19CS10071 (Anurat Bhattacharya)

Q10)

	k=4	k=8	k=12
Linear Algebra	0	7	7
Data Science	0	0	2
Artificial Intelligence	0	0	2
European Central Bank	2	3	10
Financial Bank	1	3	9
International Monetary Fund	2	3	9
Basket Ball	3	5	1
Swimming	3	4	1
Cricket	3	1	0

Here k=4 seems to be having the best performance

This is so because in the given document list, we have roughly 3 clusters. So for k=8 and 12 over fitting occurs. Ideally, for say k=9, we can Each class assigned to its own unique cluster. This would be grossly overfitted. So for k=4 we have best result

doc_clustering

September 14, 2021

[ ]:

```
import numpy as np
```

[ ]:

```
import pandas as pd
import tqdm
import wikipedia as wiki
articles = ['Linear algebra','Data Science','Artificial intelligence','European␣
 ↪Central Bank',
'Financial technology',
'International Monetary Fund',
'Basketball',
'Swimming',
'Cricket']
wiki_lst = []
title = []
for article in articles:
  wiki_lst.append(wiki.page(article).content)
  title.append(article)
wiki_lst
```

[ ]:

```
from sklearn.feature_extraction.text  import TfidfVectorizer
vectorizer = TfidfVectorizer(stop_words = ('english'))
X = vectorizer.fit_transform(wiki_lst).toarray()
```

[ ]:

```
class KMeans:
  def __init__(self,nCls,Xt,**kwargs):
    self.k = nCls #number of clusters
    self.X = Xt
    self.n = len(self.X)
    self.c = np.zeros(self.X.shape[0],dtype=int)
    self.Z = np.random.rand(nCls,self.X.shape[1])#initialize with random values
    if('seed' in kwargs): self.Z = kwargs['seed']
    #print(self.Z)
    #print(self.X)
    self.Jclus = 0
    self.frq  = np.zeros((nCls,1))
    self.reassignCluster()
    self.calcJclus()
    self.refDict = {}
    self.iterations = 0
    #print(self.frq)
  def calcJclus(self):
    self.Jclus = 0
    for i in range(self.n):
      self.Jclus+=np.linalg.norm(self.X[i]-self.Z[self.c[i]])
    self.Jclus/=float(self.n)
  def reassignClusterRepr(self):
    self.Z = np.zeros((self.k,self.X.shape[1]))
    for i in range(self.n):
      self.Z[self.c[i]]=self.Z[self.c[i]]+self.X[i]
    for i in range(self.k):
      if(self.frq[i]!=0):  self.Z[i]/=self.frq[i]

  def reassignCluster(self):
    #print('1 '+self.frq.shape)
    self.frq = np.zeros((self.k,1))
    #print('2 '+self.frq.shape)
    for i in range(self.n):
      self.c[i] = np.argmin(np.linalg.norm(self.Z-self.X[i],axis=1))
      self.frq[self.c[i]]+=1
  def getClusterRepr(self):
    return self.Z

  def clusterId(self,img):
    #img must be in form (28*28) normalised to [0,1]
    return np.argmin(np.linalg.norm(self.Z-img,axis=1))
  def predict(self,img):
    return self.refDict[self.clusterId(img)]
  def fitData(self):
    oldJclus = 0.0
    while oldJclus!=self.Jclus:
      self.reassignCluster()
      self.reassignClusterRepr()
      oldJclus = self.Jclus
      self.calcJclus()
      self.iterations+=1
```

[ ]:

```
print(articles)
for nClus in [4,8,12]:
  model = KMeans(nClus,X,seed = np.array([X[i] for i in np.random.
↪choice(len(X),size=nClus)]))
  model.fitData()
  print("{} Clusters has cluster classes: {}".format(nClus,model.c))
```

```
['Linear algebra', 'Data Science', 'Artificial intelligence', 'European Central
Bank', 'Financial technology', 'International Monetary Fund', 'Basketball',
'Swimming', 'Cricket']
4 Clusters has cluster classes: [0 0 0 2 1 2 3 3 3]
8 Clusters has cluster classes: [7 0 0 3 3 3 5 4 1]
12 Clusters has cluster classes: [ 7  2  2 10  9  9  1  1  0]
```

[ ]: