task

April 12, 2021

1 Tarefa 1: Álgebra Linear e Otimização para ML - MO431A

Universidade Estadual de Campinas (UNICAMP), Instituto de Computação (IC)

Prof. Jacques Wainer, 2021s1

[1]:
```
# RA & Name
print('265673: ' + 'Gabriel Luciano Gomes')
print('192880: ' + 'Lucas Borges Rondon')
print('265674: ' + 'Paulo Júnio Reis Rodrigues')
```

```
265673: Gabriel Luciano Gomes
192880: Lucas Borges Rondon
265674: Paulo Júnio Reis Rodrigues
```

1.1 Imports necessários para a tarefa

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from sklearn.preprocessing import Normalizer
from sklearn.decomposition import PCA
```

1.2 Leitura da base de dados

[3]:
```
X = np.load('db/dados.npy')
```

1.2.1 Impressão das três primeiras imagens da base de dados

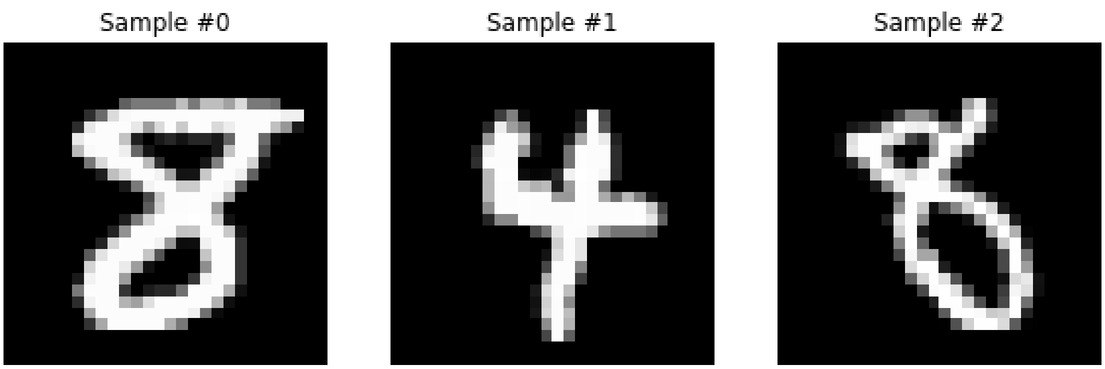
[4]:
```
fig = plt.figure(figsize=(10, 7))

for i in range(3):
    #reshape figure
    img = np.reshape(X[i], (28, 28))

    #positioning figure
    fig.add_subplot(1, 3, i+1)

    # showing image
```

```
    plt.imshow(img)
    plt.axis('off')
    plt.title("Sample #{0}".format(i))
    plt.imshow(img, cmap=cm.gray)
```



1.2.2 Normalização do Conjunto de Dados

[5]:
```
X = Normalizer().transform(X)
```

1.3 Fatoração da Matriz

[6]:
```
## Full Matrix
U, D, Vt = np.linalg.svd(X, full_matrices=True)
# Shape das matrizes
print(f'U  full matriz shape = {U.shape}')  #Autovectors
print(f'D  full matriz shape = {D.shape}')  #Autovalues
print(f'Vt full matriz shape = {Vt.shape}') #Orthogonal


## Compact Matrix
Uc, Dc, Vtc = np.linalg.svd(X, full_matrices=False)
# Shape das matrizes compactas
print(f'U  compact matriz shape = {Uc.shape}')  #Autovectors
print(f'D  compact matriz shape = {Dc.shape}')  #Autovalues
print(f'Vt compact matriz shape = {Vtc.shape}') #Orthogonal
```

```
U  full matriz shape = (10500, 10500)
D  full matriz shape = (784,)
Vt full matriz shape = (784, 784)
U  compact matriz shape = (10500, 784)
D  compact matriz shape = (784,)
Vt compact matriz shape = (784, 784)
```

1.4 Redução de Dimensões

1.4.1 Redução para 100 dimensões

[7]:
```
pca = PCA(n_components=100)
pca.fit(X)
reducedMatrix = pca.transform(X)

