SIT720 Assignment One

Name: Jose Arturo Gil Alonso

Student ID: 218 659 676

PART 1: CLUSTERING

In [77]:

```
# 1.1

#Importing the library
import numpy as np
import pandas as pd
from sklearn.cluster import KMeans
from sklearn import metrics
import sklearn as sk
from sklearn import preprocessing
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import os
%matplotlib inline
```

In [78]:

```
os.getcwd()
```

Out[78]:

```
'C:\\Users\\josea\\Desktop\\Python'
```

In [79]:

```
SID=218659676
fID=SID%5
print(fID)
```

```
1
```

In [80]:

```
# File reading

dt1=pd.read_csv('digitData1.csv',delimiter=",",header=None).values
print(dt1.shape)
print(dt1)
dts1=pd.read_csv('digitData1.csv',delimiter=",",header=None)
```

```
(1669, 65)
[[0. 0. 5. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 1.]
 [0. 0. 0. ... 9. 0. 2.]
 ...
 [0. 0. 1. ... 0. 0. 8.]
 [0. 0. 8. ... 0. 0. 0.]
 [0. 0. 4. ... 8. 0. 1.]]
```

In [81]:

```
# Creating an empty array
import numpy as np
```

```
import numpy as np
M = np.loadtxt(open("digitData1.csv", "rb"), delimiter=",")
X = np.empty((M.shape[0], M.shape[1]-1 ))
X = M[:,:-1]
trueLabels = M[:,-1]
print(M.shape)
print(X.shape)
print(trueLabels.shape)
```

```
(1669, 65)
(1669, 64)
(1669,)
```

In [82]:

```
# 1.2 K-means clustering with 5 clusters using Euclidean distance

from sklearn.cluster import KMeans
from sklearn import metrics
import sklearn as sk
adjARI=[]
adjMI=[]

for i in range (50):
    kmeans = KMeans(n_clusters=5, init='random', n_init=50)
    kmeans.fit(X)
    labelsEuc = kmeans.labels_
    ARI=metrics.adjusted_rand_score(trueLabels, labelsEuc)
    adjARI.append(ARI)

    MI=sk.metrics.adjusted_mutual_info_score(trueLabels, labelsEuc, average_method='arithmetic' )
    adjMI.append(MI)

print(adjARI)
```

```
[0.3557459918001672, 0.387942285780558, 0.38481646315181034, 0.38960319135349525,
0.3557459918001672, 0.3852347492146593, 0.35503971928708844, 0.353678995814719, 0.387238255657014,
0.38490539789925426, 0.35512178329542465, 0.38818866576408173, 0.38481646315181034,
0.353678995814719, 0.38420188424099594, 0.3878356288432527, 0.35503971928708844,
0.3885396736721546, 0.38889889938269745, 0.35512178329542465, 0.3878356288432527,
0.387942285780558, 0.3837585078875056, 0.387238255657014, 0.3842983185270869, 0.3885396736721546,
0.3850388661823886, 0.38960319135349525, 0.3511155237881957, 0.3837585078875056,
0.3874661189542459, 0.38889889938269745, 0.3886859928254798, 0.3850388661823886,
0.35503971928708844, 0.38652684191374703, 0.35512178329542465, 0.38420188424099594,
0.353678995814719, 0.3837585078875056, 0.35512178329542465, 0.38960319135349525,
0.38889889938269745, 0.3839804275817222, 0.38654173619808246, 0.35503971928708844,
0.35512178329542465, 0.3517937934870084, 0.3837585078875056, 0.3837585078875056]
```

In [83]:

```
# Average ARI over 50 repetations


aveadjARI=np.mean(np.array(adjARI))
aveadjMI=np.mean(np.array(adjMI))

print(aveadjARI)
print(aveadjMI)
```

```
0.3762511851505198
0.5659908875071735
```

