Name: Deepika Kanade
Assignment 3- EE 511

Q 1)

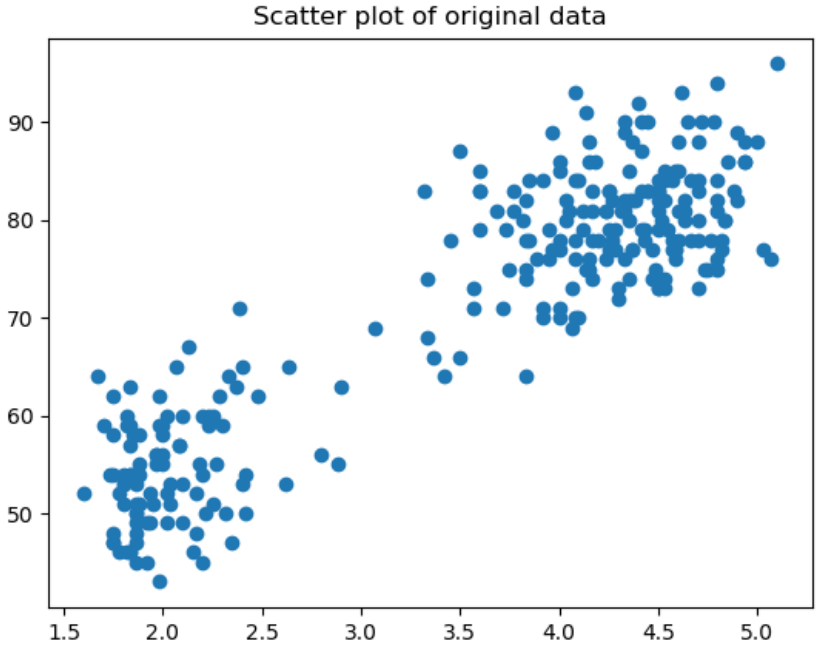
Task: To generate a 2-D scatter plot of the data and run a k-means clustering routine on the data for k=2.

Code:

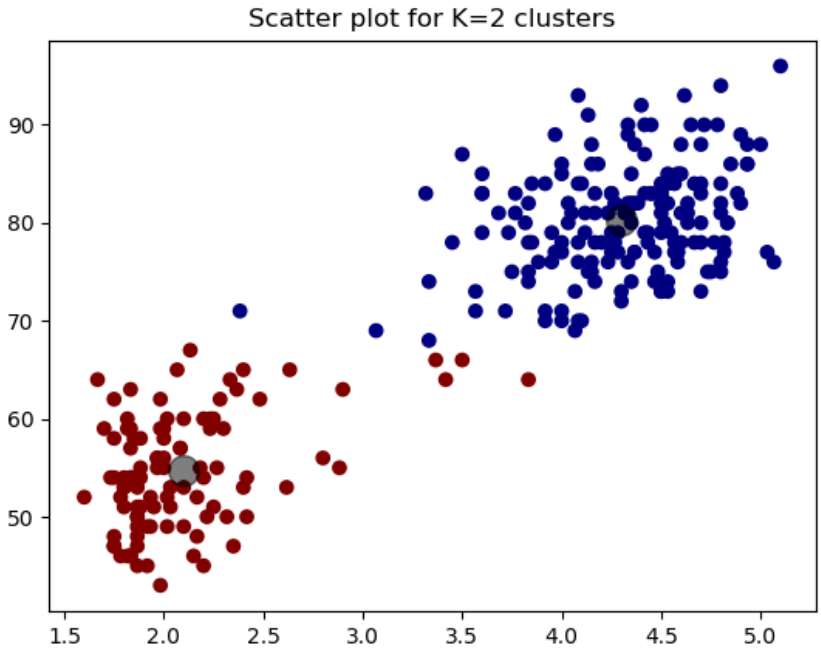
```
"""
Created on Tue Mar 13 22:46:24 2018

@author: deepikakanade
"""

import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

trainFile = '/Users/deepikakanade/Desktop/oldfaithful.csv'
Data = np.genfromtxt(trainFile, delimiter = ',')

plt.figure()
plt.scatter(Data[:,1],Data[:,2])

#K-means algorithm for K=2
kmeans = KMeans(n_clusters=2)
kmeans.fit(Data[:,[1,2]])

plt.figure()
plt.scatter(Data[:,1],Data[:,2],c=kmeans.labels_, cmap='jet')

#plotting the center of the two clusters
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
```

Outputs :-

2D scatter plot of original data



Scatter plot for K means algorithm with K=2 clusters



Observations :-

K-means algorithm is used to cluster the data points in the 'old faithful' dataset. The data consists of 2 dimensions-the first dimension is the duration of the geyser eruptions, the second is the waiting time between eruptions. The scatter plot of the data is plotted, and it can be observed that there are two distinct clusters between the data points. The data can be successfully clustered using K-means algorithm wherein it is divided into 2 clusters. The scatter plot of the data with 2 clusters along with the cluster centroids is plotted in the figure displayed above.