# Shape of reduced Matrix
print(f'Reduced Matrix shape: {reducedMatrix.shape}')
```

```
Reduced Matrix shape: (10500, 100)
```

1.4.2 Matriz reconstruída

[8]:
```
reconstruct_matrix = pca.inverse_transform(reducedMatrix)

print(f'Reconstruct matrix shape: {reconstruct_matrix.shape}')
```

```
Reconstruct matrix shape: (10500, 784)
```

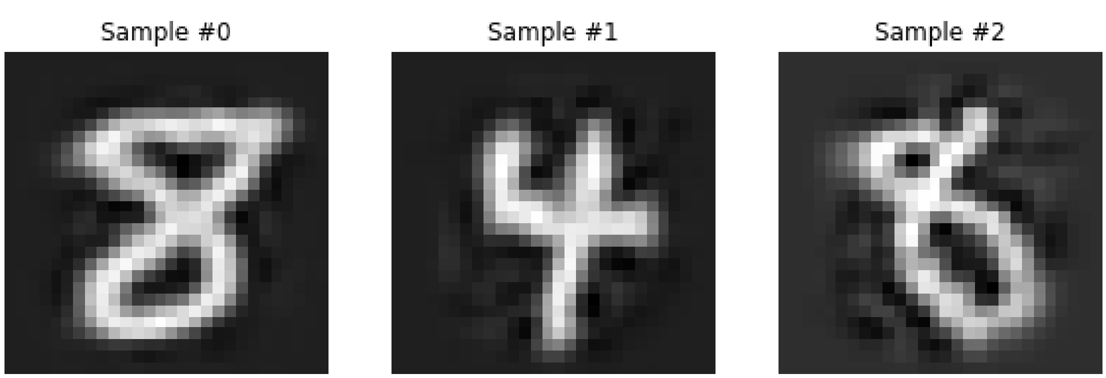
1.5 Impressão das três primeiras imagens reconstruídas

[9]:
```
fig = plt.figure(figsize=(10, 7))

for i in range(3):
    #reshape figure
    img = np.reshape(reconstruct_matrix[i], (28, 28))

    #positioning figure
    fig.add_subplot(1, 3, i+1)

    # showing image
    plt.imshow(img)
    plt.axis('off')
    plt.title("Sample #{0}".format(i))
    plt.imshow(img, cmap=cm.gray)
```



1.6 Impressão dos três primeiros Eigen-dígitos

[10]:
```python
# Compute U, D and Vt matrices for the reduced matrix
U_reduced, D_reduced, Vt_reduced = np.linalg.svd(reducedMatrix,␣
↪full_matrices=False)

print(f'Digit #0: \n{Vt_reduced[0]} \n')
print(f'Digit #1: \n {Vt_reduced[1]} \n')
print(f'Digit #2: \n{Vt_reduced[2]}')
```

```
Digit #0:
[ 1.00000000e+00 -3.39766684e-16  7.19365625e-16  7.35533261e-17
 -2.23864569e-16 -4.88467613e-17 -2.49493038e-17 -7.19728316e-17
  7.91527060e-18 -1.07121253e-16  8.03888728e-17 -9.02664543e-17
  9.07183736e-17  1.08761487e-16 -9.53791070e-17  1.46863699e-16
 -4.85135242e-17  2.95167094e-18  2.93319179e-16 -7.99369643e-17
 -3.22482217e-16  5.99683506e-16  2.39320145e-16 -7.29759155e-16
  2.60092543e-16 -2.67665526e-15 -4.24632868e-16 -1.46713089e-15
  1.07406139e-15  1.18939669e-15  6.76687107e-15  2.93496091e-15
 -1.18524249e-14 -8.72681947e-15 -1.78041176e-15 -6.39395062e-15
  4.83368621e-17 -4.09400267e-15 -5.81523851e-16  1.39129500e-14
  9.52863401e-15 -1.51142546e-14  3.30715826e-14 -4.86003281e-14
 -6.66363671e-15  1.58000109e-14  7.44159390e-15 -1.30957891e-14
  2.23348608e-14  1.27310833e-14  3.54397325e-14  5.81677890e-14
 -6.62394285e-14  2.28348979e-14 -1.05621882e-13 -5.09246788e-14
  7.06129313e-14 -6.58783287e-15 -1.92992521e-13 -2.35853176e-15
 -1.85300246e-14  1.54767499e-13  1.06694869e-13 -2.11910940e-13
  1.65209162e-13 -7.54738020e-14  4.27024770e-14 -5.62444577e-15
 -2.04593568e-13  2.09715565e-15 -3.64378354e-14  3.85785723e-13
  4.21405918e-13 -2.78443830e-13  2.15626406e-13 -1.26727632e-13
 -2.41816259e-13 -5.97212640e-13 -8.30238646e-14  3.82631578e-13
 -1.13942008e-14  9.84799649e-14 -4.94215573e-13 -3.33832715e-13
 -2.47724192e-13 -3.68872384e-13 -4.20199817e-13  6.61275182e-15
 -5.14583760e-13  1.40126356e-12  1.30360375e-12 -1.27109025e-12
  1.64840860e-13 -1.16416151e-12  3.83209101e-13 -2.72757646e-12
  3.34196643e-12 -1.98087903e-12 -9.55438090e-13  5.24595719e-13]