In [84]:

```
# 1.3 Average ARI over 20 repetations

for i in range (20):
    Kmeans = KMeans(n_clusters=5, n_init=20)
    kmeans.fit(X)
    labelsEuc = kmeans.labels_
    ARI=metrics.adjusted_rand_score(trueLabels,labelsEuc)
    adjARI.append(ARI)
print(adjARI)
```

```python
print(adjARI)
aveadjARI=np.mean(np.array(adjARI))
print('The average of 20 init',aveadjARI)
```

```
[0.3557459918001672, 0.387942285780558, 0.38481646315181034, 0.38960319135349525,
0.3557459918001672, 0.3852347492146593, 0.35503971928708844, 0.353678995814719, 0.387238255657014,
0.38490539789925426, 0.35512178329542465, 0.38818866576408173, 0.38481646315181034,
0.353678995814719, 0.38420188424099594, 0.3878356288432527, 0.35503971928708844,
0.3885396736721546, 0.38889889938269745, 0.35512178329542465, 0.3878356288432527,
0.387942285780558, 0.3837585078875056, 0.387238255657014, 0.3842983185270869, 0.3885396736721546,
0.3850388661823886, 0.38960319135349525, 0.3511155237881957, 0.3837585078875056,
0.3874661189542459, 0.38889889938269745, 0.3886859928254798, 0.3850388661823886,
0.35503971928708844, 0.38652684191374703, 0.35512178329542465, 0.38420188424099594,
0.353678995814719, 0.3837585078875056, 0.35512178329542465, 0.38960319135349525,
0.38889889938269745, 0.3839804275817222, 0.38654173619808246, 0.35503971928708844,
0.35512178329542465, 0.3517937934870084, 0.3837585078875056, 0.3837585078875056,
0.387238255657014, 0.3518860694483161, 0.38652684191374703, 0.35503971928708844,
0.387238255657014, 0.3857327393080279, 0.3885396736721546, 0.3557459918001672,
0.35503971928708844, 0.387942285780558, 0.38628288954432133, 0.38960319135349525,
0.3837585078875056, 0.3851327676826454, 0.3850388661823886, 0.38362407778351565,
0.387942285780558, 0.38960319135349525, 0.3885396736721546, 0.38420188424099594]
The average of 20 init 0.3773888020688319
```

As it is known, kmeans++ allocates one centre point randomly and then look for the centres given the first one. Thus, when there is a random starting number of centroids, the objective function decreases with each iteration of the algorithm.

If there is an ARI value of 0.7 after a single run of k-means clustering, as it can be seen in the results, those show 0.38 and 0.57 after 20 repetations, which are lower or closer than the 0.7 of a single run, therefore, the previous explains that after more repetations or iterations the result is better as the cluster centers are not very spread in the feauture space

In [85]:

```python
# 1.4

from sklearn import preprocessing

xnorm = preprocessing.normalize(X,norm='l2')

adjARIcos= []
adjMIcos= []

for i in range (50):
    kmeans= KMeans(n_clusters=5, init='random',n_init=50)
    kmeans.fit(xnorm)
    predictedLabels =kmeans.labels_
    ARI_cosine= metrics.adjusted_rand_score(trueLabels,predictedLabels)
    adjARIcos.append(ARI_cosine)
    MI_cosine=
sk.metrics.adjusted_mutual_info_score(trueLabels,predictedLabels,average_method='arithmetic')
    adjMIcos.append(MI_cosine)

print(adjARIcos)
```

```
[0.35129582971688783, 0.3565874833797658, 0.35209143529093934, 0.35102748960455676,
0.3511998783894273, 0.35209143529093934, 0.35150828894767877, 0.35172374080232205,
0.35162161009687504, 0.35229730746220905, 0.3511998783894273, 0.35229730746220905,
0.35162161009687504, 0.35209143529093934, 0.35084841979515463, 0.3511998783894273,
0.35229730746220905, 0.35209143529093934, 0.3565874833797658, 0.35162161009687504,
0.35209143529093934, 0.3514090308581345, 0.35172374080232205, 0.35229730746220905,
0.3511998783894273, 0.3515913909883225, 0.35162161009687504, 0.3515913909883225,
0.35172374080232205, 0.3511998783894273, 0.35183455473787767, 0.35209143529093934,
0.35209143529093934, 0.3511998783894273, 0.35276641260710917, 0.35209143529093934,
0.3511998783894273, 0.35229730746220905, 0.3515913909883225, 0.35162161009687504,
0.3517411054931323, 0.35229730746220905, 0.3511998783894273, 0.35209143529093934,
0.35209143529093934, 0.35209143529093934, 0.35209143529093934, 0.35150799767675595,
0.35229730746220905, 0.35209143529093934]
```

