11)+2}{2}\right) \quad y = \left(\frac{(-4/11)+2}{2}\right) \quad z = -\left(\frac{(-4/11)+2}{2}\right)$$

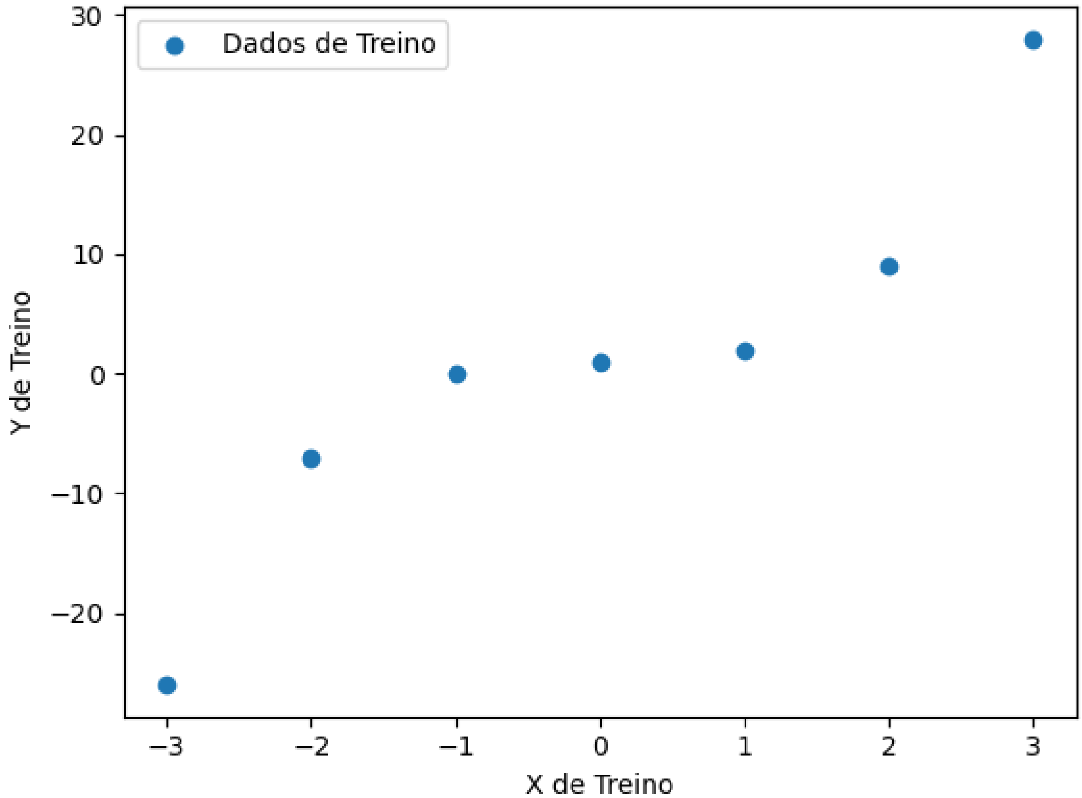
$$\boldsymbol{P} = \left(-\frac{5}{11}; \frac{9}{11}; \frac{13}{11}\right)$$

Lista4_Ex4

November 30, 2022

[1]:
```python
import numpy as np # biblioteca para computação científica
import matplotlib.pyplot as plt # biblioteca para visualização de dados

x_treino = np.array([0, 1, 2, 3, -1, -2, -3]) # Valores de X
y_treino = np.array([1, 2, 9, 28, 0, -7, -26]) # Valores de Y

x_media = np.mean(x_treino) # cálculo da média de x_treino
y_media = np.mean(y_treino)  # cálculo da média de y_treino

a = np.sum(x_treino * (y_treino - y_media)) / np.sum(x_treino * (x_treino -
 ↪x_media)) #coeficiente angular estimado

b = y_media - (a * x_media) # coeficiente linear estimado

funcao = a * x_treino + b # função estimada

R_quadrado = np.sum((funcao-y_media) ** 2) / np.sum((y_treino - y_media) ** 2)
 ↪#coeficiente de determinação

print(f"R² é: {np.round(R_quadrado, 2)}\n")

plt.figure()
plt.scatter(x_treino, y_treino, label = "Dados de Treino")
plt.xlabel("X de Treino")
plt.ylabel("Y de Treino")
plt.legend()
plt.show()
```

```
R² é: 0.86
```