Digit #1:
 [ 3.39766650e-16  1.00000000e+00  4.75487705e-15 -3.33066907e-16
 -1.60982339e-15 -1.66533454e-15  4.16333634e-16  4.16333634e-17
 -1.17961196e-16  4.77048956e-18 -5.96311195e-17  2.32019265e-16
  1.67154159e-15  6.07129500e-16 -3.43457884e-16  7.35045980e-16
 -1.01685820e-15  2.02855490e-15  4.62644120e-15 -2.87492709e-15
  4.45029468e-15  1.10285834e-14 -9.42343177e-16  1.09483584e-14
  6.86451008e-15 -1.10267981e-14  8.54109551e-15 -1.39470853e-14
```

```
  7.71509095e-15  5.14799691e-15  3.98306346e-14  1.45863069e-14
  2.51034167e-14 -1.04328185e-14 -1.93574483e-14  3.83600969e-14
 -1.22869328e-13  6.42110287e-14 -7.53000691e-15 -5.72408325e-14
  6.71686291e-14 -1.46697593e-13  7.64672756e-14  4.92027503e-14
 -1.50967216e-13 -3.20867985e-14  1.39340182e-13 -5.13749301e-13
  1.54991723e-13  5.08014408e-13  7.32296454e-13 -1.36065480e-13
 -7.04602237e-14  2.28275628e-13  6.89483231e-15  1.37377354e-12
  3.86264147e-13 -3.97559904e-13 -2.55714390e-13 -1.99737089e-12
  1.37361572e-12  2.30307978e-12 -1.08994014e-12  1.48081406e-12
 -1.73988109e-12  1.86437981e-13  6.93647711e-13  3.34405985e-12
  5.23485958e-13  5.10522459e-12 -1.87076209e-12  4.14205978e-13
 -6.71241763e-13 -3.42744453e-12  1.84963377e-12  9.50506377e-13
  5.69586467e-12 -3.87738971e-12  2.86295663e-13  2.67824740e-12
 -6.18187669e-13  5.32981775e-12  6.45712006e-12 -4.35401887e-12
 -1.23991371e-11 -8.47051311e-12 -2.17868742e-13 -1.07741227e-11
 -7.85522985e-12  1.52958393e-12 -5.90671976e-12 -8.22387727e-12
 -1.42887629e-11 -5.26009230e-12  3.72480271e-12 -1.96265001e-11
 -1.27712446e-11  5.72887740e-12 -9.08767798e-12  8.08029338e-12]

