Matemática para Ciencia de los Datos

Trabajo Práctico 1

Profesor: Luis Alexánder Calvo Valverde

Instituto Tecnológico de Costa Rica,

Programa Ciencia de Datos

Fecha de entrega: Lunes 24 de Abril del 2023, a más tardar a las 3:00 pm.

Medio de entrega: Por medio del TEC-Digital.

Entregables: Un archivo jupyter (.IPYNB).

Estudiante(s):

1. **Marco Ferraro Rodriguez**

In [ ]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Pregunta 1 (20 puntos, 10 pts c/u)

Demuestre de manera matemática si los siguientes sistemas $L\{\cdot\}$ (con entrada $u(t)$ y salida $g(t)$) son lineales o no lineales (**escriba las fórmulas en celdas de texto**). Una vez hecho su mayor esfuerzo, si no sabe cómo seguir matemáticamente, puede sustituir con valores y mostrar por contra-ejemplo.

a)
$g\left(t\right) = \log_{10} u(t)$

$L\left\{\alpha f_1\left(x\right) + \beta f_2\left(x\right)\right\} = \alpha L\left\{f_1(x)\right\} + \beta L\left\{f_2(x)\right\}$

$\log_{10}\left(\alpha u_1\left(t\right) + \beta u_2(t)\right) =? \, \alpha \log_{10} u_1\left(t\right) + \beta \log_{10} u_2\left(t\right)$

$\log_{10}\left(\alpha u_1\left(t\right) + \beta u_2(t)\right) =? \log_{10}\left(u_1(t)^{\alpha} u_2(t)^{\beta}\right)$

No es un sistema lineal

b)
$g\left(t\right) = 5 * u(t) + 13$

$$L\left\{\alpha f_1\left(x\right)+\beta f_2\left(x\right)\right\}=\alpha L\left\{f_1(x)\right\}+\beta L\left\{f_2(x)\right\}$$

$$5(\alpha u_1\left(t\right)+\beta u_2(t))+13 =? \alpha(5u_1\left(t\right)+13)+\beta(5u_2\left(t\right)+13)$$

$$5\alpha u_1\left(t\right)+5\beta u_2(t)+13 =? 5\alpha u_1\left(t\right)+13\alpha+5\beta u_2\left(t\right)+13\beta$$

$$5\alpha u_1\left(t\right)+5\beta u_2(t)+13 =? 5\alpha u_1\left(t\right)+5\beta u_2\left(t\right)13(\alpha+\beta)$$

No es un sistema lineal

Pregunta 2 (20 puntos, 10 pts c/u)

Para cada uno de los siguientes vectores calcule la norma L_2:

1. De manera matemática.
2. Programe una implementación en python de lo anterior, pero sin utilizar la función **norm** de la biblioteca.
3. Luego compare su resultado con una versión usando **norm**.

In [ ]:
```
def norm(vector, p):
  sum = 0
  for i in vector:
    sum += i**p
  return sum**(1/p)
```

a)

$$a=\begin{bmatrix}-9\\7\end{bmatrix}$$

$$norma=\sqrt{9^2+7^2}$$

$$norma=\sqrt{81+49}$$

$$norma=\sqrt{130}$$

$$norma=11.4$$

In [ ]:
```
norm([-9, 7], p=2)
```

Out[ ]:
```
11.40175425099138
```

In [ ]:
```
from numpy import linalg as LA

LA.norm([-9, 7])
```

Out[ ]:
```
11.40175425099138
```

b)

$$b = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 5 \end{bmatrix}$$

$$norma = \sqrt{1^2 + 2^2 + 4^2 + 5^2}$$

$$norma = \sqrt{1 + 4 + 16 + 25}$$

$$norma = \sqrt{46}$$

$$norma = 6.78$$

In [ ]:
```
norm([1, 2, 4, 5], p=2)
```

Out[ ]:
```
6.782329983125268
```

In [ ]:
```
from numpy import linalg as LA

LA.norm([1, 2, 4, 5])
```

Out[ ]:
```
6.782329983125268
```

Pregunta 3 (20 puntos, 10 pts c/u)

