dm-1-assignment-4

December 16, 2023

[1]:
```
import os
import numpy as np, pandas as pd
import keras

import torch
from torch import nn, optim
from torchvision import datasets, models, transforms

from sklearn.cluster import KMeans
from sklearn.cluster import BisectingKMeans
from sklearn.cluster import SpectralClustering
from sklearn.cluster import DBSCAN
from sklearn.cluster import AgglomerativeClustering
from sklearn.metrics import fowlkes_mallows_score
from sklearn.metrics import silhouette_score
```

[3]:
```
directory = '/content/drive/MyDrive/Reena_Dataset'
```

[180]:
```
y = []
idx = 0
classes = os.listdir(directory)
for i in classes:
  folders = os.listdir(os.path.join(directory,i))
  for j in folders:
    y.append(idx)
  idx+=1
y = np.array(y)
```

1

a) Resize

[9]:
```
def resize(data):
  datatransforms = transforms.Compose([
                              transforms.Resize(224),
                              transforms.ToTensor(),
                             ])
  dataset = datasets.ImageFolder(data, transform=datatransforms)
```

```
  dataiter  = torch.utils.data.DataLoader(dataset, batch_size=1, shuffle =␣
↪False)
  return dataiter
```

[11]:
```
data_resize = resize(directory)
```

b) Normalize

[14]:
```
def normalize(data):
  mean = torch.mean(data)
  std = torch.std(data)
  tensor = (data-mean)/std
  return tensor
```

[15]:
```
data_normalize = []
for i in data_resize:
  data_normalize.append(normalize(i[0]))
```

c) Extract Features

[29]:
```
model = models.resnet18()
model = torch.nn.Sequential(*(list(model.children())[:-1]))
def get_features(data):
  with torch.no_grad():
    feat = model(data)
  return feat
```

[30]:
```
extracted_features = []
for i in data_normalize:
  extracted_features.append(get_features(i))
```

2. Dimensionality Reduction

[38]:
```
def dim_redn(data):
  red_feat = torch.nn.functional.adaptive_avg_pool2d(data, (1, 1))
  red_feat = red_feat.view(512)
  red_feat = np.array(red_feat)
  return red_feat
```

[39]:
```
features_2d = []
for i in extracted_features:
  features_2d.append(dim_redn(i))
features_2d = np.array(features_2d)
features_2d.shape
```

[39]:
```
(659, 512)
```

3. Clustering Algorithm

a

[40]:
```
kmeans_random = KMeans(n_clusters=4, init="random", n_init = 'auto').
  ↪fit(features_2d)
```

b

[41]:
```
kmeans_plus = KMeans(n_clusters=4, init="k-means++", n_init = 'auto').
  ↪fit(features_2d)
```

c

[42]:
```
bisect_kmeans = BisectingKMeans(n_clusters=4,init='random').fit(features_2d)
```

d

[43]:
```
spectral_cluster = SpectralClustering(n_clusters=4).fit(features_2d)
```

DBSCAN

[165]:
```
dbscan = DBSCAN(eps=1.72, min_samples=1).fit(features_2d)
labels = db.labels_
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print("Estimated number of clusters: %d" % n_clusters_)
print("Estimated number of noise points: %d" % n_noise_)
```

```
Estimated number of clusters: 4
Estimated number of noise points: 0
```

To get 4 clusters i used eps=1.72 and min_samples = 1

Agglomerative clustering

Single link

[161]:
```
single = AgglomerativeClustering(n_clusters=4, linkage = 'single').
  ↪fit(features_2d)
```

Complete link

[162]:
```
complete = AgglomerativeClustering(n_clusters=4, linkage = 'complete').
  ↪fit(features_2d)
```

Group Average

[163]:
```
grp_avg = AgglomerativeClustering(n_clusters=4, linkage = 'average').
  ↪fit(features_2d)
```

Ward's method

[164]:
```
ward = AgglomerativeClustering(n_clusters=4, linkage = 'ward').fit(features_2d)
```

4. Clustering Evaluations

[166]:
```
models =␣
↪[kmeans_random,kmeans_plus,bisect_kmeans,spectral_cluster,dbscan,single,complete,grp_avg,wa
```

Fowlkes mallows score

[190]:
```
for i in models:
    pred = i.fit_predict(features_2d)
    fms = fowlkes_mallows_score(y,pred)
    print(f"Model : {i}            fowlkes_mallows_score : {fms}")
```

```
Model : KMeans(init='random', n_clusters=4, n_init='auto')
fowlkes_mallows_score : 0.2874917442183226
Model : KMeans(n_clusters=4, n_init='auto')             fowlkes_mallows_score :
0.287288217443725
Model : BisectingKMeans(n_clusters=4)             fowlkes_mallows_score :
0.27126030250816296
Model : SpectralClustering(n_clusters=4)             fowlkes_mallows_score :
0.2724794013028717
Model : DBSCAN(eps=1.72, min_samples=1)             fowlkes_mallows_score :
0.49818585278832117
Model : AgglomerativeClustering(linkage='single', n_clusters=4)
fowlkes_mallows_score : 0.49818585278832117
Model : AgglomerativeClustering(linkage='complete', n_clusters=4)
fowlkes_mallows_score : 0.46306862249956965
Model : AgglomerativeClustering(linkage='average', n_clusters=4)
fowlkes_mallows_score : 0.48608862674196596
Model : AgglomerativeClustering(n_clusters=4)             fowlkes_mallows_score
: 0.29428673737342675
```

Ranking: 1. AgglomerativeClustering - Single 2. DBSCAN 3. AgglomerativeClustering - Average 4. AgglomerativeClustering - complete 5. AgglomerativeClustering - ward 6. kMeans - random 7. Kmeans - kmeans++ 8. SpectralClustering 9. BisectingKMeans

Silhouette score

[191]:
```
y = np.array(y).reshape(-1,1)
for i in models:
    pred = i.fit_predict(features_2d)
    ss = silhouette_score(y,pred)
    print(f"Model : {i}            silhouette_score : {ss}")
```

```
Model : KMeans(init='random', n_clusters=4, n_init='auto')
silhouette_score : -0.06521150411431226
Model : KMeans(n_clusters=4, n_init='auto')             silhouette_score :
-0.0948270657557989
Model : BisectingKMeans(n_clusters=4)             silhouette_score :
```

```
-0.09643550697279615
Model : SpectralClustering(n_clusters=4)             silhouette_score :
-0.07716534096345021
Model : DBSCAN(eps=1.72, min_samples=1)             silhouette_score :
-0.510595391115971
Model : AgglomerativeClustering(linkage='single', n_clusters=4)
silhouette_score : -0.510595391115971
Model : AgglomerativeClustering(linkage='complete', n_clusters=4)
silhouette_score : -0.07697784500838915
Model : AgglomerativeClustering(linkage='average', n_clusters=4)
silhouette_score : -0.5010831924155199
Model : AgglomerativeClustering(n_clusters=4)             silhouette_score :
-0.08293079765902359
```

Ranking: 1. kMeans - random 2. AgglomerativeClustering - complete 3. SpectralClustering 4. AgglomerativeClustering - ward 5. Kmeans - kmeans++ 6. BisectingKMeans 7. AgglomerativeClustering - Average 8. AgglomerativeClustering - Single 9. DBSCAN