LAHORE UNIVERSITY OF MANAGEMENT SCIENCES
Syed Babar Ali School of Science and Engineering

EE212 Mathematical Foundations for Machine Learning and Data Science
Fall Semester 2021

Programming Assignment 1 – Linear Independence, Basis & Matrix Rank

---

Total Marks: 100

Contribution to Final Assessment: 2%

Submission deadline: 23:55, Wednesday, September 29, 2021.

Goal

The goal of this programming assignment is to strengthen the concepts related to linear independence, basis and rank of a matrix along with their implementation in Python.

Instructions

Name your files Task1.py, Task2.py and so on. Compress them in a **single** file and name it as PAXX_YourRollNumber. Submit this file on LMS before the deadline. Late submissions will not be accepted.

Before starting, import the following libraries from Python:

```
import numpy as np
```

Task 1: Linear Independence (40 Marks)

A collection of n-vectors $a_1, a_2, ..., a_k$ (with $k \geq 1$) is called linearly independent if the coefficients $\beta_1, \beta_2, ..., \beta_k$ are all equal to zero, causing the linear combination $\beta_1 a_1 + \beta_2 a_2 + ... + \beta_k a_k$ to result in a zero vector i.e.

$$\beta_1 a_1 + \beta_2 a_2 + ... + \beta_k a_k = 0$$

if and only if

$$\beta_1 = \beta_2 = ...\beta_k = 0$$

1. Using the above relationship, evaluate β_1 and β_2 for the following set of vectors. Comment if they are linearly independent or not.

$$\boldsymbol{a_1} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}, \quad \boldsymbol{a_2} = \begin{pmatrix} -6 \\ -4 \end{pmatrix}$$

2. Consider the vector $\boldsymbol{b_1} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$. Use the Python code given below to visualize this vector.

```
v1 = [ 2, 1]
M = np.array([v1])
ax = plt.axes()
ax.arrow(0,0,M[0,0],M[0,1],
    head_width=0.10,head_length=0.1,color = 'b')
plt.plot(0,0,'ok')
maxes = 1.1*np.amax(abs(M), axis = 0)
plt.xlim([-maxes[0],maxes[0]])
plt.ylim([-maxes[1],maxes[1]])
plt.grid(b=True, which='major')
ax.set_axisbelow(True)
plt.show()
```

3. Modify the code above to add another vector $\boldsymbol{b_2} = \begin{pmatrix} 6 \\ 3 \end{pmatrix}$.

   Visualize the two vectors and comment if they are linearly independent.

4. Add another vector $\boldsymbol{b_2} = \begin{pmatrix} -4 \\ -2 \end{pmatrix}$ and comment if the three vectors are linearly independent.

5. By plotting the following set of vectors or otherwise, identify if they span a line or a plane.

   - $\boldsymbol{c_1} = \begin{pmatrix} 4 \\ 2 \end{pmatrix} \quad \boldsymbol{c_2} = \begin{pmatrix} -8 \\ -4 \end{pmatrix}$
   - $\boldsymbol{d_1} = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \quad \boldsymbol{d_2} = \begin{pmatrix} -2 \\ 5 \end{pmatrix}$
   - $\boldsymbol{e_1} = \begin{pmatrix} -3 \\ 6 \end{pmatrix} \quad \boldsymbol{e_2} = \begin{pmatrix} -1 \\ 2 \end{pmatrix} \quad \boldsymbol{e_3} = \begin{pmatrix} 4 \\ -8 \end{pmatrix}$
   - $\boldsymbol{f_1} = \begin{pmatrix} -3 \\ 6 \end{pmatrix} \quad \boldsymbol{f_2} = \begin{pmatrix} -1 \\ 2 \end{pmatrix} \quad \boldsymbol{f_3} = \begin{pmatrix} 4 \\ -9 \end{pmatrix}$

Task 2: Basis (30 Marks)

As you would recall from your lectures, a collection of n linearly independent n-vectors is called a *basis* and if $a_1, a_2, ..., a_n \in \mathbb{R}^n$ then any vector $b \in \mathbb{R}^n$ can be represented uniquely as $b = \beta_1 a_1 + \beta_2 a_2 + ... + \beta_n a_n$

1. Consider the set of vectors that are a basis: $\boldsymbol{a_1} = \begin{pmatrix} 1.2 \\ -2.6 \end{pmatrix}$, $\boldsymbol{a_2} = \begin{pmatrix} -0.3 \\ -3.7 \end{pmatrix}$.

   Calculate β_1 and β_2 such that $b = \beta_1 a_1 + \beta_2 a_2$, where $\boldsymbol{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

2. Consider another set of vectors that are a basis: $\boldsymbol{a_1} = \begin{pmatrix} 2 \\ -2 \\ 3 \end{pmatrix}$, $\boldsymbol{a_2} = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix}$.$\boldsymbol{a_3} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$. Calculate β_1, β_2 and β_3 such that $b = \beta_1 a_1 + \beta_2 a_2 + \beta_3 a_3$, where $\boldsymbol{b} = \begin{pmatrix} 1 \\ 3 \\ 2 \end{pmatrix}$.

3. Modify the following Python script to verify your answers in part 1:

```
# 5x + 3y = 40
# 1x + 2y = 18
A = np.array([[5,3],[1,2]])
B = np.array([40,18])
C = np.linalg.solve(A,B)
print(C)
```

4. Modify the Python script in part 3 to verify your answer in part 2.

Task 3: Gram-Schmidt Orthogonalization (20 Marks)

The Gram-Schmidt Orthogonalization is a procedure which takes a non-orthogonal set of linearly independent functions and constructs an orthogonal basis. We often normalize these basis to generate orthonormal basis.

1. Consider the vector set: $\boldsymbol{g_1} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}$ $\boldsymbol{g_2} = \begin{pmatrix} -2 \\ 1 \end{pmatrix}$.

   - Visualize these vectors using the Python code given in Task 1.
   - Find the orthonormal basis using the Gram-Schmidt process.
   - Visualize the orthonormal basis.
   - Use the following Python code to verify your answers to Gram-Schmidt Orthonormalization.

```
a1 = np.array([4,3])
a2 = np.array([-2,1])

def normalize(v):
    return v / np.sqrt(v.dot(v))

V = [a1,a2]
orthonorm_vectors = V
orthonorm_vectors[0] = normalize(V[0])
for i in range(1,len(V)):
    for j in range(0, i):
        orthonorm_vectors[i] =
                        normalize(V[i] - (np.dot(V[i],V[j])) * V[j])
```

```
for i in orthonorm_vectors:
    print(i)
```

2. Repeat part 1 for the following set of vectors:

   - $\boldsymbol{h_1} = \begin{pmatrix} 3 \\ 4 \end{pmatrix} \quad \boldsymbol{h_2} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$

Task 4: Finding nearest vector (10 Marks)

In this task, you are required to choose a vector and then find the nearest vector from a given set of vectors. For a vector, the nearest vector to it is the one with the shortest euclidean distance from it. This distance may be computed as the square root of the sum of squares of the difference between the two vectors. Given the following set of vectors:

$\boldsymbol{i_1} = \begin{pmatrix} 6 \\ -4 \end{pmatrix}, \quad \boldsymbol{i_2} = \begin{pmatrix} 3 \\ 2 \end{pmatrix}. \quad \boldsymbol{i_3} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \quad \boldsymbol{i_4} = \begin{pmatrix} 2 \\ -2 \end{pmatrix}.$

1. Visualize the vectors in Python
2. Using Python methods $np.subtract, np.sum, np.sqrt, np.nonzero, np.min, np.where$ or otherwise, find the nearest vector to any vector of your choice.