En Python, calcule el producto punto (o producto escalar entre vectores) $a \cdot b$ para los siguientes pares de vectores, una versión utilizando **dot**, y otra sin utilizar dicha función (programa en python con ciclos):

In [ ]:
```
def dot(vector_a, vector_b):
    if len(vector_a) == len(vector_b):
        sum = 0
        for i in range(len(vector_a)):
            sum += vector_a[i] * vector_b[i]
        return sum
    else:
        return "error"
```

a)

$$a = \begin{bmatrix} 2 \\ 7 \end{bmatrix}, b = \begin{bmatrix} 10 \\ 2 \end{bmatrix}$$

In [ ]:
```
dot([2, 7], [10, 2])
```

Out[ ]:
```
34
```

In [ ]:
```
import numpy as np

np.dot([2, 7], [10, 2])
```

Out[ ]:
```
34
```

b)

$$a = \begin{bmatrix} -1 \\ 8 \\ 3 \end{bmatrix}, b = \begin{bmatrix} 2 \\ 6 \\ 5 \end{bmatrix}$$

In [ ]:
```
dot([-1, 8, 3], [2, 6, 5])
```

Out[ ]: 61

In [ ]:
```
import numpy as np

np.dot([-1, 8, 3], [2, 6, 5])
```

Out[ ]: 61

Pregunta 4 (20 puntos, 10 pts c/u)

a) Proponga dos vectores: x e y que sean colineales (con dos elementos cada uno). Programe en python para mostrar que son colineales y luego grafíquelos en un mismo gráfico en python. En el cuaderno visto en clase viene un ejemplo de uso de import matplotlib.pyplot as plt

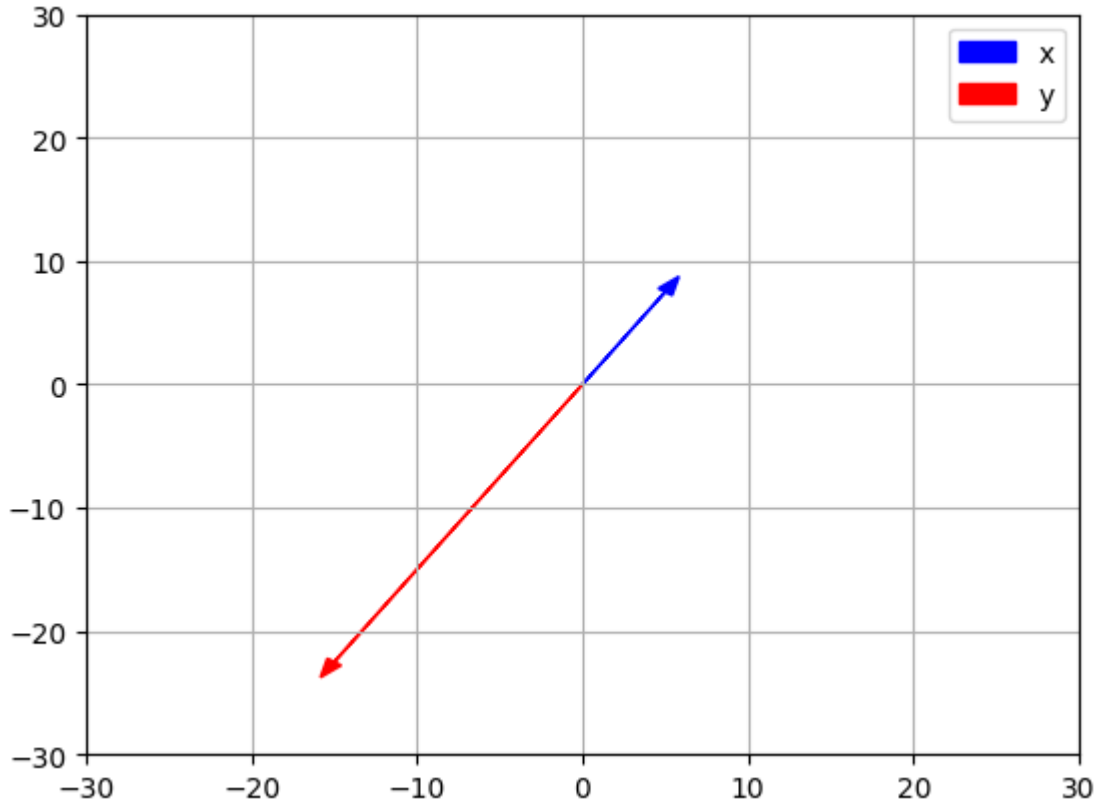
In [ ]:
```
import numpy as np

def are_collinear(vector_a, vector_b):
    vector_a = np.array(vector_a)
    vector_b = np.array(vector_b)
    alpha = vector_b[0] / vector_a[0]

    if np.array_equal(vector_a * alpha, vector_b):
        return True
    else:
        return False
```

