Matemáticas para Ciencias de los Datos

-Profesor: M.Sc.Saul Calderon.

- Estudiantes:
  - Daniel Madriz Granados, Noelia Rojas Ramírez, Steven Jiménez.

Nota: 89

1. Sistemas Lineales

1.1 Demostración Sistemas Lineales

Demuestre si los siguientes sistemasL{x·}(con entradau(t)y salidag(t),yh(t)una función cualquiera) son lineales o no lineales. Además, muestrelo con una implementación en Pytorch, usando como entrada un arre-glo de 50 valores generados al azar. Si va a demostrar por contraejemplo,muestre las entradas y salidas de la corrida en Pytorch que demuestranel no cumplimiento de la propiedad.

Un operador L es lineal si cumple con las propiedades de homogeneidad y superposición, las cuales se pueden resumir en la siguiente ecuación:

$$L\{\alpha u_1(t) + \beta u_2(t)\} = \alpha L\{u_1(t)\} + \beta L\{u_2(t)\}$$

Nota:

- Se le llamará izquierda a $L\{\alpha u_1(t) + \beta u_2(t)\}$
- Se le llamará derecha a $\alpha L\{u_1(t)\} + \beta L\{u_2(t)\}$

a) Demuestra si el siguiente sistema es lineal:

$$g(t) = u(t) + 7$$

Izquierda:

$$\{\alpha u_1(t) + \beta u_2(t)\} + 7 =$$
$$\alpha u_1(t) + \beta u_2(t) + 7$$

Derecha:

$$\alpha(u_1(t) + 7) + \beta(u_2(t) + 7) =$$
$$\alpha u_1(t) + 7\alpha + \beta u_2(t) + 7\beta$$

Son diferentes, no se cumple con las propiedades antes descritas, se procede a mostrarlo a través de un contraejemplo (expresado functionA(x)).

b) Demuestra si el siguiente sistema es lineal:

$$g(t) = u(t)h(t)$$

Izquierda:

$$\{\alpha u_1(t) + \beta u_2(t)\}h(t) =$$
$$\alpha h(t)u_1(t) + \beta h(t)u_2(t)$$

Derecha:

$$\alpha h(t)u_1(t) + \beta h(t)u_2(t)$$

Cumple con las propiedades lineales $\therefore$ Es linea. Se adjunta un ejemplo en Pytorch a través de functionB(x)).

c) Demuestra si el siguiente sistema es lineal:

$$g(t) = max(u(t))$$

Izquierda:

$$max(\alpha u_1(t) + \beta u_2(t))$$

Derecha:

$$\alpha max(u_1(t)) + \beta max(u_2(t))$$

Son diferentes, no se cumple con las propiedades antes descritas, se procede a mostrarlo a través de un contraejemplo (expresado functionC(x)).

d) Demuestra si el siguiente sistema es lineal:

$$g(t) = \sum u(i)$$

Izquierda

$$\sum_{i=1}^{t}(\alpha u_1(i) + \beta u_2(i)) =$$

$$\alpha \sum_{i=1}^{t} u_1(i) + \beta \sum_{i=1}^{t} u_2(i)$$

Derecha

$$\alpha \sum_{i=1}^{t} u_1(i) + \beta \sum_{i=1}^{t} u_2(i)$$

Cumple con las propiedades lineales

$\therefore$ Es linea. Se adjunta un ejemplo en Pytorch a través de functionD(x)).

e) Demuestra si el siguiente sistema es lineal:

$$g(t) = |u(t)|$$

Izquierda

$$|\alpha u_1(t)| + |\beta u_2(t)| =$$

$$|\alpha||u_1(t)| + |\beta||u_2(t)|$$

Derecha

$$\alpha|u_1(t)| + \beta|u_2(t)|$$

Son diferentes, no se cumple con las propiedades antes descritas, se procede a mostrarlo a través de un contraejemplo (expresado functionE(x)).

```
import torch

def functionA(x):
  y = x + 7;
  return y;

def functionB(x, y = 1):
```

```
  y = x * (y * 2);
  return y;

def functionC(x):
  y = torch.max(x);
  return y;

def functionD(x):
  y = torch.sum(x);
  return y;

def functionE(x):
  y = torch.abs(x);
  return y;

def checkLinearity(operator):
  randomArray = torch.randn(50) # Creation of the vector with 50 random numbers
  print("Arreglo creado con valores aleatorios: ", randomArray.data.cpu().numpy())

  # Sample scalars
  alpha = torch.randn(1,1)
  beta = torch.randn(1,1)

  # Sample functions
  f1 = lambda x: x + 2
  f2=  lambda x: x * x

  # Compute left and right side of the equation
  left = operator(alpha * f1(randomArray) + beta * f2(randomArray))
  right = alpha * operator(f1(randomArray)) + beta * operator(f2(randomArray))

  difference = torch.abs(right - left);
  sumDifference = difference.sum()

  print ("Diferencia entre el lado izquierdo y el derecho: ", sumDifference.data.cpu().numpy())
  # Is linear?
  # We use .data.cpu().numpy() to get the actual object value
  return (sumDifference < 0.0001).data.cpu().numpy() # We compare with less than 0.0001 as computed results may consider "round"

print("Es el sistema representado por functionA lineal? ", checkLinearity(functionA))
print("******************************************************************************")
print("******************************************************************************")
print("Es el sistema representado por functionB lineal? ", checkLinearity(functionB))
print("******************************************************************************")
print("******************************************************************************")
print("Es el sistema representado por functionC lineal? ", checkLinearity(functionC))
print("******************************************************************************")
print("******************************************************************************")
print("Es el sistema representado por functionD lineal? ", checkLinearity(functionD))
print("******************************************************************************")
print("******************************************************************************")
print("Es el sistema representado por functionE lineal? ", checkLinearity(functionE))
```

```
     Arreglo creado con valores aleatorios:  [-1.0872052   0.46310717 -0.5277105  -0.7152138   1.3534558   0.72213995
      -0.90398055 -1.2796117   1.0695658  -0.5587702   0.7912219   0.09167498
      -0.3692724  -0.12817708  0.8442319   0.29147056 -0.6038284  -0.508965
      -0.32411152  1.683339    1.8080183  -0.5494128   1.5138677  -0.8697871
```

```
  0.1027972   0.648396   -1.254196    0.04437498  0.37699902 -0.89238894
  1.4208422   1.4014034   0.7356373   0.32883334  0.29643905  2.1160989
  0.17884605 -0.29284433 -0.86429113 -1.0360291   0.56238115 -0.877877
 -0.6685695  -0.26160273  0.45353222  0.9233992  -0.15729183 -1.3823491
 -1.312184    1.8096968 ]
Diferencia entre el lado izquierdo y el derecho:  66.51644
Es el sistema representado por functionA lineal?  False
********************************************************************************
********************************************************************************
Arreglo creado con valores aleatorios:  [ 6.1142194e-01  8.7807244e-01 -2.2038130e-01 -9.0495610e-01
  9.1561002e-01  6.1842430e-01 -1.8531455e+00 -1.0424274e+00
 -9.2629683e-01  3.0307448e-01 -4.8289114e-01  1.9101125e+00
  9.8261374e-01  4.5821837e-01  1.1941226e+00 -6.1051232e-01
 -1.3273082e+00  6.6701615e-01 -2.9346099e-01  9.4589293e-01
 -5.2438194e-01 -1.7707424e-01 -9.9918830e-01 -1.7092322e+00
 -4.7974041e-01 -1.0562003e+00 -7.9531111e-03 -8.3175117e-01
  1.8200359e+00  1.5543154e+00 -1.4834989e+00  1.4997025e+00
  1.9101223e+00  1.1056085e-03 -1.4335161e+00  1.0702033e-01
  1.2802147e+00  4.4507596e-01 -1.0988294e+00  6.2877708e-03
  1.5770802e+00  1.3839130e-01 -1.6001310e+00  5.1290005e-01
  3.2814702e-01  3.4363762e-01  1.4191319e+00 -1.2187649e-01
 -8.0624694e-01 -8.3028156e-01]
Diferencia entre el lado izquierdo y el derecho:  0.0
Es el sistema representado por functionB lineal?  True
********************************************************************************
********************************************************************************
Arreglo creado con valores aleatorios:  [-1.8580798  -0.17072627 -0.15305458 -0.84778374  0.08044115  1.1116965
 -0.8422217   1.5301759   0.36997405  0.578725   -1.6979996   2.4114602
 -0.29688966 -0.42867264 -0.4734125   1.2947494   0.5000272  -1.1840674
  0.75342464  0.46515045 -0.03532398 -1.067158    0.8559349   1.0311605
 -0.84729564  0.77643085 -0.531099   -0.78467447 -1.0203689  -0.7886994
 -0.57446903  0.11065058 -0.9128803   0.9839004   1.0609312   0.88411933
 -0.32874277 -1.0998181  -1.1303011   0.99734384 -1.7940208  -0.797823
 -0.33508438  0.87481207 -1.7314018   0.16208404 -1.1511633   0.21374875
  0.8600797   0.40984705]
Diferencia entre el lado izquierdo y el derecho:  2.515665
Es el sistema representado por functionC lineal?  False
********************************************************************************
********************************************************************************
Arreglo creado con valores aleatorios:  [-0.8019426  -0.01979659 -0.56703824 -0.09442317 -0.0463709  -0.7489956
 -0.14343998  0.41138944 -0.6539377   0.8987014  -1.5791702  -1.1156611
  0.86898607  0.89286715  1.3717375  -0.37332284  0.06658975  0.20103419
 -0.6331365   1.1546828   1.5613773  -0.26151925  0.52366257  1.7273475
 -1.0760318   1.4829243  -0.66054803 -0.48411283  1.0047699  -0.4394836
  0.11793954  0.22193083 -1.069817   -2.50652     0.2826057  -0.72826046
 -1.1030939   0.03345194 -0.73712146  0.12929955  1.3405935  -0.3642312
 -0.2490339  -0.7729519  -0.07100192 -0.8027343   0.70216155 -0.5123718
 -1.6718476   0.6715004 ]
Diferencia entre el lado izquierdo y el derecho:  5.722046e-06
Es el sistema representado por functionD lineal?  True
********************************************************************************
********************************************************************************
Arreglo creado con valores aleatorios:  [ 0.5861437  -0.8596702  -0.91870165  1.4324187   0.12189835  0.94417787
 -0.49790865 -0.00626099  0.6023196   0.54414254  1.0827678   0.38243145
 -0.1466911   0.60973567  1.4331726  -0.00733385  0.31446916 -1.120392
```

1.2 Demostración Homogeneidad y Superposición

Demuestre si los siguientes sistemasL{x·} (con entradau(t)y salidag(t),yh(t)una función cualquiera) son lineales o no lineales. Además, muestrelo con una implementación en Pytorch, usando como entrada un arre-glo de 50 valores generados al azar. Si va a demostrar por contraejemplo,muestre las entradas y salidas de la corrida en Pytorch que demuestranel no cumplimiento de la propiedad.