Digit #2:
[-7.19366327e-16 -4.30607156e-15  1.00000000e+00 -2.44249065e-15
 -5.55111512e-17  1.99840144e-15 -1.49880108e-15 -8.60422844e-16
  3.33066907e-16 -1.71217207e-15 -3.55444840e-15 -8.67361738e-16
  5.55284985e-15  4.42300276e-15 -1.79451723e-15  9.41020401e-15
 -3.53079755e-15 -1.56571770e-14  2.17108765e-14  1.63099476e-14
  8.62206513e-15  3.53012866e-14 -6.30721193e-14  1.00641535e-15
 -1.66230238e-14 -1.29786098e-13  5.50859921e-14 -3.88205657e-14
 -8.98310607e-14  1.31105460e-14  1.01040229e-13  3.17493843e-13
 -3.05125756e-14 -3.05221191e-13 -5.81255175e-13 -2.24680912e-13
 -1.54175136e-13 -2.14069446e-13  5.30476311e-13 -3.57881285e-13
  4.39371814e-13  8.68609896e-14  1.12932713e-12 -1.07117325e-12
 -5.77727075e-13 -2.29351070e-13 -1.97797627e-13  6.73396442e-13
  1.45185480e-12  1.27149387e-12  4.98729139e-12  6.75782030e-13
 -1.35644101e-12  3.13889719e-12 -2.00074755e-12  8.64071493e-12
  4.26548283e-12  1.49979598e-12 -4.70500699e-12 -2.93108912e-12
 -3.14122336e-12  1.76317413e-11  2.09454342e-12  4.12678625e-12
 -2.47494783e-12 -3.69377788e-12 -9.73929970e-12  2.72482705e-12
 -1.24870345e-11  9.96518395e-12  3.64469384e-12  1.13461307e-11
  1.39124338e-11 -2.57881941e-11  6.78255394e-13 -2.99914944e-11
  1.22500010e-11  1.14684681e-11 -1.58120181e-11  4.39831872e-11
  3.10469410e-11 -1.33928067e-12  3.11216602e-12 -7.49532859e-12
 -1.84849253e-11 -3.07102404e-11  2.41615079e-11 -5.55202646e-11
 -7.59800324e-11  2.14910543e-11  1.72268095e-11 -3.34632170e-11
 -4.53636108e-11 -3.79580078e-11 -1.83501834e-11 -1.09429514e-10
  2.18147755e-11 -4.19733914e-11 -4.76409434e-11 -5.78506340e-12]
```

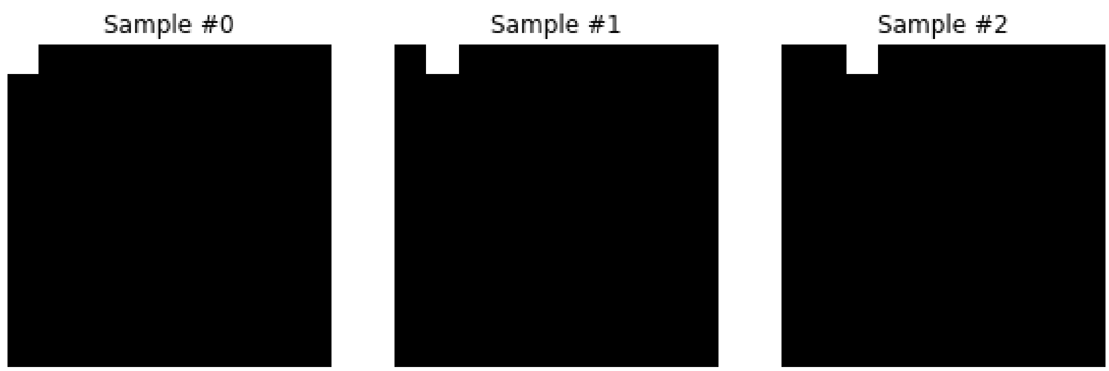
[16]:

```python
## Exemplo dos eigen-digitos Vt reduzido

fig = plt.figure(figsize=(10, 10))
# The eigen values are the bases of the reduced subspace, thus Vt will be used
for i in range(3):
    #reshape figure
    img = np.reshape(Vt_reduced[i], (10, 10))

    #positioning figure
    fig.add_subplot(1, 3, i+1)

    # showing image
    plt.imshow(img)
    plt.axis('off')
    plt.title("Sample #{0}".format(i))
    plt.imshow(img, cmap=cm.gray)
```



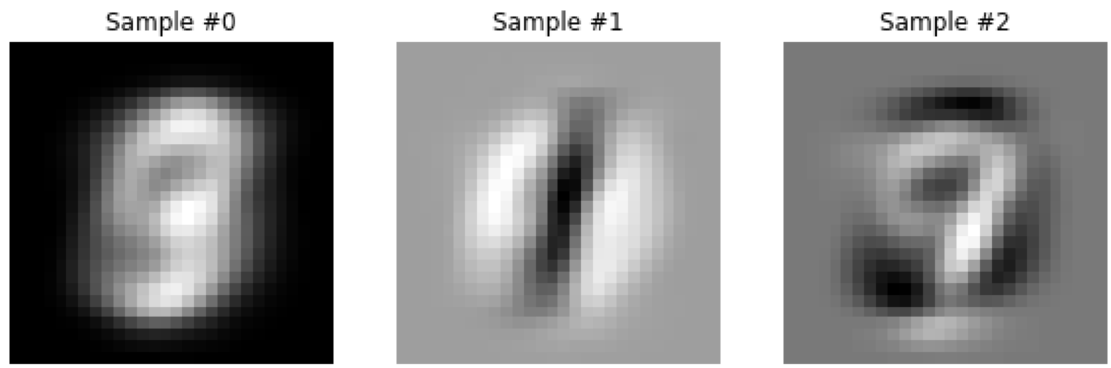
[11]:

```python
## Exemplo dos eigen-digitos

fig = plt.figure(figsize=(10, 10))
# The eigen values are the bases of the reduced subspace, thus Vt will be used
for i in range(3):
    #reshape figure
    img = np.reshape(Vt[i], (28, 28))

    #positioning figure
    fig.add_subplot(1, 3, i+1)

    # showing image
    plt.imshow(img)
    plt.axis('off')
    plt.title("Sample #{0}".format(i))
    plt.imshow(img, cmap=cm.gray)
```



1.7 Decisão do número de dimensões

[12]:
```python
### Quantas dimensões manter, seguindo a regra do singular values > 1

pca = PCA(n_components=271)
pca.fit(X)
reducedMatrix_SV = pca.transform(X)
_, d_sv, _ = np.linalg.svd(reducedMatrix_SV, full_matrices=False)

print(f'Dimension = 271: \n{d_sv} \n')

pca = PCA(n_components=272)
pca.fit(X)
reducedMatrix_SV = pca.transform(X)
_, d_sv, _ = np.linalg.svd(reducedMatrix_SV, full_matrices=False)

print(f'Dimension = 272: \n{d_sv} \n')

pca = PCA(n_components=273)
pca.fit(X)
reducedMatrix_SV = pca.transform(X)
_, d_sv, _ = np.linalg.svd(reducedMatrix_SV, full_matrices=False)

print(f'Dimension = 273: \n{d_sv} \n')
```

```
Dimension = 271:
[24.51434288 21.43744457 19.17619695 17.5770927  17.05931516 16.00231653
 14.00159945 13.47501055 12.59501385 12.16161005 11.44954515 11.02509206
 10.44599499 10.11402987  9.87969893  9.64052497  9.11089116  8.87856643
  8.58417009  8.29615847  8.11199601  8.02389258  7.82169358  7.70312277
  7.5896568   7.47133427  7.08977191  7.04697554  6.93245047  6.86556306
  6.59010844  6.54521171  6.26687166  6.13278505  6.00575661  5.94428514
```

```
  5.85083872  5.77089225  5.68175232  5.64338019  5.48822746  5.36674803
  5.2107998   5.16211478  5.11429388  5.06978195  4.91267062  4.83181728
  4.7912345   4.64802365  4.61708609  4.59911137  4.5125278   4.44330892
  4.38900387  4.31910455  4.25508196  4.2100488   4.14278335  4.11408033
  4.07507434  4.0382842   3.97260353  3.86713814  3.8603942   3.80136052
  3.74986391  3.69529344  3.62987587  3.57312063  3.56705989  3.53389097
  3.47773835  3.45126165  3.42048281  3.37125543  3.29817338  3.25471654
  3.23353396  3.20057022  3.13364573  3.12391179  3.07201823  3.05687513
  3.01020644  2.98779096  2.97924344  2.93812458  2.92035164  2.89140342
  2.82412676  2.8103578   2.78627636  2.75477589  2.73869034  2.72838866
  2.69511272  2.66128797  2.63361899  2.62050201  2.61496476  2.57056982
  2.545864    2.52123743  2.4942682   2.47783845  2.46047834  2.42701819
  2.41688815  2.40056245  2.37061322  2.34525425  2.33594805  2.31749757
  2.31538504  2.28160225  2.25541786  2.24266396  2.22479813  2.20636591
  2.18622324  2.18253542  2.16195214  2.14748606  2.12924754  2.10816137
  2.09640226  2.08068119  2.06848579  2.05066208  2.0386956   2.03045311
  2.01886315  2.0054091   1.99484871  1.97885267  1.96922041  1.96715595
  1.93386031  1.92141624  1.9029976   1.88094155  1.87654833  1.86349675
  1.84902658  1.84564468  1.82704592  1.82025637  1.81054619  1.79452505
  1.78931561  1.77894054  1.77522678  1.76666871  1.75334819  1.74749437
  1.74189711  1.72146287  1.71496292  1.70626669  1.69801383  1.68883087
  1.67448847  1.65007195  1.64828994  1.64378071  1.64205021  1.61910781
  1.6154502   1.60887863  1.59917077  1.59280994  1.57787843  1.56870306
  1.55838459  1.55491302  1.53882717  1.53557607  1.52791207  1.5198672
  1.50613407  1.50370383  1.49176024  1.48594267  1.48385752  1.47944077
  1.46683475  1.46443429  1.4587874   1.45489619  1.44349818  1.43776714
  1.431605    1.42957935  1.42181484  1.41941538  1.41048228  1.40476291
  1.39525872  1.38491062  1.3790581   1.36950959  1.36576925  1.36080135
  1.35448454  1.34667108  1.33480547  1.33344866  1.32452841  1.32415496
  1.31779258  1.31256647  1.30860131  1.30226852  1.30037925  1.29586484
  1.28932506  1.27558582  1.27388497  1.27229399  1.26900207  1.26386424
  1.25993841  1.24739924  1.2437552   1.23676425  1.23411313  1.2258612
  1.22205908  1.21664167  1.21341764  1.20755521  1.19842327  1.19086525
  1.18215686  1.17635242  1.17213863  1.16871379  1.16577561  1.16326657
  1.15858584  1.15085382  1.14902701  1.1398877   1.13958011  1.13284671
  1.12711179  1.12415238  1.12181089  1.11451611  1.10953615  1.10546756
  1.09378855  1.0927559   1.08994834  1.08452146  1.07620642  1.07124646
  1.06814439  1.06626293  1.06181902  1.05769433  1.04473364  1.04030644
  1.0365692   1.0341361   1.03013249  1.02494495  1.0193978   1.00671683
  1.0033831 ]