In [ ]:
```
import numpy as np

x = np.array([5.0, 7.5])
y = np.array([-15.0, -22.5])

are_collinear(x, y)
```

Out[ ]: True

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

def plot_vectors(x, y, x_lim=30, y_lim=30):
  # Create a plot
  plt.figure()

  # Plot the vectors as arrows
  plt.arrow(0, 0, x[0], x[1], head_width=1, head_length=1.5, color='b', label='x')
  plt.arrow(0, 0, y[0], y[1], head_width=1, head_length=1.5, color='r', label='y')



  # Set the x and y limits of the plot
  plt.xlim(-1 * x_lim, x_lim)
  plt.ylim(-1 * y_lim, y_lim)

  # Add a grid and legend
  plt.grid()
  plt.legend()

  # Show the plot
  plt.show()
```

In [ ]:
```
plot_vectors(x, y)
```



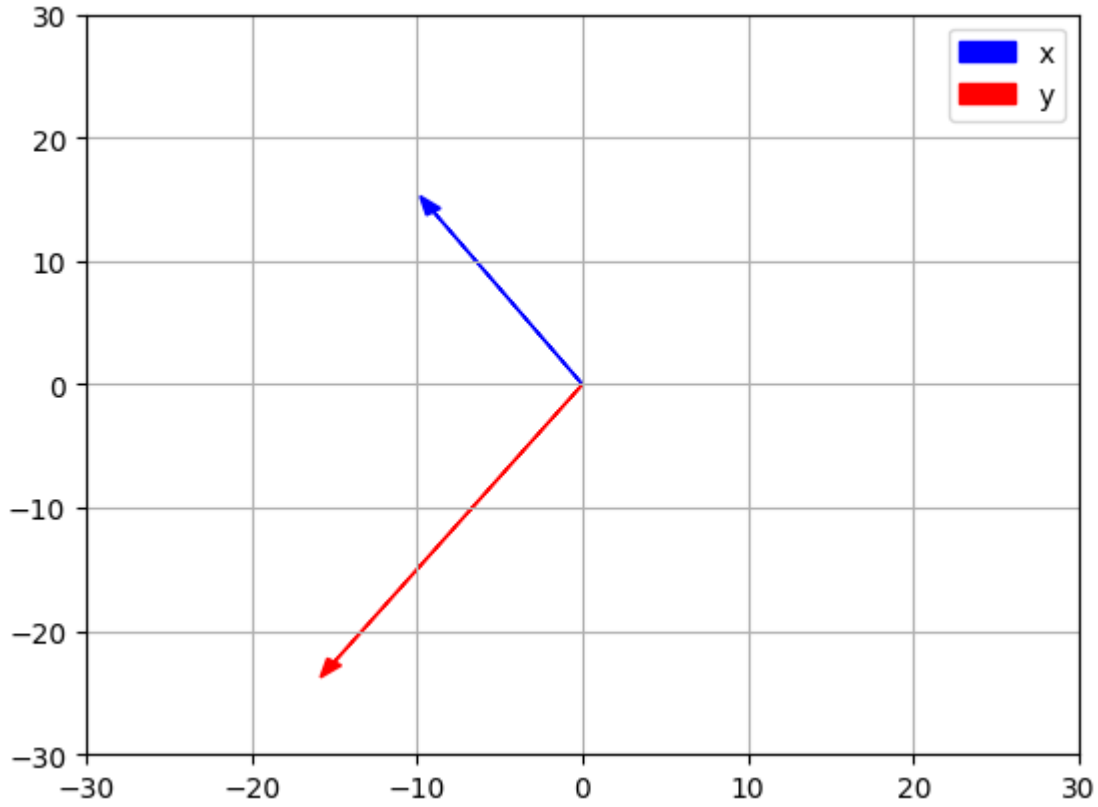
b)

Ahora modifique uno de los vectores para que no sean colineales y luego grafique para mostrar cómo se ven dos vectores que no son colineales.

In [ ]:
```
x = [-9, 14]

