# Tarefa 1

## Alunos

- Eliézer Zarpelão (RA: 141320)
- Nícolas Assumpção (RA: 121245)

```
from google.colab import drive
drive.mount('/content/drive')

#import urllib.request
#urllib.request.urlretrieve("https://www.ic.unicamp.br/~wainer/cursos/1s2021/431/dados.npy", '
```

```
Mounted at /content/drive
```

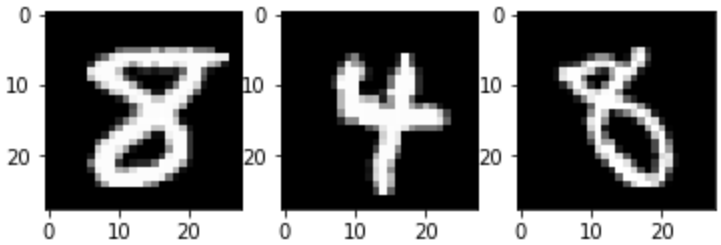
## **1-** Leia o arquivo dados.npy

```
import numpy as np

data = np.load("./drive/MyDrive/MO431/dados.npy")
print(f"Dimensão da matriz de dados: {data.shape}")
```

```
Dimensão da matriz de dados: (10500, 784)
```

## **2-** Imprima a imagem dos 3 primeiros dígitos

```
import matplotlib.pyplot as plt
import cv2

_, subp = plt.subplots(1, 3)
for image in range(3):
  subp[image].imshow(data[image,:].reshape(28,28), 'gray')
```




## **3-** Faça a fatoração SVD da matriz X.

```
# Normalizar a matriz para deixar média = 1
```

```
data_norm = data - np.mean(data)
np.mean(data_norm)
```

```
3.3509002073484214e-14
```

```
u, s, vh = np.linalg.svd(data_norm, full_matrices=False)

u_full = u
d_full = np.zeros(data.shape)
d_full[:s.shape[0],:s.shape[0]] = np.diag(s)

print("SVD full matrix")
print(f"Shape of U: {u_full.shape}")
print(f"Shape of S: {d_full.shape}")
print(f"Shape of V: {vh.shape}\n")

u_compact = u[:, :s.shape[0]]
d_compact = np.diag(s)

print("SVD compact")
print(f"Shape of U: {u_compact.shape}")
print(f"Shape of S: {d_compact.shape}")
print(f"Shape of V: {vh.shape}")
```

```
SVD full matrix
Shape of U: (10500, 784)
Shape of S: (10500, 784)
Shape of V: (784, 784)

SVD compact
Shape of U: (10500, 784)
Shape of S: (784, 784)
Shape of V: (784, 784)
```

## **4-** SVD truncado

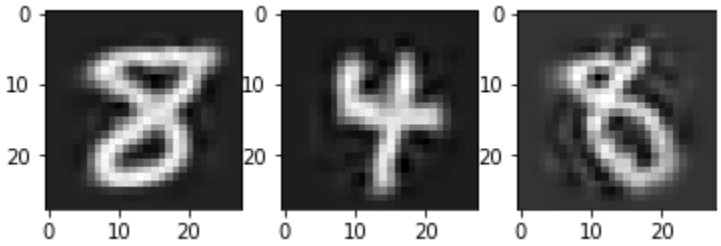
```
# Projection Matrix
ud_proj = np.dot(u_compact[:, :100], d_compact[:100, :100])
print(f"4.1 - Dimensões da matriz projetada: {ud_proj.shape}")

# Reconstructed Matrix
reconstructed_data = np.dot(np.dot(u_compact[:, :100], d_compact[:100, :100]), vh[:100, :])
print(f"4.2 - Dimensões da matriz reconstruída: {reconstructed_data.shape}")
```

```
4.1 - Dimensões da matriz projetada: (10500, 100)
4.2 - Dimensões da matriz reconstruída: (10500, 784)
```

## **4-** Imprima a imagem reconstruída dos 3 primeiros dígitos

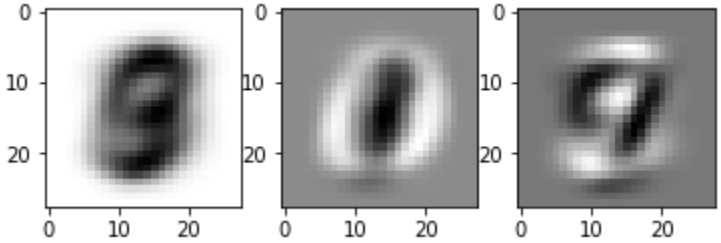
```
_, subp = plt.subplots(1, 3)
```

```
for image in range(3):
  subp[image].imshow(reconstructed_data[image,:].reshape(28,28), 'gray')
```



## **6-** Imprima os 3 primeiros eigen-dígitos

```
# Print the 3 firsts eigen-digits
_, subp = plt.subplots(1, 3)
for image in range(3):
  subp[image].imshow(vh[image,:].reshape(28,28), 'gray')
```



## **7-** Decidindo o número de dimensões

```
print(f"7.1 - Devemos manter {np.sum(s>1)} dimensões para usar a regra de singular values > 1"

v_80 = False
s_sum = np.sum(s)
partial_sum = 0
for i, v in enumerate(s):
  partial_sum += v
  if partial_sum/s_sum > .8 and not v_80:
    v_80 = True
    print(f"7.2 - Devemos manter {i+1} dimensões para capturar 80% da variância")
  if partial_sum/s_sum > .95:
    print(f"7.3 - Devemos manter {i+1} dimensões para capturar 95% da variância")
    break
```

```
7.1 - Devemos manter 672 dimensões para usar a regra de singular values > 1
7.2 - Devemos manter 228 dimensões para capturar 80% da variância
7.3 - Devemos manter 423 dimensões para capturar 95% da variância
```