Dados de Treino
Y de Treino
X de Treino
30
20
10
0
−10
−20
−3
−2
−1
0
1
2
3

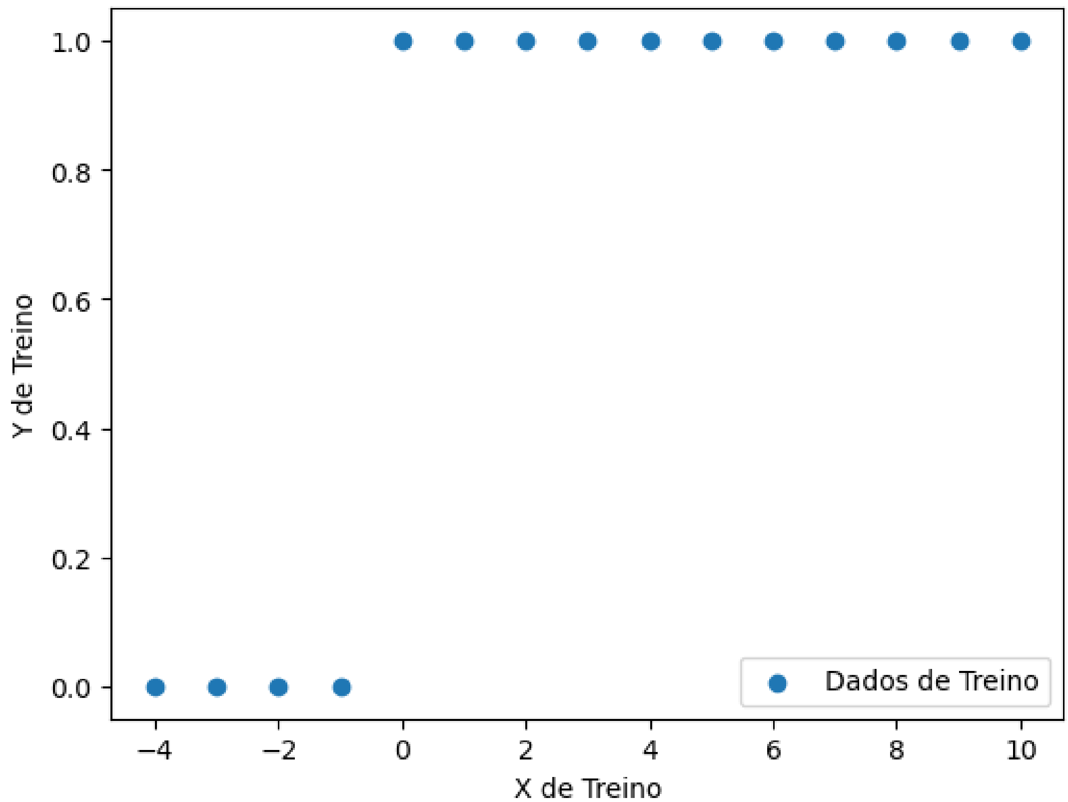
Lista4_Ex5

November 30, 2022

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import math

def sigmoid(x):
  return 1 / (1 + math.exp(-x))

# Conjunto de Treinamento
X = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -1, -2, -3, -4]
Y = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]

data = pd.DataFrame((zip(X, Y)), columns = ["X", "Y"])

X = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, -1, -2, -3, -4])
Y = np.array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0])