Q 2)

Task: To generate random data with 2 subpopulation means and apply the Expectation-Maximization algorithm on them.

Code:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Mar 22 22:35:49 2018

@author: deepikakanade
"""

import numpy as np
import matplotlib.pyplot as plt
import math
from sklearn.datasets.samples_generator import make_blobs
from numpy.core.umath_tests import inner1d
from sklearn import preprocessing

class GMMClustering:

    def __init__(self, n_clusters, max_iterations, initializeMethod):
        self.name = "GMMClustering"
        self.n_clusters = n_clusters
```

```
        #self.n_features = n_features
        #self.n_data = n_data
        self.max_iterations = max_iterations
        self.initializeMethod = initializeMethod
        self.labels_ = []

    def fit(self, X):

        self.n_data, self.n_features  = X.shape

        if self.initializeMethod == "Random":
            self.mean_vectors = np.random.uniform(0,1, size=[self.n_clusters,self.n_features])
            self.covariance_matrices = np.random.uniform(0,1,
size=[self.n_features,self.n_features,self.n_clusters])
            self.alpha_vectors = np.array([0.5,0.5])

        self.log_likelihood_values = []
        for iteration in range(0,self.max_iterations):
            ### Expectation Step
            for i in range(0, self.n_clusters):
                temp_likelihood = inner1d(((X -
self.mean_vectors[i]).dot(np.linalg.inv(self.covariance_matrices[i]))), ((X -
self.mean_vectors[i])))
                likelihood = np.exp(-temp_likelihood/2.0)
                likelihood = likelihood / ((2* math.pi *
abs(np.linalg.det(self.covariance_matrices[i])))**0.5)
                e_step = self.alpha_vectors[i] * likelihood
                if i==0:
                    expectation = e_step
                else:
                    expectation = np.vstack([expectation,e_step])
            expectation = expectation /  np.sum(expectation, axis=0)

            ### Maximization step (Updation)
            for i in range(0, self.n_clusters):
                ### update mean vector
                mean_temp = np.multiply(X.T, expectation[i]).T
                self.mean_vectors[i,:] = np.sum(mean_temp, axis=0) / np.sum(expectation[i])

                covariance_temp1 = np.einsum('ij,kj->jik',(X - self.mean_vectors[i]).T,(X -
self.mean_vectors[i]).T)
                covariance_temp2 = np.multiply(covariance_temp1.T, expectation[i]).T
                self.covariance_matrices[i,:,:] = np.sum(covariance_temp2, axis=0) /
np.sum(expectation[i])
                ### update alpha vector
                self.alpha_vectors[i] = np.sum(expectation[i]) /  self.n_data
```

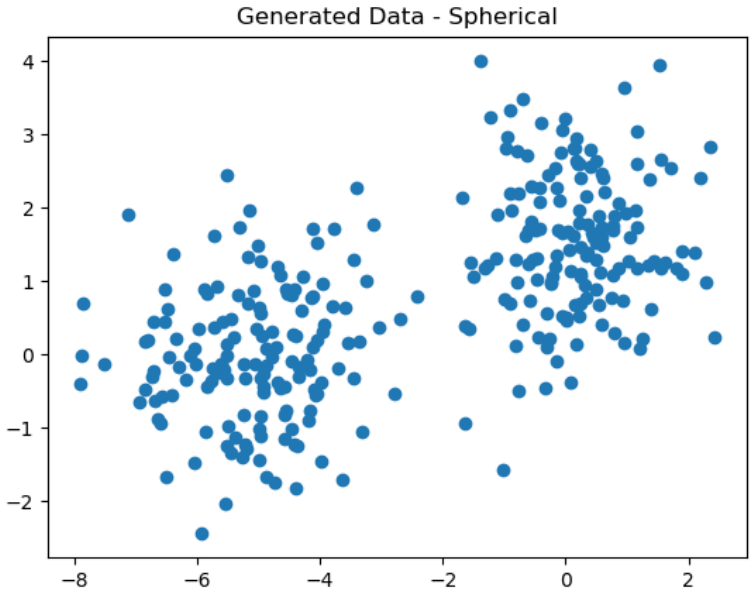
```
        ### evaluate log likelihood
        for i in range(0, self.n_clusters):
            temp_likelihood2 = inner1d(((X -
self.mean_vectors[i]).dot(np.linalg.inv(self.covariance_matrices[i]))), ((X -
self.mean_vectors[i])))
            likelihood2 = np.exp(-temp_likelihood2/2.0)
            likelihood2 = likelihood2 / ((2* math.pi *
abs(np.linalg.det(self.covariance_matrices[i])))**0.5)
            e_step2 = self.alpha_vectors[i] * likelihood2
            if i==0:
                expectation2 = e_step2
            else:
                expectation2 = np.vstack([expectation2,e_step2])
        log_likelihood = np.sum(expectation2, axis=0)
        log_likelihood = np.sum(np.log(log_likelihood))
        self.log_likelihood_values.append(log_likelihood)
        if iteration > 0:
            if self.log_likelihood_values[iteration] == self.log_likelihood_values[iteration-1]:
                print("Converged at iteration ", iteration)
                break
        ### assign cluster values
        self.labels_ = np.argmax(expectation2,axis=0)