Nota Para los siguientes ejercicios se considera lo siguiente en la propiedad de superpocisión:

$$L\{f_1(x)+f_2(x)\}=L\{f_1(x)\}+L\{f_2(x)\}$$

- Se le llamará izquierda a $L\{f_1(x)+f_2(x)\}$
- Se le llamará derecha a $L\{f_1(x)\}+L\{f_2(x)\}$

a) Norma de Manhattan l_1

Propiedad de homogeneidad absoluta

Tenemos que:

p= 1

$x=(x_1,x_2,x_3,...,x_n)$ con x $\in\mathbb{R}^n$

$||\vec{x}||_1=(|x_1|^1+|x_2|^1+|x_3|^1+...+|x_n|^1)^{1/1}$

Ahora con $f(t\vec{x})=|t|f(\vec{x})$

$||\vec{x}||_1=(|tx_1|^1+|tx_2|^1+|tx_3|^1+...+|tx_n|^1)^{1/1}$

$=|tx_1|^1+|tx_2|^1+|tx_3|^1+...+|tx_n|^1$

$=|t|(|x_1|+|x_2|+|x_3|+...+|x_n|)$

$=|t|\cdot||\vec{x}||_1$

Cumple con la propiedad

Propiedad de superpocisión

$L\{f(x)\}=(\sum_{i=1}^{n}|x_i|^1)^{1/1}=\sum_{i=1}^{n}|x_i|$

Izquierda

$\sum_{i=1}^{n}|x_{1i}+x_{2i}|$

Derecha

$\sum_{i=1}^{n}|x_{1i}|+\sum_{i=1}^{n}|x_{2i}|$

$Izquierda\neq Derecha$

No cumple con la propiedad

Contraejemplo

Propiedad de superposición

$\vec{x}_1=(-3,2)$

$\vec{x}_2=(5,2)$

Izquierda

|-3 +5| + |2+2|= |2| + |4| = |2| + |4| = 6

Derecha

|-3| + |2| + |5| + |5| + |2| = 12

```
import math

# Propiedad homogeneidad

torch.manual_seed(0)

def op1(x):
  y = torch.sum(torch.abs(x));
  return y;

def homogeneidad(operator):
  x0= torch.zeros(50)
  x= torch.randint(low=-20, high=80, size=(50,15))

  #compute left and right side of the equation
  t=3
  for i in range(50):
    right= operator(t*x[i])
    left= abs(t)* operator(x[i])
    difference = torch.abs(right - left);
    x0[i]=difference

  sumDifference = x0.sum()
  #Probar la propiedad
  return (sumDifference < 0.001)

# Verificación
print("Propiedad Homogeneidad", homogeneidad(op1).data.cpu().numpy())

# Propiedad de superposición
def superposicion(operator):
  x0= torch.zeros(50)
  x= torch.randint(low=-20, high=80, size=(50,15))
  y= torch.randint(low=-30, high=70, size=(50,15))

  #compute left and right side of the equation

  for i in range(50):
    right= operator(x[i]) + operator(y[i])
    left= operator(x[i] + y[i])
    difference = torch.abs(right - left);
    x0[i]=difference

  sumDifference = x0.sum()
  #Probar la propiedad
  return (sumDifference < 0.001)

print("Propiedad Superposicion", superposicion(op1).data.cpu().numpy())
```

```
Propiedad Homogeneidad True
Propiedad Superposicion False
```

b) Norma Euclidiana l_3

Propiedad de homogeneidad absoluta

Se tiene que:

p=3

$x = (x_1, x_2, x_3, \ldots, x_n)$ con x $\in \mathbb{R}^n$

$||\vec{x}||_3 = \sqrt[3]{|x_1|^3 + |x_2|^3 + |x_3|^3 + \ldots + |x_n|^3}$

Ahora con $f(t\vec{x}) = |t| f(\vec{x})$

$||t\vec{x}||_3 =$

$\sqrt[3]{|tx_1|^3 + |tx_2|^3 + |tx_3|^3 + \ldots + |tx_n|^3} =$

$\sqrt[3]{|t|^3 \cdot |x_1|^3 + |t|^3 \cdot |x_2|^3 + |t|^3 \cdot |x_3|^3 + \ldots + |t|^3 \cdot |x_n|^3} =$

$= |t| \sqrt[3]{|x_1|^3 + |x_2|^3 + |x_3|^3 + \ldots + |x_n|^3}$

$= |t| (|x_1|^3 + |x_2|^3 + |x_3|^3 + \ldots + |x_n|^3)^{1/3}$

$||t\vec{x}|| = |t| \cdot ||\vec{x}||_3$

Cumple la propiedad

Propiedad de superposición

$L\{f(x)\} = \sqrt[3]{\sum_{i=1}^{n} |x_i|^3}$

Izquierda

$\sqrt[3]{\sum_{i=1}^{n} |x_{1i} + x_{2i}|^3}$

Derecha $\sqrt[3]{\sum_{i=1}^{n} |x_{1i}|^3} + \sqrt[3]{\sum_{i=1}^{n} |x_{2i}|^3}$

$Izquierda \neq Derecha$

No se cumple la propiedad

Contraejemplo

Propiedad de superposición

$\vec{x}_1 = (1, 2)$

$\vec{x}_2 = (3, 4)$

$L\{f(x)\} = \sqrt[3]{\sum_{i=1}^{n} |x_i|^3}$

Izquierda

$\sqrt[3]{|1+3|^3 + |2+4|^3} = 6.54$

Derecha

$\sqrt[3]{|1|^3 + |2|^3} + \sqrt[3]{|3|^3 + |4|^3} = 6.58$

```
def op2(x):
  y= torch.pow(torch.sum(torch.pow(torch.abs(x),3)),1/3);
  return(y);

torch.manual_seed(10)

# Verificación
print("Propiedad Homogeneidad", homogeneidad(op2).data.cpu().numpy())
print("Propiedad superposicion", superposicion(op2).data.cpu().numpy())
```

```
Propiedad Homogeneidad True
Propiedad superposicion False
```

c) Norma l_{00}

Propiedad de homogeneidad absoluta

Se tiene que:

p= ∞

$x = (x_1, x_2, x_3, \ldots, x_n)$ con x $\in \mathbb{R}^n$

$||\vec{x}||_\infty = \sqrt[\infty]{|x_1|^\infty + |x_2|^\infty + \ldots + |x_n|^\infty} = |x_m| = máx(|x_i|)$

Ahora con $f(t\vec{x}) = |t| f(\vec{x})$

$máx(|t \cdot x_i|) =$

$|t| \cdot máx(|x_i|) =$

$|t| \cdot ||\vec{x}||_\infty$

Cumple con la propiedad

Propiedad de Superposición

$L\{f(x)\} = máx(|x_i|)$

Izquierda

$máx(|x_{1i} + x_{2i}|)$

Derecha

$máx(|x_{1i}|) + máx(|x_{2i}|)$

$Izquierda \neq Derecha$

No se cumple la propiedad

Contraejemplo

$\vec{x}_1 = (1, -8)$

$\vec{x}_2 = (2, 8)$

Izquierda

$máx(|1+2|, |-8+8|) =$

$máx(3, 0) = 3$

Derecha

$máx(|1|, |-8|) + máx(|2|, |8|) =$

$8 + 8 = 16$

```
def op3(x):
  y = torch.max(torch.abs(x));
  return y;

torch.manual_seed(25)

# Verificación
print("Propiedad Homogeneidad", homogeneidad(op3).data.cpu().numpy())
print("Propiedad superposicion", superposicion(op3).data.cpu().numpy())
```

```
Propiedad Homogeneidad True
Propiedad superposicion False
```