    # Função para normalizar os dados de treinamento
def normalizeData(data):
    # Para cada atributo (coluna) do dataframe
    for feature in data.columns:
        # Obtem o maior e menor valor da coluna correspondente
        maxValue = data[feature].max()
        minValue = data[feature].min()

        # Realiza a normalização do dado atual, para que o mesmo sempre
        # corresponda a um valor entre 0 e 1
        data[feature] = (data[feature] - minValue) / (maxValue - minValue)

    return data



data = normalizeData(data)
```

```python
x_media = np.mean(X) # cálculo da média de x_treino
y_media = np.mean(Y)  # cálculo da média de y_treino

a = np.sum(X * (Y - y_media)) / np.sum(X * (X - x_media)) #coeficiente angular
 ↪estimado

b = y_media - (a * x_media) # coeficiente linear estimado

funcao = sigmoid(14) # função estimada

R_quadrado = np.sum((funcao-y_media) ** 2) / np.sum((Y - y_media) ** 2)
 ↪#coeficiente de determinação

print(f"R² é: {np.round(R_quadrado, 2)}\n")
print("y(14) =", funcao)

plt.figure()
plt.scatter(X, Y, label = "Dados de Treino")
plt.xlabel("X de Treino")
plt.ylabel("Y de Treino")
plt.legend()
plt.show()
```

```
R² é: 0.02

y(14) = 0.9999991684719722
```


1.0
0.8
0.6
0.4
0.2
0.0
Y de Treino
Dados de Treino
−4
−2
0
2
4
6
8
10
X de Treino

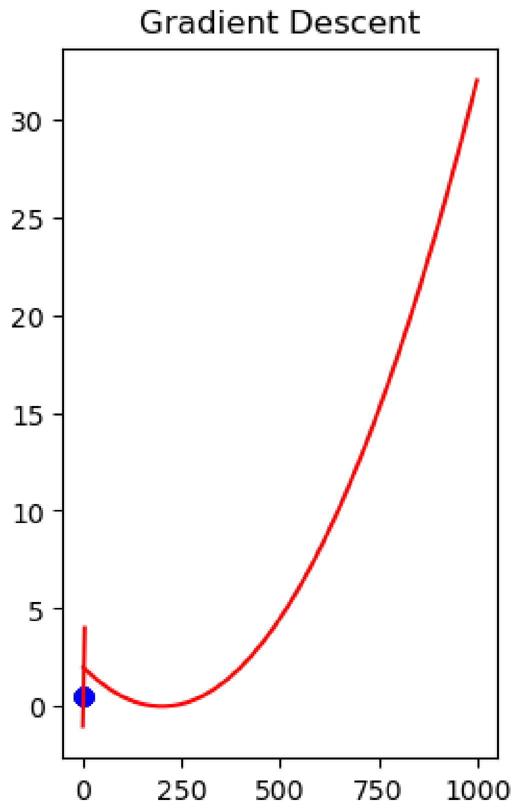
Lista4_Ex6

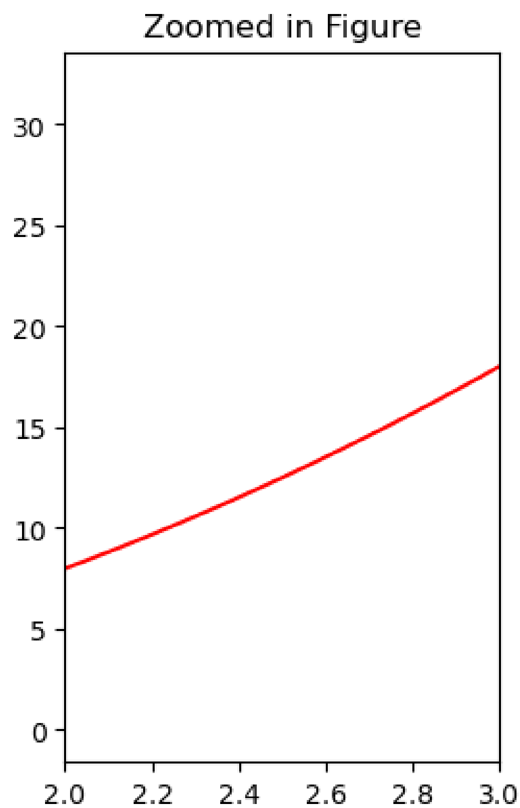
December 1, 2022

[2]:
```python
import numpy as np
import matplotlib.pyplot as plt