n_features=2
n_clusters = 2
n_data = 300

### Generate data Spherical structure
centers = [[-5, 0], [0, 1.5]]
X1, y1 = make_blobs(n_samples=n_data, centers=centers, random_state=40)
X1_scaled = preprocessing.scale(X1)

plt.figure()
plt.scatter(X1[:,0],X1[:,1])
plt.title('Generated Data - Spherical')

gmm_model1 = GMMClustering(n_clusters, 100, "Random")
gmm_model1.fit(X1_scaled)
plt.figure()
plt.scatter(X1[:,0], X1[:,1], c=gmm_model1.labels_, cmap='jet')
plt.title('Generated Data - Spherical - GMM clustered')

### Generate data Ellipsoid structure
```

```
centers = [[-5, 0], [0, 1.5]]
X2, y2 = make_blobs(n_samples=n_data, centers=centers, random_state=40)
transformation = [[0.4, 0.2], [-0.4, 1.2]]
X2 = np.dot(X2, transformation)
X2_scaled = preprocessing.scale(X2)

plt.figure()
plt.scatter(X2[:,0],X2[:,1])
plt.title('Generated Data - Ellipsoid')

gmm_model2 = GMMClustering(n_clusters, 100, "Random")
gmm_model2.fit(X2_scaled)
plt.figure()
plt.scatter(X2[:,0], X2[:,1], c=gmm_model2.labels_, cmap='jet')
plt.title('Generated Data - Ellipsoid - GMM clustered')

### Generate poorly seperated subpopulations
centers = [[-5, 0], [-4, 0]]
X3, y3 = make_blobs(n_samples=n_data, centers=centers, random_state=40)
X3_scaled = preprocessing.scale(X3)

plt.figure()
plt.scatter(X3[:,0],X3[:,1])
plt.title('Generated Data - Poorly separated data')

gmm_model3 = GMMClustering(n_clusters, 1000, "Random")
gmm_model3.fit(X3_scaled)
plt.figure()
plt.scatter(X3[:,0], X3[:,1], c=gmm_model3.labels_, cmap='jet')
plt.title('Generated Data - Poorly separated data  - GMM clustered')

### old-faithful data
X4 = np.genfromtxt("/Users/deepikakanade/Desktop/oldfaithful.csv",delimiter = ',')
X4_scaled = preprocessing.scale(X4)

plt.figure()
plt.scatter(X4[:,1], X4[:,2])
plt.title("Scatter plot of old-faithful data")
plt.xlabel("old faithful geyser eruptions")
plt.ylabel("waiting time between eruptions")