2. Vectores

2.1 Graficación y propiedades de los vectores

2.1.a) Usando Python grafique los siguientes vectores

1. Graficación y propiedades de los vectores

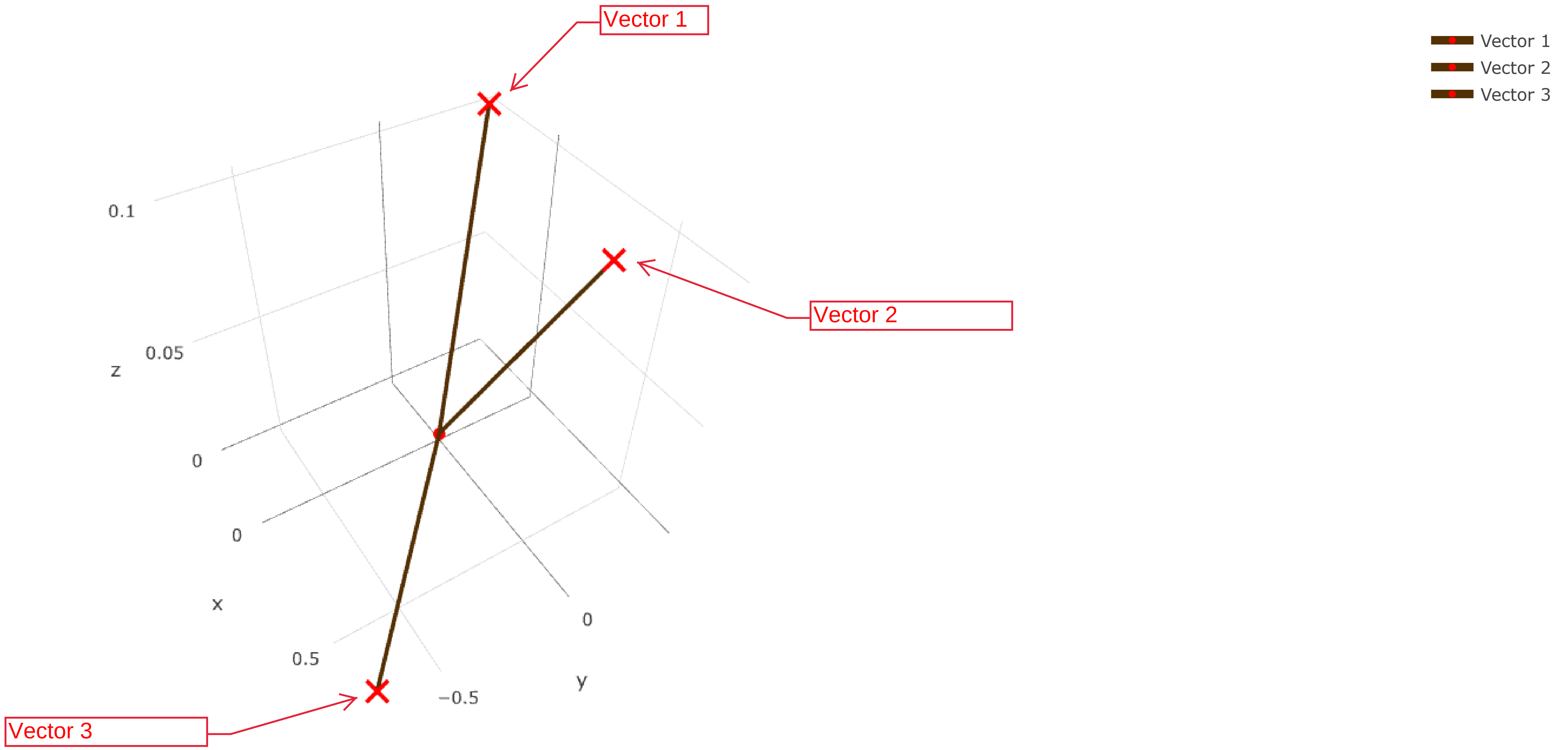
a) Usando Python grafique los siguientes vectores $\vec{v}_1 = \begin{bmatrix} -0.3 \\ 0.4 \\ 0.1 \end{bmatrix}$, $\vec{v}_2 = \begin{bmatrix} 0.5 \\ 0.2 \\ 0.1 \end{bmatrix}$ y $\vec{v}_3 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} \\ 0 \end{bmatrix}$

```
import datetime
from datetime import date
import pandas as pd
import numpy as np
from plotly import __version__
```

```
%matplotlib inline

import plotly.offline as pyo
import plotly.graph_objs as go
from plotly.offline import iplot

import cufflinks as cf
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot


cf.go_offline()

init_notebook_mode(connected=False)

def configure_plotly_browser_state():
  import IPython
  display(IPython.core.display.HTML('''
        <script src="/static/components/requirejs/require.js"></script>
        <script>
          requirejs.config({
            paths: {
              base: '/static/base',
              plotly: 'https://cdn.plot.ly/plotly-1.5.1.min.js?noext',
            },
          });
        </script>
        '''))
```

```
import math
configure_plotly_browser_state()
#prepare plotting points

#Prepare centroid vector
def buildVector(X,Y,Z,name):
  vector = go.Scatter3d(name = name, x = [0,X], y = [0,Y], z = [0,Z],
                      marker = dict(size = 5, color = "rgb(255,0,0)", symbol = ["circle","x"]),
                      line = dict( color = "rgb(84,48,5)", width = 6)
                     )
  return vector;

vector1 = [-0.3, 0.4, 0.1]
vector1Plot = buildVector(vector1[0],vector1[1],vector1[2],"Vector 1")

vector2 = [0.5, 0.2, 0.1]
vector2Plot = buildVector(vector2[0],vector2[1],vector2[2],"Vector 2")

vector3 = [(1/math.sqrt(2)),-(1/math.sqrt(2)),0]
vector3Plot = buildVector(vector3[0],vector3[1],vector3[2],"Vector 3")


data = [vector1Plot,vector2Plot,vector3Plot]
layout = go.Layout(margin = dict( l = 0, r = 0, b = 0, t = 0))
```

```
layout = go.Layout(margin = dict( l = 0, r = 0, b = 0, t = 0))

fig = go.Figure(data=data,layout=layout)
fig.show()
```



2.1.b) Demuestre cuáles de los vectores anteriores son unitarios

```
import numpy as np

def NormaL2(vector):
  vectorTensor = torch.FloatTensor(vector)
  vectorTensor = torch.pow(vectorTensor, 2)
  sum = torch.sum(vectorTensor)
  normaL2 = math.sqrt(sum)
  return normaL2;

def vectorUnitario(norma):
  if norma < 1.001 and norma > 0.999 :
    return True;
  else:
    return False;


print("Norma del vector 1:", NormaL2(vector1))
print("Es el vector 1 un vector unitario?:", vectorUnitario(NormaL2(vector1)))
print("*******************************************")
print("Norma del vector 2:", NormaL2(vector2))
```

```
print("Es el vector 2 un vector unitario?:", vectorUnitario(NormaL2(vector2)))
print("******************************************")
print("Norma del vector 3:", NormaL2(vector3))
print("Es el vector 3 un vector unitario?:", vectorUnitario(NormaL2(vector3)))
```

```
Norma del vector 1: 0.5099019420077323
Es el vector 1 un vector unitario?: False
*******************************************
Norma del vector 2: 0.5477225411817617
Es el vector 2 un vector unitario?: False
*******************************************
Norma del vector 3: 0.9999999701976772
Es el vector 3 un vector unitario?: True
```

2.1.c) Calcule el ángulo en grados, entre los vectores v1 y v2, v2 y v3 y v1 y v3, implementando la fórmula en Pytorch, sin usar las funcionescorrespondientes de la biblioteca

```
def dotProduct(vectorA , vectorB):
  if len(vectorA) == len(vectorB):
    vectorTensorA = torch.FloatTensor(vectorA)
    vectorTensorB = torch.FloatTensor(vectorB)
    dotProductvar = torch.sum(torch.mul(vectorTensorA,vectorTensorB))
    return dotProductvar;
  else:
    print("La dimension de los vectores no coinciden")

def AnguloEntreVector(vectorA, vectorB):
  anguloRad = math.acos(dotProduct(vectorA, vectorB)/(NormaL2(vectorA)*NormaL2(vectorB)))
  #print(anguloRad)
  anguloGrados = math.degrees(anguloRad)
  return anguloGrados;

print("Angulo entre vector1 y vector2",AnguloEntreVector(vector1,vector2),"grados")
print("Angulo entre vector2 y vector3",AnguloEntreVector(vector2,vector3),"grados")
print("Angulo entre vector3 y vector1",AnguloEntreVector(vector3,vector1),"grados")
```

```
Angulo entre vector1 y vector2 102.4058168815314 grados
Angulo entre vector2 y vector3 67.21350132021153 grados
Angulo entre vector3 y vector1 166.1021313427507 grados
```