Dimension = 272:
[24.51434288 21.43744457 19.17619695 17.5770927  17.05931516 16.00231653
 14.00159945 13.47501055 12.59501385 12.16161005 11.44954515 11.02509206
 10.44599499 10.11402987  9.87969893  9.64052497  9.11089116  8.87856643
  8.58417009  8.29615847  8.11199601  8.02389258  7.82169358  7.70312277
  7.5896568   7.47133427  7.08977191  7.04697554  6.93245047  6.86556306
  6.59010844  6.54521171  6.26687166  6.13278505  6.00575661  5.94428514
```

```
  5.85083872  5.77089225  5.68175232  5.64338019  5.48822746  5.36674803
  5.2107998   5.16211478  5.11429388  5.06978195  4.91267062  4.83181728
  4.7912345   4.64802365  4.61708609  4.59911137  4.5125278   4.44330892
  4.38900387  4.31910455  4.25508196  4.2100488   4.14278335  4.11408033
  4.07507434  4.0382842   3.97260353  3.86713814  3.8603942   3.80136052
  3.74986391  3.69529344  3.62987587  3.57312063  3.56705989  3.53389097
  3.47773835  3.45126165  3.42048281  3.37125543  3.29817338  3.25471654
  3.23353396  3.20057022  3.13364573  3.12391179  3.07201823  3.05687513
  3.01020644  2.98779096  2.97924344  2.93812458  2.92035164  2.89140342
  2.82412676  2.8103578   2.78627636  2.75477589  2.73869034  2.72838866
  2.69511272  2.66128797  2.63361899  2.62050201  2.61496476  2.57056982
  2.545864    2.52123743  2.4942682   2.47783845  2.46047833  2.42701821
  2.41688817  2.40056244  2.37061321  2.34525425  2.33594807  2.31749759
  2.31538505  2.28160228  2.25541784  2.24266392  2.22479807  2.20636589
  2.18622318  2.18253535  2.16195211  2.14748609  2.12924759  2.10816119
  2.09640227  2.08068141  2.0684859   2.0506617   2.03869541  2.03045343
  2.01886348  2.0054092   1.99484876  1.97885198  1.96922105  1.96715584
  1.93385976  1.9214165   1.90299588  1.88094317  1.87654886  1.8634994
  1.84902591  1.8456441   1.82704679  1.82025619  1.81054526  1.79452379
  1.78931541  1.77893887  1.77523066  1.76667499  1.75334103  1.74749403
  1.74189714  1.72146961  1.71495555  1.70626582  1.69802276  1.68883266
  1.67448977  1.65007899  1.64829189  1.64378718  1.64206027  1.6190976
  1.61541115  1.60886848  1.59920765  1.59283555  1.577844    1.56870367
  1.55841067  1.55492063  1.5388697   1.53553648  1.52790422  1.51981617
  1.50620556  1.5036605   1.49177902  1.48596352  1.48395468  1.47951871
  1.46679199  1.46456231  1.45860902  1.45486565  1.44349367  1.43787904
  1.43168291  1.42948687  1.4218755   1.41947699  1.41052895  1.40488041
  1.39537128  1.38494827  1.3791019   1.36959596  1.36634214  1.36072284
  1.35453078  1.34624447  1.33479582  1.33365341  1.32544778  1.3244731
  1.31781346  1.31342374  1.30856135  1.30220973  1.29996992  1.29593092
  1.28880534  1.27725344  1.27396916  1.27229331  1.2693653   1.26433496
  1.26018595  1.24732898  1.24430564  1.23817345  1.23450392  1.22603372
  1.22280176  1.21658697  1.21377325  1.20559541  1.19768977  1.19371966
  1.18625921  1.17672617  1.17281898  1.16793797  1.1674316   1.1616023
  1.15507892  1.15104933  1.14277684  1.140533    1.13712388  1.13471481
  1.12600231  1.12021134  1.11462284  1.11259029  1.10870654  1.10476673
  1.09977687  1.09446622  1.08929485  1.08405215  1.08254822  1.07727403
  1.06821899  1.06702221  1.05986795  1.05795899  1.05434908  1.04400614
  1.03501698  1.03228585  1.03018746  1.02672332  1.01457157  1.0096156
  1.00231384  1.0004503 ]
```