gmm_model4 = GMMClustering(n_clusters, 50, "Random")
gmm_model4.fit(X4_scaled[:,1:])
plt.figure()
plt.scatter(X4[:,1], X4[:,2], c=gmm_model4.labels_, cmap='jet')
plt.title("Scatter plot of old-faithful data - GMM clustered")
```

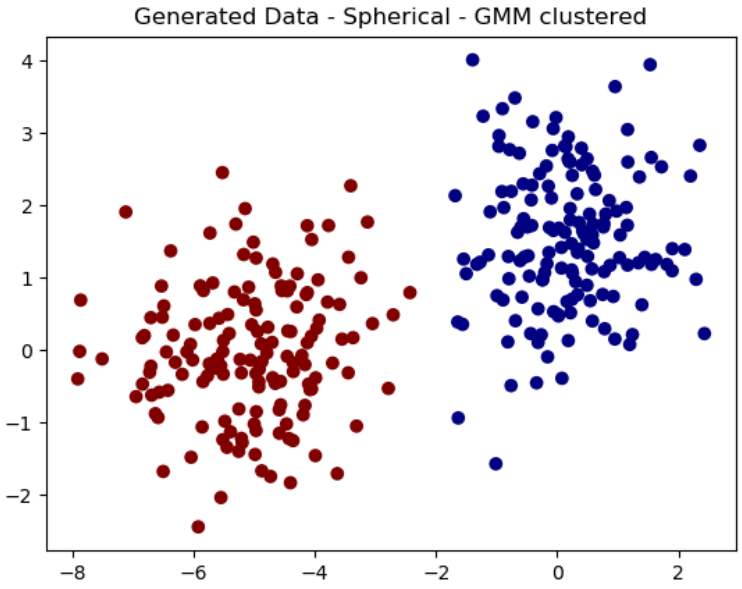
```
plt.xlabel("old faithful geyser eruptions")
plt.ylabel("waiting time between eruptions")
```

Output:

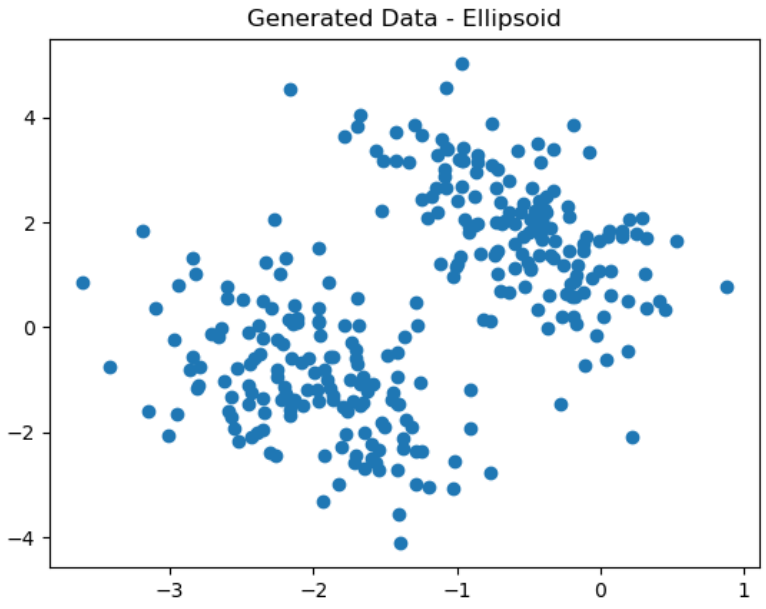
GMM distributions with Spherical data



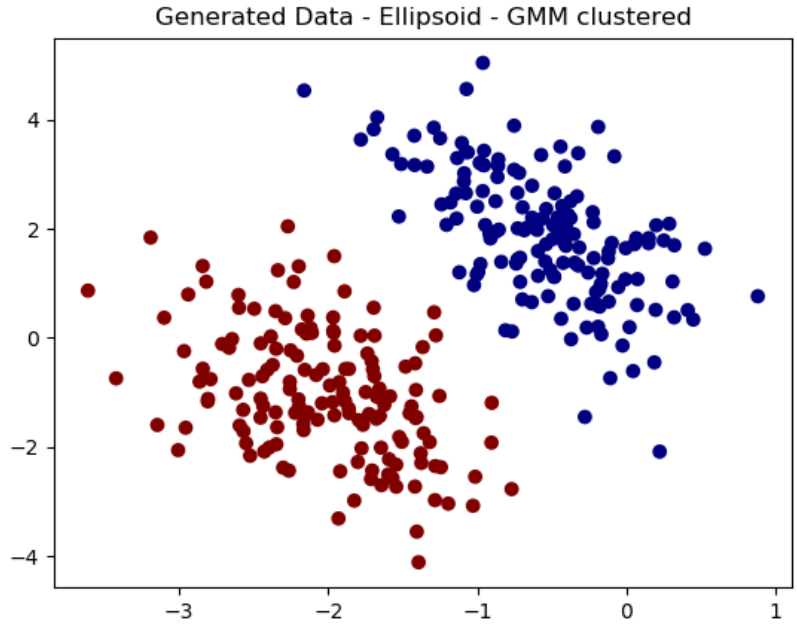
GMM distributions with clustered Spherical data