2.1.d) Calcule la distancia en $\ell_1,\ \ell_2,\ y\,\ell_\infty$ entre los vectores v1 y v2, v2 y v3 y v1 y v3, implementando la fórmula en en Pytorch, sin usar lasfunciones correspondientes de la biblioteca. Compare el resultadoobtenido con el uso de la funcióntorch.norm.

```
def NormaManhattanL1(vector):
  vectorTensor = torch.FloatTensor(vector)
  normaL1 = torch.sum(torch.abs(vectorTensor))
  return normaL1;

def NormaInfinito(vector):
```

```
  vectorTensor = torch.FloatTensor(vector)
  normaInf = torch.max(torch.abs(vectorTensor))
  return normaInf;



print("Implementacion paso a paso de las normas")
print("Norma Manhattan")
print("Norma Manhattan o L1 para el vector1 =",NormaManhattanL1(vector1).data.cpu().numpy())
print("Norma Manhattan o L1 para el vector2 =",NormaManhattanL1(vector2).data.cpu().numpy())
print("Norma Manhattan o L1 para el vector3 =",NormaManhattanL1(vector3).data.cpu().numpy())
print("Norma Euclideana")
print("Norma Euclideana o L2 para el vector1 =",NormaL2(vector1))
print("Norma Euclideana o L2 para el vector2 =",NormaL2(vector2))
print("Norma Euclideana o L2 para el vector3 =",NormaL2(vector3))
print("Norma Infinito")
print("Norma Infinito para el vector1 =",NormaInfinito(vector1).data.cpu().numpy())
print("Norma Infinito para el vector2 =",NormaInfinito(vector2).data.cpu().numpy())
print("Norma Infinito para el vector3 =",NormaInfinito(vector3).data.cpu().numpy())

print("*******************************************************************")

print("Implementation de las normas utilizando pytorch.norm")
Tensor1 = torch.tensor(vector1)
Tensor2 = torch.tensor(vector2)
Tensor3 = torch.tensor(vector3)
print("Norma Manhattan")
print("Norma Manhattan o L1 para el vector1 =",torch.norm(Tensor1, p=1).data.cpu().numpy())
print("Norma Manhattan o L1 para el vector2 =",torch.norm(Tensor2, p=1).data.cpu().numpy())
print("Norma Manhattan o L1 para el vector3 =",torch.norm(Tensor3, p=1).data.cpu().numpy())
print("Norma Euclideana")
print("Norma Euclideana o L2 para el vector1 =",torch.norm(Tensor1, p=2).data.cpu().numpy())
print("Norma Euclideana o L2 para el vector2 =",torch.norm(Tensor2, p=2).data.cpu().numpy())
print("Norma Euclideana o L2 para el vector3 =",torch.norm(Tensor3, p=2).data.cpu().numpy())
print("Norma Infinito")
print("Norma Infinito para el vector1 =",torch.norm(Tensor1, p=float("inf")).data.cpu().numpy())
print("Norma Infinito para el vector2 =",torch.norm(Tensor2, p=float("inf")).data.cpu().numpy())
print("Norma Infinito para el vector3 =",torch.norm(Tensor3, p=float("inf")).data.cpu().numpy())
```

```
Implementacion paso a paso de las normas
Norma Manhattan
Norma Manhattan o L1 para el vector1 = 0.8000001
Norma Manhattan o L1 para el vector2 = 0.8
Norma Manhattan o L1 para el vector3 = 1.4142135
Norma Euclideana
Norma Euclideana o L2 para el vector1 = 0.5099019420077323
Norma Euclideana o L2 para el vector2 = 0.5477225411817617
Norma Euclideana o L2 para el vector3 = 0.9999999701976772
Norma Infinito
Norma Infinito para el vector1 = 0.4
Norma Infinito para el vector2 = 0.5
Norma Infinito para el vector3 = 0.70710677
*******************************************************************
Implementation de las normas utilizando pytorch.norm
Norma Manhattan
Norma Manhattan o L1 para el vector1 = 0.8000001
Norma Manhattan o L1 para el vector2 = 0.8
Norma Manhattan o L1 para el vector3 = 1.4142135
Norma Euclideana
```

```
Norma Euclideana o L2 para el vector1 = 0.50990194
Norma Euclideana o L2 para el vector2 = 0.5477225
Norma Euclideana o L2 para el vector3 = 0.99999994
Norma Infinito
Norma Infinito para el vector1 = 0.4
Norma Infinito para el vector2 = 0.5
Norma Infinito para el vector3 = 0.70710677
```

2.2 Propiedades de los vectores

Propiedades del producto punto: demuestre lo siguiente. Además, muéstrelo con una implementación en Pytorch, usando como entrada un arre-glo de 50 arreglos generados al azar, adjunte un pantallazo con la salida de la comparación del resultado a ambos lados de la igualdad, o en su defecto, demuestre el no cumplimiento de la propiedad con un contra-ejemplo.

a) Bilinearidad del producto punto $\vec{u} \cdot (r\vec{v} + \vec{w}) = r(\vec{u} \cdot \vec{v} + \vec{u} \cdot \vec{w})$

Prueba de que la hipótesis es falsa

Dado:

$$\vec{u} = \begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix} \vec{v} = \begin{bmatrix} 4 \\ -5 \\ 6 \end{bmatrix} \vec{w} = \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} r = 2$$

Izquierda:

$$\begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix} \cdot \left(2 \cdot \begin{bmatrix} 4 \\ -5 \\ 6 \end{bmatrix} + \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \right)$$
$$\begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix} \cdot \left(\begin{bmatrix} 8 \\ -10 \\ 12 \end{bmatrix} + \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \right)$$
$$\begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix} \cdot \begin{bmatrix} 15 \\ -2 \\ 21 \end{bmatrix}$$
$$-1 \cdot 15 + 2 \cdot -2 + -3 \cdot 21$$
$$-82$$

Derecha:

$$2 \cdot \left(\begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix} \cdot \begin{bmatrix} 4 \\ -5 \\ 6 \end{bmatrix} + \begin{bmatrix} -1 \\ 2 \\ -3 \end{bmatrix} \cdot \begin{bmatrix} 7 \\ 8 \\ 9 \end{bmatrix} \right)$$
$$2 \cdot ((-1 \cdot 4 + 2 \cdot -5 + -3 \cdot 6) + (-1 \cdot 7 + 2 \cdot 8 + -3 \cdot 9))$$
$$2 \cdot (-4 - 10 - 18 - 7 + 16 - 27)$$
$$2 \cdot -50$$

$$-100$$

Es decir:

$$-82 \neq -100$$
$$\therefore \text{La hipótesis es falsa}$$

```
def checkBilinearityDoProduct():
  # r scalar
  r = 2

  # Sample vectors
  u = torch.tensor([-1, 2, -3])
  v = torch.tensor([4, -5, 6])
  w = torch.tensor([7, 8, 9])

  # Compute left and right side of the equation
  left = torch.dot(u, (r * v + w))
  right = r * (torch.dot(u, v) + torch.dot(u, w))

  print ("Lado izquierdo de la ecuación:", left.data.cpu().numpy())
  print ("Lado derecho de la equción:", right.data.cpu().numpy())

  return ((left == right)).data.cpu().numpy()

print("Es el resultado de ambos lados el mismo?", checkBilinearityDoProduct())
```

```
Lado izquierdo de la ecuación: -82
Lado derecho de la equción: -100
Es el resultado de ambos lados el mismo? False
```

b) No asociatividad del producto punto $\vec{u} \cdot (\vec{v} \cdot \vec{w}) \neq (\vec{u} \cdot \vec{v}) \cdot \vec{w}$

Prueba de que la hipótesis es falsa

Dado:

$$\vec{u} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \vec{v} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \vec{w} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

Izquierda:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \left(\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right)$$
$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \cdot (1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1)$$

$$\begin{bmatrix}1\\1\\1\end{bmatrix} \cdot (1+1+1)$$

$$\begin{bmatrix}1\\1\\1\end{bmatrix} \cdot 3$$

$$\begin{bmatrix}3\\3\\3\end{bmatrix}$$

Derecha:

$$\left(\begin{bmatrix}1\\1\\1\end{bmatrix} \cdot \begin{bmatrix}1\\1\\1\end{bmatrix}\right) \cdot \begin{bmatrix}1\\1\\1\end{bmatrix}$$

$$(1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1) \cdot \begin{bmatrix}1\\1\\1\end{bmatrix}$$

$$(1+1+1) \cdot \begin{bmatrix}1\\1\\1\end{bmatrix}$$

$$3 \cdot \begin{bmatrix}1\\1\\1\end{bmatrix}$$

$$\begin{bmatrix}3\\3\\3\end{bmatrix}$$

Es decir:

$$\begin{bmatrix}3\\3\\3\end{bmatrix} = \begin{bmatrix}3\\3\\3\end{bmatrix}$$

$$\therefore \text{La hipótesis es falsa}$$

```
def checkNoAssociativityDoProduct():
  # Sample vectors
  u = torch.tensor([1, 1, 1])
  v = torch.tensor([1, 1, 1])
  w = torch.tensor([1, 1, 1])

  # Compute left and right side of the equation
  left = u * torch.dot(v, w)
  right = torch.dot(u, v) * w
```

```
  print ("Lado izquierdo de la ecuación:", left.data.cpu().numpy())
  print ("Lado derecho de la equción:", right.data.cpu().numpy())

  return not (torch.all(left.eq(right))).data.cpu().numpy()

print("Cumple el producto punto con la no asociatividad?", checkNoAssociativityDoProduct())
```

```
Lado izquierdo de la ecuación: [3 3 3]
Lado derecho de la equción: [3 3 3]
Cumple el producto punto con la no asociatividad? False
```

3. Funciones multivariable

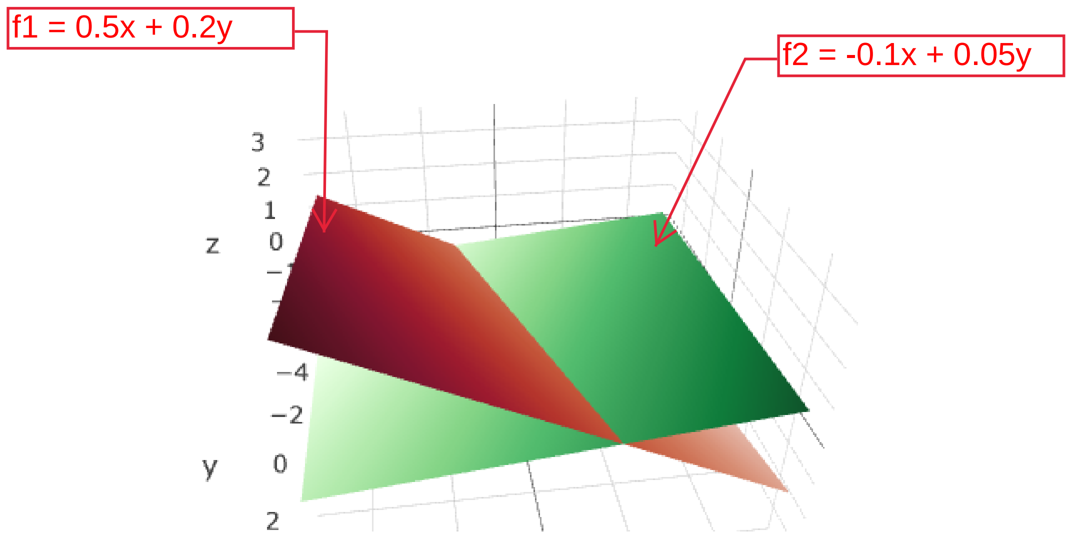
3.1 Funciones lineales multivariable:

Un hiperplano definido en un espacio $\mathbb{R}^{n+1}$ se puede expresar como una función con dominio $\vec{x} \in \mathbb{R}^n$ y codominio en $\mathbb{R}$ como sigue:
$z = f(\vec{x}) = \vec{x} \cdot \vec{w}$, con $\vec{w} \in \mathbb{R}^n$ el arreglo de coeficientes de tal funcional.