```
Dimension = 273:
[24.51434288 21.43744457 19.17619695 17.5770927  17.05931516 16.00231653
 14.00159945 13.47501055 12.59501385 12.16161005 11.44954515 11.02509206
 10.44599499 10.11402987  9.87969893  9.64052497  9.11089116  8.87856643
  8.58417009  8.29615847  8.11199601  8.02389258  7.82169358  7.70312277
  7.5896568   7.47133427  7.08977191  7.04697554  6.93245047  6.86556306
  6.59010844  6.54521171  6.26687166  6.13278505  6.00575661  5.94428514
```

```
5.85083872  5.77089225  5.68175232  5.64338019  5.48822746  5.36674803
5.2107998   5.16211478  5.11429388  5.06978195  4.91267062  4.83181728
4.7912345   4.64802365  4.61708609  4.59911137  4.5125278   4.44330892
4.38900387  4.31910455  4.25508196  4.2100488   4.14278335  4.11408033
4.07507434  4.0382842   3.97260353  3.86713814  3.8603942   3.80136052
3.74986391  3.69529344  3.62987587  3.57312063  3.56705989  3.53389097
3.47773835  3.45126165  3.42048281  3.37125543  3.29817338  3.25471654
3.23353396  3.20057022  3.13364573  3.12391179  3.07201823  3.05687513
3.01020644  2.98779096  2.97924344  2.93812458  2.92035164  2.89140342
2.82412676  2.8103578   2.78627636  2.75477589  2.73869034  2.72838866
2.69511272  2.66128797  2.63361899  2.62050201  2.61496476  2.57056982
2.54586401  2.52123743  2.4942682   2.47783845  2.46047834  2.42701821
2.41688816  2.40056245  2.37061323  2.34525425  2.33594809  2.31749758
2.31538505  2.28160224  2.25541785  2.24266393  2.22479809  2.20636588
2.1862232   2.18253539  2.16195213  2.14748596  2.12924751  2.10816137
2.09640231  2.0806814   2.06848575  2.05066206  2.0386956   2.03045305
2.0188636   2.00540925  1.99484851  1.97885211  1.96922109  1.96715536
1.93386012  1.92141706  1.90299707  1.880943    1.87654812  1.86349767
1.8490271   1.84564522  1.82704709  1.82025462  1.81054504  1.7945242
1.78931604  1.77894096  1.77522948  1.76667275  1.75335039  1.74749429
1.74189737  1.72147548  1.71496682  1.7062722   1.69802808  1.68882468
1.67448462  1.65008579  1.64827872  1.64378768  1.64206024  1.61908037
1.615456    1.60886149  1.59919258  1.59284768  1.57787545  1.56868386
1.55839186  1.55493153  1.53884047  1.53553477  1.52789977  1.51981479
1.50611933  1.50371492  1.4918263   1.48597322  1.48393615  1.47951383
1.46679063  1.4645669   1.458772    1.45471276  1.44350736  1.43776823
1.4315854   1.42963407  1.42180897  1.41967546  1.41040352  1.40489959
1.39547396  1.3847551   1.37910426  1.3693187   1.36628753  1.36034943
1.35465008  1.34659345  1.334973    1.3338194   1.32553069  1.32457072
1.31803666  1.31298321  1.3095333   1.30227679  1.30001966  1.29589355
1.288374    1.27706179  1.27354313  1.27285109  1.26908929  1.26499692
1.26083172  1.24755635  1.24335739  1.23927429  1.2340761   1.22618291
1.22241706  1.2187916   1.2121575   1.20725764  1.19918619  1.19528259
1.1871018   1.17915205  1.17261232  1.16684032  1.16544924  1.1608401
1.15747176  1.15157311  1.14945166  1.14445721  1.14076006  1.13543797
1.1296425   1.12274899  1.12054438  1.11585458  1.10690411  1.1056521
1.0996559   1.09411336  1.09044135  1.08711449  1.07893526  1.0732773
1.06886685  1.06591299  1.06441055  1.05616473  1.05329581  1.04482647
1.04095526  1.03516311  1.0277982   1.02597015  1.01504247  1.01432455
1.00359351  0.99791314  0.99306401]
```

1.7.1 Conclusão

Seguindo a regra de singular values > 1, deve-se manter 272 dimensões

[13]:
```
### Capturar 80% da variância
```