are_collinear(x, y)
```

Out[ ]: False

In [ ]:
```
plot_vectors(x, y)
```



Notas:

- Proponga significa que Ustedes, estudiando las propiedades de los vectores colineales propone los valores del vector.
- En ambos casos tome como origen el punto (0,0)

Pregunta 5 (20 puntos, 10 pts c/u)

a) Cargue el archivo llamado "Datos_01.csv".

¿Existen atributos colineanes? ¿Cuáles? Programe en python para mostrar su respuesta.

In [ ]:
```
separador = "-"*40
archivo = "Datos_01.csv"

# carga el archivo en pandas
dataFrame = pd.read_csv(archivo, header = 0, delimiter=';')

# guarda el nombre de las columnas en una lista
colNames = dataFrame.columns

# muestra los primeros elementos del dataFrame
print(separador)
print("Datos en dataFrame:")
print(dataFrame.head() )

# Convertir de pandas a numpy
datos = pd.DataFrame(dataFrame).to_numpy()

# En cada vector columna hay un atributo
# El atributo1 está en datos[:,0]
# El atributo2 está en datos[:,1]
# y así sucesivamente
```

```
----------------------------------------
Datos en dataFrame:
   atributo1  atributo2  atributo3  atributo4  atributo5
0        515         15   0.408462     1287.5          8
1        357         22   0.642985      892.5          9
2        633         20   0.582240     1582.5          7
3        295         17   0.531009      737.5          9
4        946         14   0.340640     2365.0          3
```

In [ ]:
```
# Programar para determinar vectores colineales

n_fields = datos.shape[1]
start = 1

for i in range(n_fields):
    for j in range(start, n_fields):
        if (are_collinear(datos[:, i], datos[:, j])):
            print(f"Columns {i} and {j} are collinear")
    start += 1
```

```
Columns 0 and 3 are collinear
```

b)

En el archivo "reales.csv" se encuentran los valores reales de un conjunto de datos, y en el archivo "predicciones.csv" lo que predijo un algoritmo.

Cargue ambos archivos y muestre la norma 2 y la norma 5, de la diferencia entre el real y el predicho.

Finalmente, grafique el predicho y el real en un mismo gráfico para comparar.

In [ ]:
```
# carga archivos

archivo = "reales.csv"
dataFrameReales = pd.read_csv(archivo, header = 0)


archivo = "predicciones.csv"
dataFramePredichos = pd.read_csv(archivo, header = 0)


# Convertir de pandas a numpy
true_values = dataFrameReales.to_numpy().squeeze()
predict_values = dataFramePredichos.to_numpy().squeeze()


true_values.shape
# Graficar
```

Out[ ]:
```
(506,)
```

In [ ]:
```
# Calcular las normas 2

norm(true_values - predict_values, 2)
```

Out[ ]:
```
72.92464045094769
```

In [ ]:
```
# Calcular las normas 5

norm(true_values - predict_values, 5)
```

Out[ ]:
```
13.186826079217848
```

In [ ]:
```
plot_vectors(predict_values, true_values, x_lim=15, y_lim=15)
```