Tómese $\vec{w}_1 = \begin{bmatrix} 0.5 \\ 0.2 \end{bmatrix}$ para la función f_1 y $\vec{w}_2 = \begin{bmatrix} -0.1 \\ 0.05 \end{bmatrix}$ para la función f_2 , (funciones con dominio en $\mathbb{R}^2$ y codominio en). Grafique ambos planos en Pytorch.

```
import plotly.graph_objs as go
import numpy as np
np.random.seed(1)
configure_plotly_browser_state()


N = 21
x = np.linspace(-5.0, 5.0, N)
y = np.linspace(-5.0, 5.0, N)
xGrid, yGrid = np.meshgrid(y, x)
z = 0.5*xGrid  + 0.2*yGrid
z1 = -0.1*xGrid  + 0.05*yGrid


fig = go.Figure(data=[
                      go.Surface(name="f1=0.5*x+0.2*y", z=z, x=x, y=y,showscale=False, colorscale='amp'),
                      go.Surface(name="f2=-0.1*x+0.05*y",z=z1, x=x, y=y, showscale=False,colorscale='greens')
                      ])

fig.update_layout(title='Ejercicio 3', autosize=False,
                  width=500, height=500,
                  margin=dict(l=65, r=50, b=65, t=90))
fig.show()
```

[object Object]



3.2 El vector gradiente:

Para cada una de las siguientes funciones multivariable: (1) grafique su superficie con dominio entre -10 y 10 (2) calcule el vector gradiente manualmente, evaluelo y grafique el vector unitario en la dirección del gradiente para los dos puntos especificados (en la misma figura de la superficie) y (3) calcule la magnitud de tal vector gradiente en cada punto (4) Calcule lo que se conoce como la matriz Hessiana.

3.2.a

$f\left(x,y\right)=x^3y^2+1$

Evaluación del gradiente en los puntos $P_0=(0,0)$ y $P_1=(7.4,-6.3)$

1) Grafique su superficie con dominio entre -10 y 10

Ver código y gráfico

---

2)Calcule el vector gradiente manualmente, evaluelo y grafique el vector unitario en la dirección del gradiente para los dos puntos especificados (en la misma figura de la superficie)

$f(x_0,y_0)=x^3y^2+1$

$\nabla f(x_0,y_0)$=[$\frac{\partial f}{\partial x_{(x_0,y_0)}}$]$i+$[$\frac{\partial f}{\partial y_{(x_0,y_0)}}$]$j$

$\frac{\partial f}{\partial x_{(x_0,y_0)}}=3x^2y^2$

$\frac{\partial f}{\partial y_{(x_0,y_0)}}=2x^3y$

$\nabla f(x_0,y_0)$=[$3x^2y^2$]i+[$2x^3y$]j

Evaluación del gradiente en punto $P_0=(0,0)$

$\nabla f(0,0)$=[$3*0^20^2$]i+[$2*0^30$]$j=0i+0j$

Evaluación del gradiente en punto $P_1=(7.4,-6.3)$

$$\nabla f(7.4,-6.3) = [3*(7.4)^2(-6.3)^2]i + [2*(7.4)^3(-6.3)]j = 6520.27i - 5105.82j$$

---

3)Calcule la magnitud de tal vector gradiente en cada punto

Ver resultado del código, el cual está indicado por los enunciados

Norma del vector gradiente P0 = 0.0

Norma del vector gradiente P1 = 8281.50

---

4) Calcule lo que se conoce como la matriz Hessiana. H_f

$$H_f = \begin{bmatrix} \frac{\partial f}{\partial x^2_{(x_0,y_0)}} & \frac{\partial f}{\partial x_{(x_0,y_0)}\partial y_{(x_0,y_0)}} \\ \frac{\partial f}{\partial y_{(x_0,y_0)}\partial x_{(x_0,y_0)}} & \frac{\partial f}{\partial y^2_{(x_0,y_0)}} \end{bmatrix}$$

$$\frac{\partial f}{\partial x_{(x_0,y_0)}} = 3x^2y^2$$

$$\frac{\partial f}{\partial y_{(x_0,y_0)}\partial x_{(x_0,y_0)}} = 6x^2y$$

$$\frac{\partial f}{\partial x^2_{(x_0,y_0)}} = 6xy^2$$

$$\frac{\partial f}{\partial y_{(x_0,y_0)}} = 2x^3y$$

$$\frac{\partial f}{\partial x_{(x_0,y_0)}\partial y_{(x_0,y_0)}} = 6x^2y$$

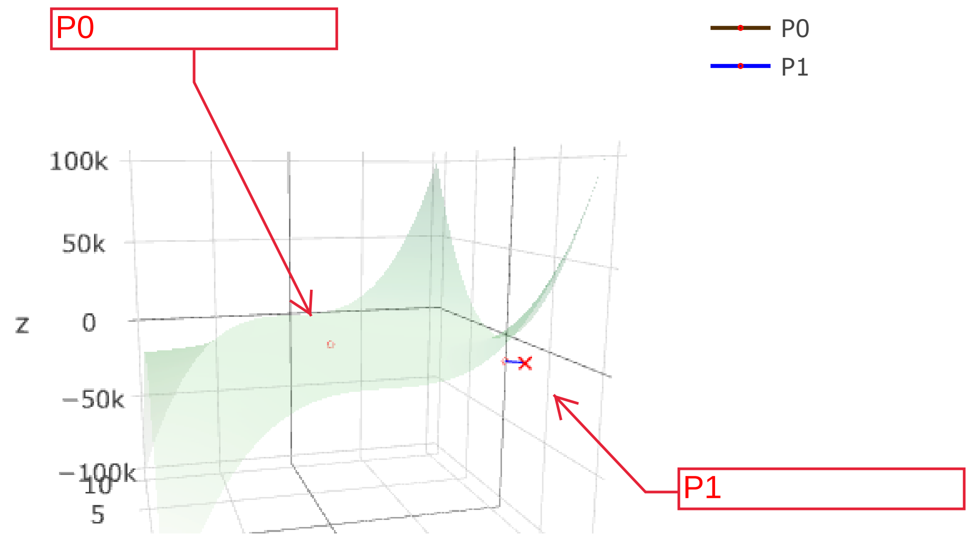
$$\frac{\partial f}{\partial y^2_{(x_0,y_0)}} = 2x^3$$

$$H_f = \begin{bmatrix} 6xy^2 & 6x^2y \\ 6x^2y & 2x^3 \end{bmatrix}$$

```
import plotly.graph_objs as go
import numpy as np
np.random.seed(1)
configure_plotly_browser_state()

def NormaL2(vector):
  vectorTensor = torch.FloatTensor(vector)
  vectorTensor = torch.pow(vectorTensor, 2)
  sum = torch.sum(vectorTensor)
  normaL2 = math.sqrt(sum)
  return normaL2;

print("****************************************")
print("Punto0")
print("Vector gradiente = [3*x^2*y^2]i+[2*x^3*y]j")
print("P(0.0,0.0)")
puntoEvaluacion0 = np.array([0,0])
dimX = puntoEvaluacion0.item(0)
dimY = puntoEvaluacion0.item(1)
vector0 = np.array([3*dimX*dimX*dimY*dimY,2*dimX*dimX*dimX*dimY])#calculo de vector gradiente
print("Vector gradiente = ", vector0)
```

```
magnitud = np.sqrt(np.dot(vector0,vector0))
vectorunitario0 = np.divide(vector0,magnitud)
print("Norma del vector gradiente P0 = ", NormaL2(vector0))
print("Vector gradiente unitario en P0 = ",vectorunitario0)
posicionVectorUnitario0 = puntoEvaluacion0 + vectorunitario0
print("****************************************")
print("Punto1")
print("Vector gradiente = [3*x^2*y^2]i+[2*x^3*y]j")
print("P(7.4,-6.3)")
puntoEvaluacion1 = np.array([7.4,-6.3])
dimX = puntoEvaluacion1.item(0)
dimY = puntoEvaluacion1.item(1)
vector1 = np.array([3*dimX*dimX*dimY*dimY,2*dimX*dimX*dimX*dimY])#calculo de vector gradiente
print("Vector gradiente = ", vector1)
magnitud = np.sqrt(np.dot(vector1,vector1))
vectorunitario1 = np.divide(vector1,magnitud)
print("Norma del vector gradiente P1 = ", NormaL2(vector1))
print("Vector gradiente unitario en P1 = ",vectorunitario1)
posicionVectorUnitario1 = puntoEvaluacion1 + vectorunitario1
print("****************************************")

N = 201
x = np.linspace(-10.0, 10.0, N)
y = np.linspace(-10.0, 10.0, N)
xGrid, yGrid = np.meshgrid(y, x)
z1 = xGrid*xGrid*xGrid*yGrid*yGrid+1


fig = go.Figure(data=[
                      go.Surface(name="f = x^3*y^2 + 1",z=z1, x=x, y=y, showscale=False,colorscale='greens',opacity=.9),
                      go.Scatter3d( name = "P0",
                                    x = [puntoEvaluacion0.item(0),posicionVectorUnitario0.item(0)],
                                    y = [puntoEvaluacion0.item(1),posicionVectorUnitario0.item(1)],
                                    z = [0,0],
                       marker = dict( size = 2, color = "rgb(255,0,0)", symbol = ["circle-open","x"]),
                       line = dict( color = "rgb(84,48,5)", width = 2)),
                      go.Scatter3d( name = "P1",
                                    x = [puntoEvaluacion1.item(0),posicionVectorUnitario1.item(0)],
                                    y = [puntoEvaluacion1.item(1),posicionVectorUnitario1.item(1)],
                                    z = [0,0],
                       marker = dict( size = 2, color = "rgb(255,0,0)", symbol = ["circle-open","x"]),
                       line = dict( color = "rgb(0,0,255)", width = 2))
                      ])

fig.update_layout(autosize=False,
                  width=500, height=500,
                  margin=dict(l=65, r=50, b=65, t=90))
fig.show()
```

```
****************************************
Punto0
Vector gradiente = [3*x^2*y^2]i+[2*x^3*y]j
P(0.0,0.0)
Vector gradiente =  [0 0]
Norma del vector gradiente P0 =  0.0
Vector gradiente unitario en P0 =  [nan nan]
****************************************
Punto1
Vector gradiente = [3*x^2*y^2]i+[2*x^3*y]j
P(7.4,-6.3)
Vector gradiente =  [ 6520.2732 -5105.8224]
Norma del vector gradiente P1 =  8281.508558227782
Vector gradiente unitario en P1 =  [ 0.78732916 -0.61653288]
****************************************
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:23: RuntimeWarning:

invalid value encountered in true_divide
```