f_x = lambda x,y: (x**2)+(y**2)

x = np.linspace(-1,4,1000)
y = x

f_x_derivative = lambda x,y: 2*x+2*y

def plot_gradient(x, y, x_vis, y_vis):
    y = x
    plt.subplot(1,2,2)
    plt.scatter(x_vis, y_vis, c = "b")
    plt.plot(x, y, f_x(x, y), c = "r")
    plt.title("Gradient Descent")
    plt.show()
    plt.subplot(1,2,1)
    plt.scatter(x_vis, y_vis, c = "b")
    plt.plot(x,f_x(x, y), c = "r")
    plt.xlim([2.0,3.0])
    plt.title("Zoomed in Figure")
    plt.show()

def gradient_iterations(x_start, y_start,iterations, learning_rate):

    x_grad = [x_start]
    y_grad = [f_x(x_start, y_start)]
    for i in range(iterations):

        x_start_derivative = - f_x_derivative(x_start, x_start)

        x_start += (learning_rate * x_start_derivative)

        x_grad.append(x_start)
        y_grad.append(f_x(x_start, y_start))
    print ("Local minimum occurs at: {:.2f}".format(x_start, y_start))
```

```
    print ("Number of steps: ",len(x_grad)-1)
    plot_gradient(x, f_x(x, y) ,x_grad, y_grad)

gradient_iterations(0.5, 0.5, 100, 0.0001)
```

```
Local minimum occurs at: 0.48
Number of steps:  100
```




Zoomed in Figure
30
25
20
15
10
5
0
2.0
2.2
2.4
2.6
2.8
3.0

Lista4_Ex7

November 28, 2022

[4]:
```
# Importa as bibliotecas
import numpy as np
import pandas as pd
from pandas_datareader import data as pdr
import yfinance as yf
import matplotlib.pyplot as plt
import riskfolio as rp

yf.pdr_override()
# Busca os preços das ações
## Define as ações
assets = ['ITUB4.SA', 'BBAS3.SA', 'BBDC4.SA', 'BCSA34.SA']

## Define a data início
start = '2021-11-23'
end = '2022-11-25'

# Busca os preços ajustados
prices = pdr.get_data_yahoo(assets, start = start, end = end)['Adj Close']

# Calcula os retornos e retira dados faltantes
returns = prices.pct_change().dropna()

# Cria o objeto de portfolio
port = rp.Portfolio(returns = returns)

method_mu = 'hist'  # define o método de retornos histórico para calcular o␣
 ↪retorno esperado

method_cov = 'hist' # define o método de retornos histórico para calcular a␣
 ↪matriz de covariância

# Calcula os inputs do método de otimização
port.assets_stats(method_mu = method_mu, method_cov = method_cov, d = 0.94)

# Estima o portfolio ótimo
```

```python
model='Classic' # Modelo clássico de Markowitz
rm = 'MV' # Medida de risco: mean-variance
obj = 'MinRisk' # Função objetivo: Minimização de Risco

w = port.optimization(model = model, rm = rm, obj = obj)

print(w * 100)
```

```
[*********************100%***********************]  4 of 4 completed
             weights
BBAS3.SA   24.463381
BBDC4.SA    0.203088
BCSA34.SA  22.240868
ITUB4.SA   53.092663
```