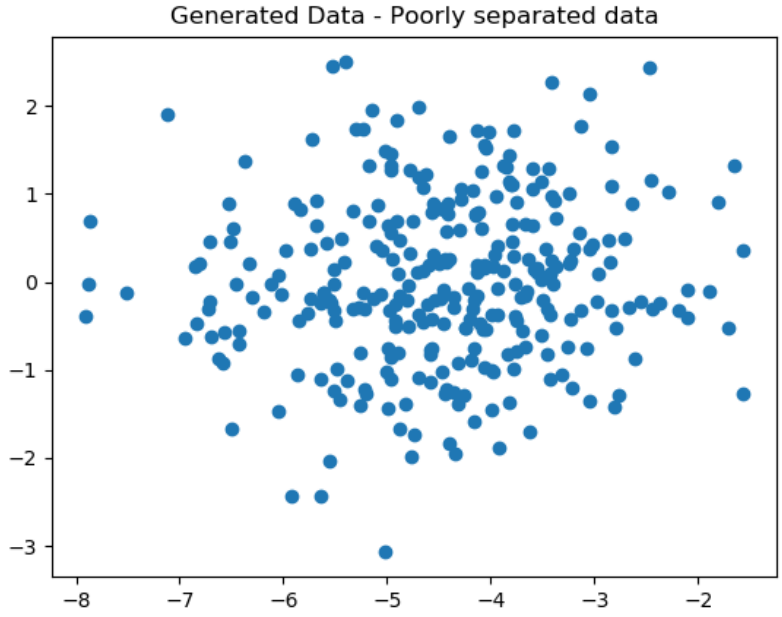
GMM distributions with Ellipsoidal data



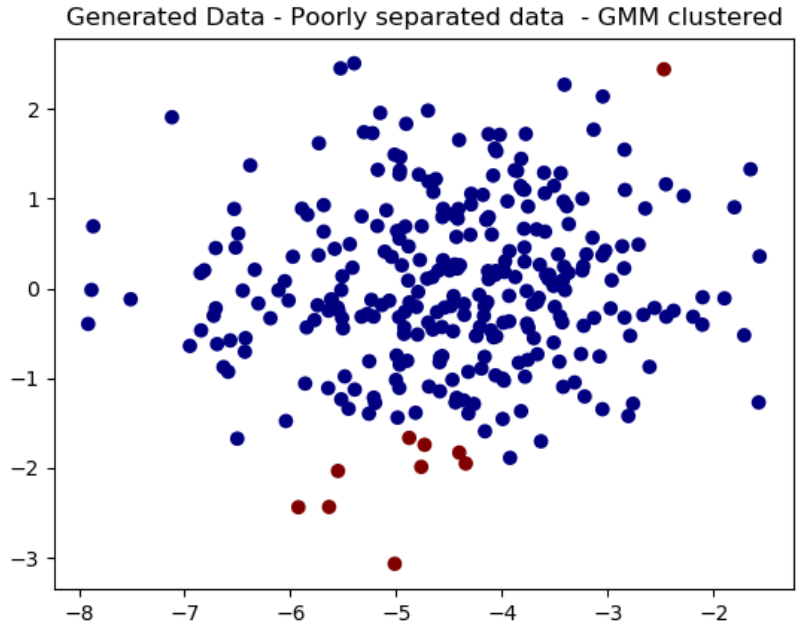
GMM distributions with clustered Ellipsoidal data



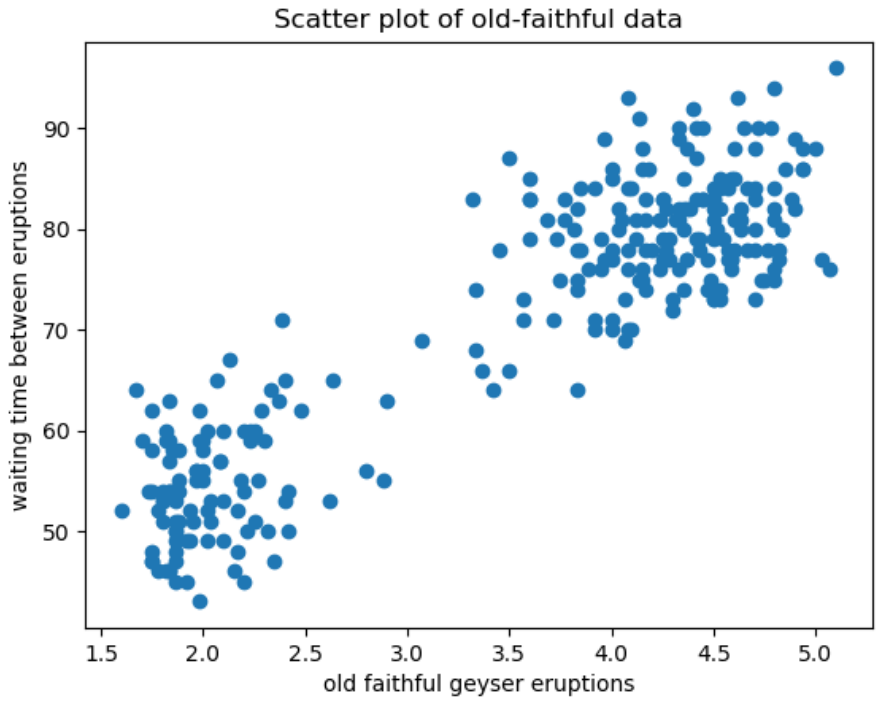
GMM distributions with poorly separated data



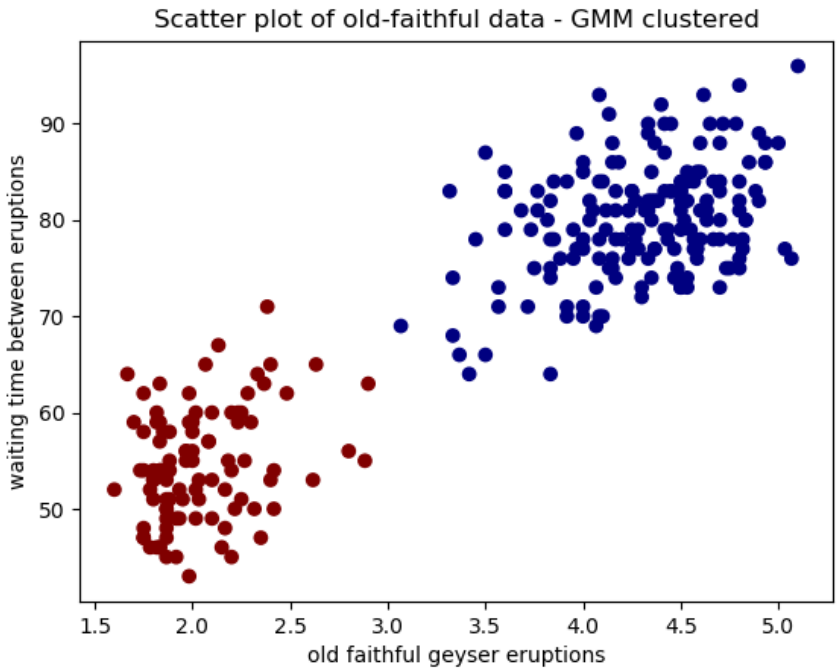
GMM distributions with clustered poorly separated data



GMM distributions with ‘old faithful’ dataset



GMM distributions with clustered ‘old faithful’ dataset



Output to indicate speed of GMM-EM with different type of distributions:

```
For spherical data:
Converged at iteration  67
For Ellipsoidal data:
Converged at iteration  43
For poorly seperated data:
Converged at iteration  340
For old faithful dataset:
Converged at iteration  27
```

Observations:

A Gaussian Mixture model was implemented using different types of data like spherical, ellipsoidal, poorly structured etc. The 2D data was generated using 2 subpopulation means.

After observing the figures closely, the quality of the clustering can be studied, as in the case of ellipsoidal or spherical type of data, the clustering is distinct, whereas in the poorly separated data, the clustering is not very visible. That is due to the fact that the data points are too close to each other and hence the clustering does not take place properly in case of poorly separated data.

The speed of the GMM-EM estimation can be studied by observing the time for convergence, which is very high in the poorly separated data compared to the spherical and ellipsoidal data. The data converged at an iteration of 340 for a poorly structured data whereas it converged at 67 and 43 respectively for the spherical and ellipsoidal data.

The GMM-EM algorithm was then run for the old faithful dataset and the obtained clustering was distinct and good.

Q. 3)

Task: To Run k-means clustering algorithm to cluster the row vectors of the data set and picking the best value of k.

Code:

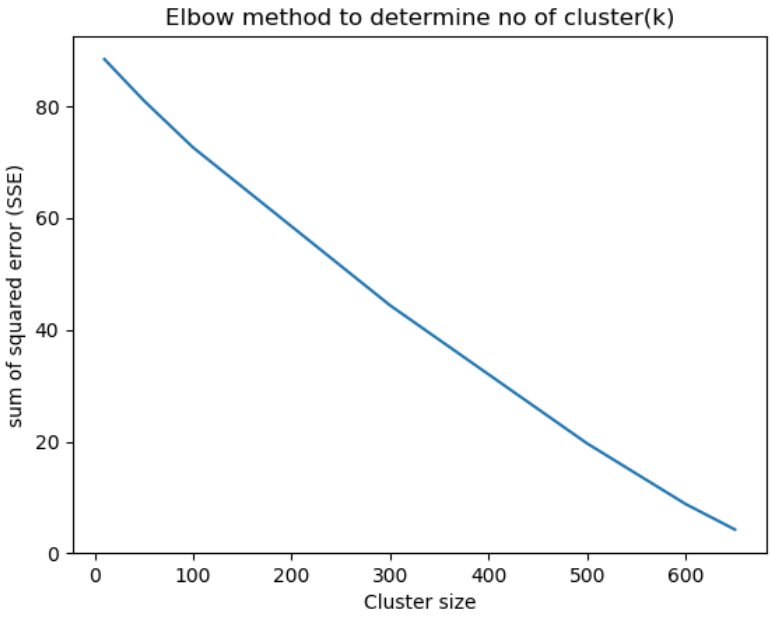
"""
Created on Tue Mar 13 23:59:08 2018