3.2.b

$f(x, y) = \sin(x^2) + x\cos(y^3)$

Evaluación del gradiente en los puntos $P_0 = (1.5, -5.5)$ y $P_1 = (-10, -10)$

Nota: Este ejercicio fue desarrollado utilizando la unidad de radianes

---

1) Grafique su superficie con dominio entre -10 y 10

Ver código y gráfico

---

2)Calcule el vector gradiente manualmente, evaluelo y grafique el vector unitario en la dirección del gradiente para los dos puntos especificados (en la misma figura de la superficie)

$f(x, y) = \sin(x^2) + x\cos(y^3)$

$\nabla f(x_0, y_0)$=$[\frac{\partial f}{\partial x_{(x_0,y_0)}}]i + [\frac{\partial f}{\partial y_{(x_0,y_0)}}]j$

$$\frac{\partial f}{\partial x_{(x_0,y_0)}} = 2xcos\left(x^2\right) + cos\left(y^3\right)$$

$$\frac{\partial f}{\partial y_{(x_0,y_0)}} = -3xy^2\sin\left(y^3\right)$$

$$\nabla f(x_0,y_0)=[2xcos\left(x^2\right) + cos\left(y^3\right)]i+[-3xy^2\sin\left(y^3\right)]j$$

Evaluación del gradiente en punto $P_0 = (1.5, -5.5)$

$$\nabla f(1.5,-5.5)=[2(1.5)cos\left((1.5)^2\right) + cos\left((-5.5)^3\right)]i+[-3(1.5)(-5.5)^2\sin\left((-5.5)^3\right)]j$$

$$\nabla f(1.5,-5.5)=[-2.87]i+[17.57]j$$

Evaluación del gradiente en punto $P_1 = (-10, -10)$

$$\nabla f(-10,-10)=[2(-10)cos\left((-10)^2\right) + cos\left((-10)^3\right)]i+[-3(-10)(-10)^2\sin\left((-10)^3\right)]j$$

$$\nabla f(-10,-10)=[-16.68]i+[-2480]j$$

---

3)Calcule la magnitud de tal vector gradiente en cada punto

Ver resultado del código, el cual está indicado por los enunciados

Norma del vector gradiente P0 = 17.80

Norma del vector gradiente P1 = 2480.69

---

4) Calcule lo que se conoce como la matriz Hessiana.H_f

$$H_f=\begin{bmatrix} \frac{\partial f}{\partial x^2_{(x_0,y_0)}} & \frac{\partial f}{\partial x_{(x_0,y_0)}\partial y_{(x_0,y_0)}} \\ \frac{\partial f}{\partial y_{(x_0,y_0)}\partial x_{(x_0,y_0)}} & \frac{\partial f}{\partial y^2_{(x_0,y_0)}} \end{bmatrix}$$

$$\frac{\partial f}{\partial x_{(x_0,y_0)}} = 2xcos(x^2) + cos(y^3)$$

$$\frac{\partial f}{\partial y_{(x_0,y_0)}\partial x_{(x_0,y_0)}} = -3y^2sin(y^3)$$

$$\frac{\partial f}{\partial x^2_{(x_0,y_0)}} = 2[cos(x^2) + (-sin(x^2))(2x)(x)] = 2cos(x^2) - 4x^2sin(x^2)$$

$$\frac{\partial f}{\partial y_{(x_0,y_0)}} = -3xy^2sin(y^3)$$

$$\frac{\partial f}{\partial x_{(x_0,y_0)}\partial y_{(x_0,y_0)}} = -3y^2sin(y^3)$$

$$\frac{\partial f}{\partial y^2_{(x_0,y_0)}} = (-3x)[(y^2)'sin(y^3) + y^2(sin(y^3))'] = -6xysin(y^3) + (-3x)[3y^4cos(y^3)] = -6xysin(y^3) - 9xy^4cos(y^3)$$

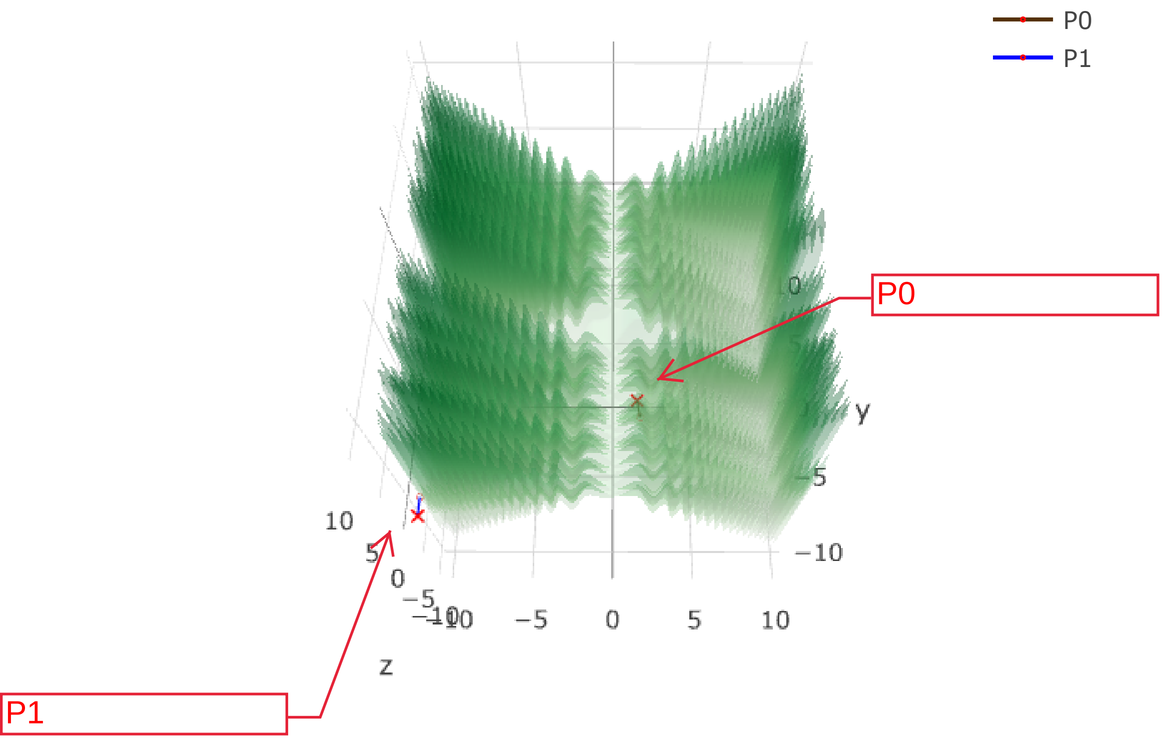
$$H_f=\begin{bmatrix} 2cos(x^2) - 4x^2sin(x^2) & -3y^2sin(y^3) \\ -3y^2sin(y^3) & -6xysin(y^3) - 9xy^4cos(y^3) \end{bmatrix}$$

```
import plotly.graph_objs as go
import numpy as np
import math
np.random.seed(1)
configure_plotly_browser_state()

def NormaL2(vector):
```

```
  vectorTensor = torch.FloatTensor(vector)
  vectorTensor = torch.pow(vectorTensor, 2)
  sum = torch.sum(vectorTensor)
  normaL2 = math.sqrt(sum)
  return normaL2;

print("****************************************")
print("Punto0")
print("Vector gradiente = [2*x*cos(x^2)+cos(y^3),-3*x*y^2*sin(y^3)]")
print("P(1.5,-5.5)")
puntoEvaluacion0 = np.array([1.5,-5.5])
dimX = puntoEvaluacion0.item(0)
dimY = puntoEvaluacion0.item(1)
vector0 = np.array([2*dimX*np.cos(dimX*dimX) + np.cos(dimY*dimY*dimY) , -3*dimX*dimY*dimY*np.sin(dimY*dimY*dimY)])#calculo de vector gradiente
print("Vector gradiente = ", vector0)
magnitud = np.sqrt(np.dot(vector0,vector0))
vectorunitario0 = np.divide(vector0,magnitud)
print("Vector gradiente unitario en P0 = ",vectorunitario0)
posicionVectorUnitario0 = puntoEvaluacion0 + vectorunitario0
print("Norma del vector gradiente P0 = ", NormaL2(vector0))
print("****************************************")
print("Punto1")
print("Vector gradiente = [2*x*cos(x^2)+cos(y^3),-3*x*y^2*sin(y^3)]")
print("P(-10,-10)")
puntoEvaluacion1 = np.array([-10,-10])
dimX = puntoEvaluacion1.item(0)
dimY = puntoEvaluacion1.item(1)
vector1 = np.array([2*dimX*np.cos(dimX*dimX) + np.cos(dimY*dimY*dimY) , -3*dimX*dimY*dimY*np.sin(dimY*dimY*dimY)])#calculo de vector gradiente
print("Vector gradiente = ", vector1)
magnitud = np.sqrt(np.dot(vector1,vector1))
vectorunitario1 = np.divide(vector1,magnitud)
print("Vector gradiente unitario en P1 = ",vectorunitario1)
posicionVectorUnitario1 = puntoEvaluacion1 + vectorunitario1
print("Norma del vector gradiente P1 = ", NormaL2(vector1))
print("****************************************")

N = 201
x = np.linspace(-10.0, 10.0, N)
y = np.linspace(-10.0, 10.0, N)
xGrid, yGrid = np.meshgrid(y, x)
z1 = np.sin(xGrid*xGrid) + xGrid*np.cos(yGrid*yGrid*yGrid)


fig = go.Figure(data=[
                      go.Surface(name="f=sin(x^2)+xcos(y^3)",z=z1, x=x, y=y, showscale=False,colorscale='greens',opacity=.9),
                      go.Scatter3d( name = "P0",
                                    x = [puntoEvaluacion0.item(0),posicionVectorUnitario0.item(0)],
                                    y = [puntoEvaluacion0.item(1),posicionVectorUnitario0.item(1)],
                                    z = [0,0],
                       marker = dict( size = 2, color = "rgb(255,0,0)", symbol = ["circle-open","x"]),
                       line = dict( color = "rgb(84,48,5)", width = 2)),
                      go.Scatter3d( name = "P1",
                                    x = [puntoEvaluacion1.item(0),posicionVectorUnitario1.item(0)],
                                    y = [puntoEvaluacion1.item(1),posicionVectorUnitario1.item(1)],
                                    z = [0,0],
                       marker = dict( size = 2, color = "rgb(255,0,0)", symbol = ["circle-open","x"]),
                       line = dict( color = "rgb(0,0,255)", width = 2))
```

```
                    ])

fig.update_layout(autosize=False,
                  width=500, height=500,
                  margin=dict(l=65, r=50, b=65, t=90))
fig.show()
```

```
****************************************
Punto0
Vector gradiente = [2*x*cos(x^2)+cos(y^3),-3*x*y^2*sin(y^3)]
P(1.5,-5.5)
Vector gradiente =  [-2.87615899 17.56689497]
Vector gradiente unitario en P0 =  [-0.1615748   0.98686047]
Norma del vector gradiente P0 =  17.80078853617911
****************************************
Punto1
Vector gradiente = [2*x*cos(x^2)+cos(y^3),-3*x*y^2*sin(y^3)]
P(-10,-10)
Vector gradiente =  [  -16.68399837 -2480.6386216 ]
Vector gradiente unitario en P1 =  [-0.00672553 -0.99997738]
Norma del vector gradiente P1 =  2480.6947615537065
****************************************
```