In [86]:

```python
aveadjARIcos=np.mean(np.array(adjARIcos))
aveadjMIcos=np.mean(np.array(adjMIcos))
print('The clustering performance after 50 initializations using Cosine is',aveadjARIcos)
print(aveadjMIcos)
```

```
The clustering performance after 50 initializations using Cosine is 0.3519607276037244
0.5341150638067561
```

It is recomended to use Euclidean Distance than Cosine for this dataset as the ARI of Euclidean Distance is greater than the Cosine distance. Therefore, as it can be seen, the Cosine distance is 0.35, which is lower than 0.38 that was previously obtained

PART 2: DIMENSIONALITY REDUCTION USING PCA/SVD:

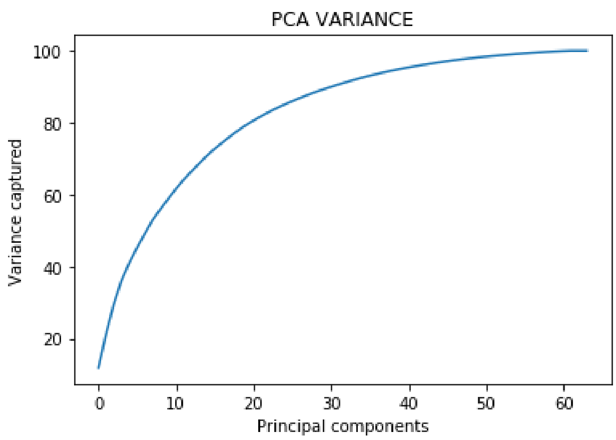
In [87]:

```
# 2.1

from sklearn.preprocessing import scale
scaler = StandardScaler()
scaler.fit(dts1)
X_scaled =scaler.transform(dts1)

from sklearn.decomposition import PCA
PCA = PCA(n_components=64)
PCA.fit(X_scaled)

vr1=np.cumsum(np.round(PCA.explained_variance_ratio_, decimals=4)*100)
print(vr1)
plt.plot(vr1)
plt.xlabel("Principal components")
plt.ylabel("Variance captured")
plt.title("PCA VARIANCE")
```

```
[ 11.79  21.31  29.65  36.22  41.04  45.28  49.2   52.96  55.93  58.78
  61.57  64.14  66.42  68.66  70.82  72.73  74.47  76.19  77.79  79.27
  80.57  81.82  82.99  84.06  85.05  85.98  86.87  87.71  88.53  89.25
  89.95  90.64  91.28  91.91  92.51  93.07  93.58  94.08  94.54  94.97
  95.37  95.75  96.11  96.46  96.79  97.1   97.39  97.66  97.92  98.16
  98.38  98.58  98.77  98.95  99.13  99.29  99.44  99.58  99.71  99.83
  99.93 100.01 100.01 100.01]
```

Out[87]:

```
Text(0.5, 1.0, 'PCA VARIANCE')
```



The minimum dimension that captures at least 95% variance is 41

In [88]:

```
# 2.2

import matplotlib.cm as cm
import matplotlib.lines
from matplotlib.legend import Legend
finalPCA = PCA.fit_transform(X)
```

```
matplotlib.pyplot.figure(num=None,figsize=(8,8),dpi=80)
colors=cm.rainbow(np.linspace(0,1,len(X)))
scatter=plt.scatter(finalPCA[:,0],finalPCA[:,1],c=trueLabels,edgecolor='none',alpha=1,
      cmap=plt.cm.get_cmap('nipy_spectral',10))
t=np.unique(trueLabels)
plotting=[matplotlib.pyplot.Line2D([],[],marker="o",ls="",color=scatter.cmap(scatter.norm(X)))for X
in t]
plt.legend(plotting,t)
plt.xlabel('V1')
plt.ylabel('V2')
```

Out[88]:

```
Text(0, 0.5, 'V2')
```