```python
# calcular variância da matriz compacta
sd_value = Dc.sum() * (8/10) # 80% do valor
print(f'80% of value: {sd_value}')

r_list = [50, 75, 125, 150, 200, 250, 275, 276, 300]

for r in r_list:
    pca = PCA(n_components=r)
    pca.fit(X)
    reducedMatrix_SD = pca.transform(X)
    _, d_sd, _ = np.linalg.svd(reducedMatrix_SD, full_matrices=False)
    print(f'Sum of D = {d_sd.sum()} \t for r = [{r}]')

print('Best value of r = 276')
```

```
80% of value: 899.2620814269975
Sum of D = 451.05904369035204    for r = [50]
Sum of D = 550.3992344229432     for r = [75]
Sum of D = 682.2646361552922     for r = [125]
Sum of D = 731.0378384900312     for r = [150]
Sum of D = 809.4309337755165     for r = [200]
Sum of D = 871.6934374729149     for r = [250]
Sum of D = 898.4175028599229     for r = [275]
Sum of D = 899.3764217283363     for r = [276]
Sum of D = 922.722794346364      for r = [300]
Best value of r = 276
```

1.7.2 Conclusão

Seguindo a regra de capturar 80% da variância, deve-se manter 276 dimensões

[14]:
```python
### Capturar 95% da variância

# calcular variância da matriz compacta
sd_value = Dc.sum() * (95/100) # 95% do valor
print(f'95% of value: {sd_value}')

r_list = [500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750]

for r in r_list:
    pca = PCA(n_components=r)
    pca.fit(X)
    reducedMatrix_SD = pca.transform(X)
    _, d_sd, _ = np.linalg.svd(reducedMatrix_SD, full_matrices=False)
    print(f'Sum of D = {d_sd.sum()} \t for r = [{r}]')

print('Best value of r = 675')
```

```
95% of value: 1067.8737216945594
Sum of D = 1043.803764210441     for r = [500]
Sum of D = 1050.6158148615561    for r = [525]
Sum of D = 1056.082571101441     for r = [550]
Sum of D = 1060.0963342652944    for r = [575]
Sum of D = 1063.0339158642344    for r = [600]
Sum of D = 1065.0181081493117    for r = [625]
Sum of D = 1066.0698635258873    for r = [650]
Sum of D = 1066.346406008531     for r = [675]
Sum of D = 1066.3464060085312    for r = [700]
Sum of D = 1066.3464060085314    for r = [725]
Sum of D = 1066.3464060085312    for r = [750]
Best value of r = 675
```

1.7.3 Conclusão

Seguindo a regra de capturar 95% da variância, deve-se manter 675 dimensões