3.2.c

$f(x,y) = 3^{2x} + 5^{4y} + 2x + y^4$

Evaluación del gradiente en los puntos $P_0 = (-4,-2)$ y $P_1 = (-2,9)$

1) Grafique su superficie con dominio entre -10 y 10

Ver código y gráfico

2)Calcule el vector gradiente manualmente, evaluelo y grafique el vector unitario en la dirección del gradiente para los dos puntos especificados (en la misma figura de la superficie)

$$f(x,y) = 3^{2x} + 5^{4y} + 2x + y^4$$

$$\nabla f(x_0, y_0) = [\frac{\partial f}{\partial x_{(x_0,y_0)}}]i + [\frac{\partial f}{\partial y_{(x_0,y_0)}}]j$$

$$\frac{\partial f}{\partial x_{(x_0,y_0)}} = 2 + 9^x ln(3) * (2x)' = 2 + 9^x ln(9)$$

$$\frac{\partial f}{\partial y_{(x_0,y_0)}} = 5^{4y} ln(5)(4y)' + 4y^3 = 4y^3 + 625^y ln(625)$$

$$\nabla f(x_0, y_0) = [2 + 9^x ln(9)]i + [4y^3 + 625^y ln(625)]j$$

Evaluación del gradiente en punto $P_0 = (-4, -2)$

$$\nabla f(-4,-2) = [2 + 9^{(-4)} ln(9)]i + [4(-2)^3 + 625^{(-2)} ln(625)]j$$

$$\nabla f(-4,-2) = [2.00]i + [-31.99]j$$

Evaluación del gradiente en punto $P_1 = (-2, 9)$

$$\nabla f(-2,9) = [2 + 9^{-2} ln(9)]i + [4 * 9^3 + 625^9 ln(625)]j$$

$$\nabla f(-2,9) = [2.03]i + [9.37E^{25}]j$$

---

3)Calcule la magnitud de tal vector gradiente en cada punto

Ver resultado del código, el cual está indicado por los enunciados

Norma del vector gradiente P0 = 32.05

Norma del vector gradiente P1 = $9.37E^{25}$

---

4) Calcule lo que se conoce como la matriz Hessiana. H_f

$$H_f = \begin{bmatrix} \frac{\partial f}{\partial x^2_{(x_0,y_0)}} & \frac{\partial f}{\partial x_{(x_0,y_0)} \partial y_{(x_0,y_0)}} \\ \frac{\partial f}{\partial y_{(x_0,y_0)} \partial x_{(x_0,y_0)}} & \frac{\partial f}{\partial y^2_{(x_0,y_0)}} \end{bmatrix}$$

$$\frac{\partial f}{\partial x_{(x_0,y_0)}} = 2 + 9^x ln(9)$$

$$\frac{\partial f}{\partial y_{(x_0,y_0)} \partial x_{(x_0,y_0)}} = 0$$

$$\frac{\partial f}{\partial x^2_{(x_0,y_0)}} = 9^x ln(9) ln(9) = 9^x (ln(9))^2$$

$$\frac{\partial f}{\partial y_{(x_0,y_0)}} = 4y^3 + 625^y ln(625)$$

$$\frac{\partial f}{\partial x_{(x_0,y_0)} \partial y_{(x_0,y_0)}} = 00$$

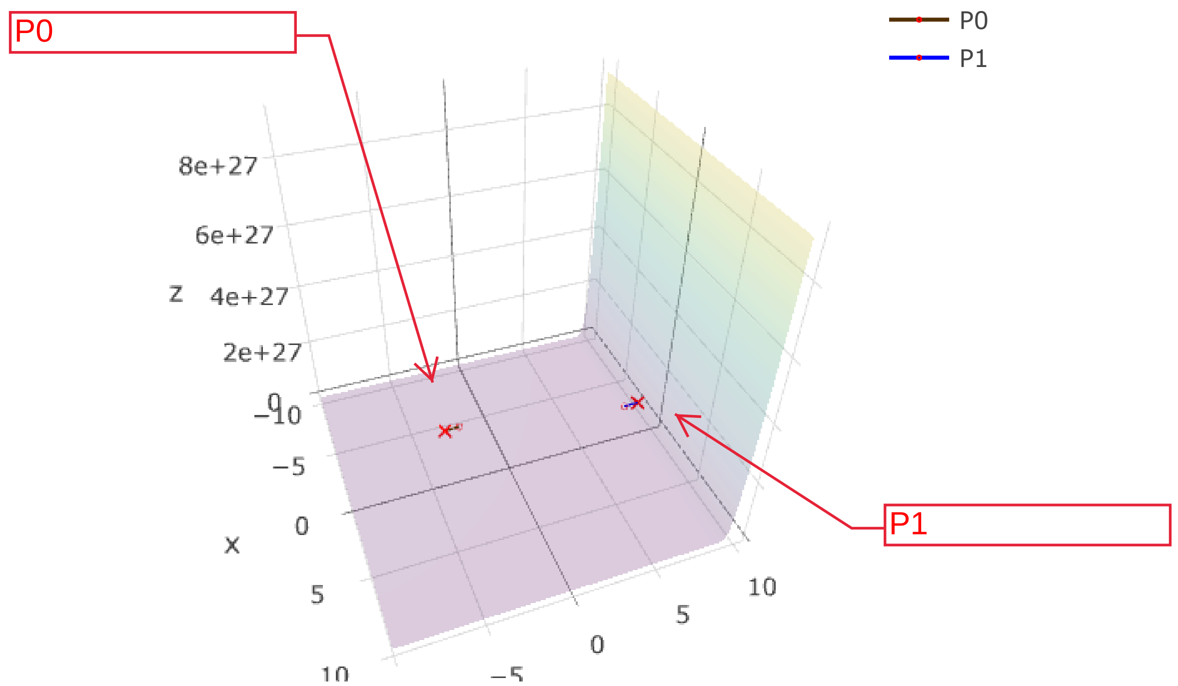
$$\frac{\partial f}{\partial y^2_{(x_0,y_0)}} = 4(3y^2) + 625^y ln(625) ln(625) = 12y^2 + 625^y (ln(625))^2$$

$$H_f = \begin{bmatrix} 9^x (ln(9))^2 & 0 \\ 0 & 12y^2 + 625^y (ln(625))^2 \end{bmatrix}$$

```
import plotly.graph_objs as go
import numpy as np
import math
np.random.seed(1)
configure_plotly_browser_state()

def NormaL2(vector):
  vectorTensor = torch.FloatTensor(vector)
  vectorTensor = torch.pow(vectorTensor, 2)
  sum = torch.sum(vectorTensor)
  normaL2 = math.sqrt(sum)
  return normaL2;

print("****************************************")
print("Punto0")
print("Vector gradiente = [2 + (9^x)(ln(9)),4(y^3)+(625^y)(ln(625))]")
print("P(-4,-2)")
puntoEvaluacion0 = np.array([-4,-2])
dimX = puntoEvaluacion0.item(0)
dimY = puntoEvaluacion0.item(1)
vector0 = np.array([2+ (pow(9,dimX))*np.log(9), 4*(pow(dimY,3))+(pow(625,dimY))*np.log(625)])#calculo de vector gradiente
print("Vector gradiente = ", vector0)
magnitud = np.sqrt(np.dot(vector0,vector0))
vectorunitario0 = np.divide(vector0,magnitud)
print("Vector gradiente unitario en P0 = ",vectorunitario0)
posicionVectorUnitario0 = puntoEvaluacion0 + vectorunitario0
print("Norma del vector gradiente P0 = ", NormaL2(vector0))
print("****************************************")
print("Punto1")
print("Vector gradiente = [2 + (9^x)(ln(9)),4(y^3)+(625^y)(ln(625))]")
print("P(-2,9)")
puntoEvaluacion1 = np.array([-2,9])
dimX = puntoEvaluacion1.item(0)
dimY = puntoEvaluacion1.item(1)
vector1 = np.array([2+ (pow(9,dimX))*np.log(9), 4*(pow(dimY,3))+(pow(625,dimY))*np.log(625)])#calculo de vector gradiente
print("Vector gradiente = ", vector1)
magnitud = np.sqrt(np.dot(vector1,vector1))
vectorunitario1 = np.divide(vector1,magnitud)
print("Vector gradiente unitario en P1 = ",vectorunitario1)
posicionVectorUnitario1 = puntoEvaluacion1 + vectorunitario1
print("Norma del vector gradiente P1 = ", NormaL2(vector1))
print("****************************************")

N = 201
x = np.linspace(-10.0, 10.0, N)
y = np.linspace(-10.0, 10.0, N)
xGrid, yGrid = np.meshgrid(y, x)
z1 = (np.power(3,2*xGrid))+(np.power(5,4*yGrid))+2*xGrid+(np.power(yGrid,4))


fig = go.Figure(data=[
                      go.Surface(name="f=3^(2x)+5^(4y)+2x+y^4",z=z1, x=x, y=y, showscale=False,colorscale='Viridis', opacity=0.92),
                      go.Scatter3d( name = "P0",
                                    x = [puntoEvaluacion0.item(0),posicionVectorUnitario0.item(0)],
                                    y = [puntoEvaluacion0.item(1),posicionVectorUnitario0.item(1)],
                                        [0,0]
```

```
                    z = [0,0],
         marker = dict( size = 2, color = "rgb(255,0,0)", symbol = ["circle-open","x"]),
         line = dict( color = "rgb(84,48,5)", width = 2)),
        go.Scatter3d( name = "P1",
                    x = [puntoEvaluacion1.item(0),posicionVectorUnitario1.item(0)],
                    y = [puntoEvaluacion1.item(1),posicionVectorUnitario1.item(1)],
                    z = [0,0],
         marker = dict( size = 2, color = "rgb(255,0,0)", symbol = ["circle-open","x"]),
         line = dict( color = "rgb(0,0,255)", width = 2))
        ])

fig.update_layout(autosize=False,
                  width=500, height=500,
                  margin=dict(l=65, r=50, b=65, t=90))
fig.show()
```

```
****************************************
Punto0
Vector gradiente = [2 + (9^x)(ln(9)),4(y^3)+(625^y)(ln(625))]
P(-4,-2)
Vector gradiente =  [  2.00033489 -31.99998352]
Vector gradiente unitario en P0 =  [ 0.06238872 -0.99805193]
Norma del vector gradiente P0 =  32.06244289103101
****************************************
Punto1
Vector gradiente = [2 + (9^x)(ln(9)),4(y^3)+(625^y)(ln(625))]
P(-2,9)
Vector gradiente =  [2.02712623e+00 9.36816163e+25]
Vector gradiente unitario en P1 =  [2.16384634e-26 1.00000000e+00]
Norma del vector gradiente P1 =  inf
****************************************
```



Pregunta 4

Distancia de Minkowski como índice de error

Datos

El error medio absoluto o MAE en inglés

$MAE = \frac{1}{m}\sum_{i=1}^{m}|\tilde{t}_i - t_i|$

Raíz del error cuadrado medio o RMSE en inglés

$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\tilde{t}_i - t_i)^2}$

Unidades de variable de interés: quintales por hectárea

¿Cuál métrica es más sensible a los valores atípicos?

Observando las particiones, la número 1 la diferencia entre el valor observado y el estimado es de dos para todos los casos, y se obtiene el mismo resultado para el MAE y RMSE.

Partición 1. Datos 1 al 10

```
ti1=torch.tensor([4.0,6,5,6,8,10,7,4,2,8])
tie1=torch.tensor([2.0,4,3,4,6,8,5,2,4,10])
difabs1=torch.abs(tie1 - ti1)
print("Diferencia abs",difabs1.data.cpu().numpy())

demae1= torch.std(difabs1)
print("DE MAE1",demae1.data.cpu().numpy())

difsqrt1= torch.pow(tie1-ti1,2)
print("Diferencia al cuadrado",difsqrt1)

dermse1=torch.std(difsqrt1)
print("DE RMSE1",dermse1.data.cpu().numpy())

#Calculo del MAE
MAE1 = 1/10 * torch.sum(difabs1)
print("MAE1",MAE1.data.cpu().numpy())

#Calculo de RMSE
RMSE1=torch.sqrt(1/10 * torch.sum(difsqrt1) )
print("RMSE1",RMSE1.data.cpu().numpy())
```

```
Diferencia abs [2. 2. 2. 2. 2. 2. 2. 2. 2. 2.]
DE MAE1 0.0
Diferencia al cuadrado tensor([4., 4., 4., 4., 4., 4., 4., 4., 4., 4.])
DE RMSE1 0.0
MAE1 2.0
RMSE1 2.0
```

Partición 2. Datos del 11 al 20

```
ti2=torch.tensor([5.0,3,2,4,20,32,5,4,7,41])
tie2=torch.tensor([4.0,2,3,5,21,29,2,7,4,38])

difabs2=torch.abs(tie2 - ti2)
print("Diferencia abs",difabs2.data.cpu().numpy())
difsqrt2=torch.tensor([1,1.0,1,1,1,9,9,9,9,9])

demae2= torch.std(difabs2)
print("DE MAE2",demae2.data.cpu().numpy())
dermse2=torch.std(difsqrt2)
print("DE RMSE2",dermse2.data.cpu().numpy())

#Calculo del MAE
MAE2 = 1/10 * torch.sum(difabs2)
print("MAE2",MAE2.data.cpu().numpy())

#Calculo de RMSE
RMSE2=torch.sqrt(1/10 * torch.sum(difsqrt2) )
print("RMSE2",RMSE2.data.cpu().numpy())
```

```
Diferencia abs [1. 1. 1. 1. 1. 3. 3. 3. 3. 3.]
DE MAE2 1.0540925
DE RMSE2 4.21637
MAE2 2.0
RMSE2 2.236068
```

Partición 3. Datos del 21 al 30

```
ti3=torch.tensor([6.0,20,31,41,50,62,73,4,7,40])
tie3=torch.tensor([6,20,31,41,50.0,62,73,4,7,20])
difabs3=torch.abs(tie3 - ti3)
print("Diferencia abs",difabs3.data.cpu().numpy())

difsqrt3= torch.pow(tie3-ti3,2)
print("Diferencia al cuadrado",difsqrt3.data.cpu().numpy())


demae3= torch.std(difabs3)
print("DE MAE3",demae3.data.cpu().numpy())
dermse3=torch.std(difsqrt3)
print("DE RMSE3",dermse3.data.cpu().numpy())


#Calculo del MAE
MAE3 = 1/10 * torch.sum(difabs3)
print("MAE3",MAE3.data.cpu().numpy())

#Calculo de RMSE
RMSE3=torch.sqrt(1/10 * torch.sum(difsqrt3) )
print("RMSE3",RMSE3.data.cpu().numpy())
```

```
Diferencia abs [ 0.  0.  0.  0.  0.  0.  0.  0.  0. 20.]
Diferencia al cuadrado [  0.   0.   0.   0.   0.   0.   0.   0.   0. 400.]
DE MAE3 6.3245554
DE RMSE3 126.491104
MAE3 2.0
RMSE3 6.3245554
```

Nota, entiendase por

Real: t_i

Estimado: $\tilde{t}_i$

Diferencia abs: $|\tilde{t}_i - t_i|$

Diferencia cuadrada: $(\tilde{t}_i - t_i)^2$

Indice	Real	Estimado	Diferencia abs	Diferencia cuadrada
1	4	2	2	4
2	6	4	2	4
3	5	3	2	4
4	6	4	2	4
5	8	6	2	4
6	10	8	2	4
7	7	5	2	4
8	4	2	2	4
9	2	4	2	4
10	8	10	2	4
11	5	4	1	1
12	3	2	1	1
13	2	3	1	1
14	4	5	1	1
15	20	21	1	1
16	32	29	3	9
17	5	2	3	9
18	4	7	3	9
19	7	4	3	9
20	41	38	3	9
21	6	6	0	0
22	20	20	0	0
23	31	31	0	0
24	41	41	0	0
25	50	50	0	0
26	62	62	0	0
27	73	73	0	0
28	4	4	0	0
29	7	7	0	0
30	40	20	20	400

Resultados	Partición 1	Partición 2	Partición 3
MAE	2	2	2
RMSE	2	2.24	6.32
Desviación estandar MAE	0	1.05	6.32
Desviación estandar RMSE	0	4.22	126.49

¿Cuál métrica es más sensible a los valores atípicos?

Observando las particiones, en la número 1 la diferencia entre el valor observado y el estimado es de dos unidades para todos los casos, y se obtiene el mismo resultado para el MAE y RMSE, lo que quiere decir que el error medio absoluto es 2 quintales por hectárea y la raíz del error medio cuadrado es 2 quintales por hectárea.

En la segunda partición las diferencias entre el valor observado y el real son de uno o tres unidades, en este caso el error medio absoluto es de 2 quintales por hectárea y la raíz del error medio cuadrado es de 2.24 quintales por hectárea, se puede decir que en esta partición no hay valores atipicos, sin embargo, el MAE se mantiene en 2 quintales por hectárea.

Por último, la tercera partición no hay diferencias entre el valor real y el estimado, salvo en la última observación donde hay un error de 20 unidades, esto afecta a RMSE, el aumento se puede deber a que el error lo eleva al cuadrado. El MAE se mantienen en 2 quintales por hectárea. Demostrando así que al tener un valor atípico no se ve afectado como el RMSE.