```
@author: deepikakanade
"""

import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
from sklearn.metrics import silhouette_score

trainFile = '/Users/deepikakanade/Desktop/nips-87-92.csv'
Data = np.genfromtxt(trainFile, delimiter = ',')

trial2 = Data[1:,2:np.shape(Data)[1]]

# k means determine k
distortions = []
index= []
#K = 15
K_dict={}
K_dict=[10,50,100,300,500,600,650]
silhoutte_score=[]
ad = {}
for k in sorted(K_dict):
    kmeanModel = KMeans(n_clusters=k)
    kmeanModel.fit(trial2)
    distortions.append(sum(np.min(cdist(trial2, kmeanModel.cluster_centers_, 'euclidean'),
axis=1)) / trial2.shape[0])
    ad[k] = kmeanModel
    silhoutte_score.append(silhouette_score(trial2, kmeanModel.labels_))


bestk = 100
best_model = ad[bestk]
for i in range(0,bestk):
    print('Cluster of: ',i)
    ad1 = Data[1:,:]
    print(ad1[best_model.labels_ == i][:,1])


# Plot the elbow
plt.figure()
plt.plot(sorted(K_dict), distortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Sum of squared distances')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```

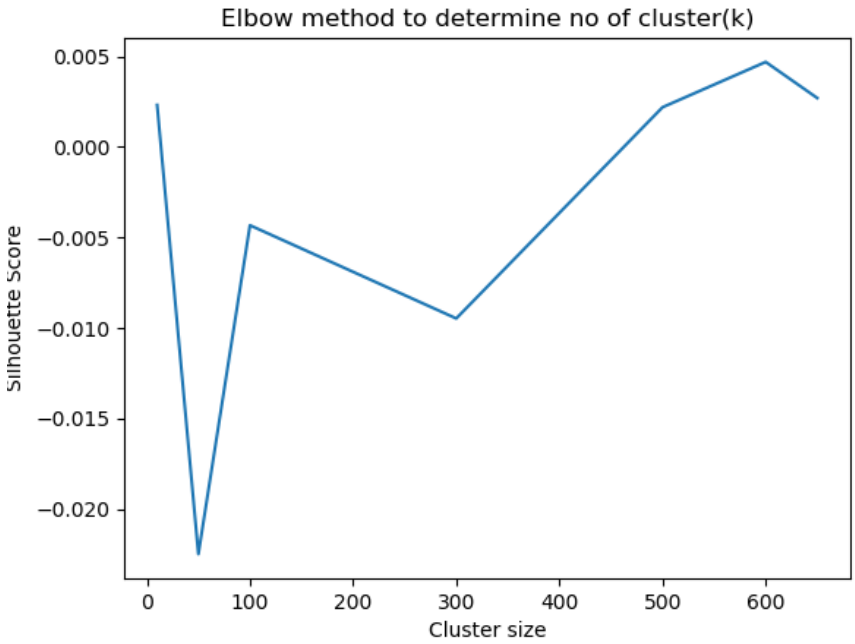
```
plt.figure()
plt.plot(sorted(K_dict), silhoutte_score, 'bx-')
plt.xlabel('k')
plt.ylabel('Silhoutte score')
plt.title('The Silhoutte showing the optimal k')
plt.show()
```

Output:

Elbow plot to obtain the optimum value of K



Silhouette score plot to obtain the optimum value of K



Clusters with document id's for the best value of K:

Cluster of: 0
[199118.]
Cluster of: 1
[199124. 199175. 199177. 199281.]
Cluster of: 2
[198810. 198960. 199094. 199095. 1990100. 1990102. 1990115.
199141. 1991102. 1991115. 1991128. 1991130. 199210. 199256.
199263.]
Cluster of: 3
[198813. 198964.]
Cluster of: 4
[19882.]
Cluster of: 5
[19879. 198775. 198843. 198845. 198865. 19892. 19893.
19894. 198910. 198914. 198918. 199012. 199038. 199052.
19916. 19917. 19919. 199115. 199166. 1992100. 1992113.
1992122. 1992126.]
Cluster of: 6
[19876. 198717. 198719. 198722. 198728. 198737. 198742.
198752. 198761. 198765. 198768. 198770. 198779. 198780.
198782. 198789. 19888. 198831. 198850. 198851. 198853.
198864. 198880. 198884. 198890. 198891. 198892. 198893.
198920. 198928. 198938. 198946. 198962. 198971. 198972.

198978. 198985. 198989. 198991. 198992. 198995. 198997.
198999. 19904. 19908. 199010. 199011. 199054. 199055.
199057. 199062. 199063. 199066. 199075. 199080. 199081.
199082. 199087. 199090. 1990106. 1990112. 1990113. 1990114.
1990118. 1990122. 1990125. 1990128. 1990133. 1990134. 1990140.
19913. 19918. 199128. 199133. 199164. 199165. 199172.
199173. 199176. 199178. 199183. 199185. 199197. 199198.
1991100. 1991106. 1991109. 1991123. 1991124. 1991126. 1991131.
1991135. 1991139. 1991142. 19924. 199213. 199228. 199233.
199257. 199258. 199262. 199266. 199277. 199292. 199297.
1992108. 1992110.]
Cluster of: 7
[198735.]
Cluster of: 8
[19887. 198966. 1992109.]
Cluster of: 9
[198724. 198763. 198818. 198860. 198870. 198873. 199015. 199222.]
Cluster of: 10
[198984. 199064. 199236. 199237.]
Cluster of: 11
[198814. 198970. 198979. 199097. 199217.]
Cluster of: 12
[199117. 199286. 199288.]
Cluster of: 13
[198769. 198773. 198854. 198936. 199043. 199046. 199047. 199147.
199151. 199188. 199193. 199246. 199252. 199296.]
Cluster of: 14
[198849.]
Cluster of: 15
[198712.]
Cluster of: 16
[19875. 1990130. 1991116. 19923. 19928. 199276.]
Cluster of: 17
[199026. 199142. 199171. 199215.]
Cluster of: 18
[198952.]
Cluster of: 19
[198921. 198924. 1990132.]
Cluster of: 20
[198993. 198994. 1990135.]
Cluster of: 21
[198750.]
Cluster of: 22
[1992114.]
Cluster of: 23
[1992118.]
Cluster of: 24
[198826. 198827. 198894. 198927. 198929. 199027. 199031. 199035.
199123. 199282.]
Cluster of: 25
[199059.]
Cluster of: 26
[198751.]
Cluster of: 27
[19877. 19902. 199111.]
Cluster of: 28

[198767.]
Cluster of: 29
[19872. 19885. 198887. 1990109. 1990138. 1991107. 1991140.
1992103. 1992119.]
Cluster of: 30
[19878. 198733. 198738. 198757. 19881. 19883. 198817.
198819. 198821. 198842. 198875. 198932. 198949. 198953.
198954. 198957. 198959. 198965. 198974. 198980. 198998.
19905. 199023. 199076. 199086. 199089. 199091. 199099.
1990107. 1990117. 1990123. 1990137. 199132. 199134. 199140.
199146. 199154. 199174. 1991117. 1991125. 1991138. 19925.
19929. 199211. 199212. 199220. 199221. 199245. 199265.
199278. 1992106.]
Cluster of: 31
[19884. 198816. 198868. 198968. 198975. 198976. 198977.
199073. 199096. 1990108. 1990111. 1990119. 1990121. 1990124.
1990126. 1990131. 199138. 199145. 199179. 199180. 199181.
199182. 199184. 199186. 199187. 1991103. 1991104. 1991110.
1991122. 1991129. 1991132. 1991133. 1991134. 1991136. 1991137.
1991143. 1991144. 199224. 199225. 199226. 199229. 199249.
199259. 199261. 199272. 199274. 199291.]
Cluster of: 32
[198784.]
Cluster of: 33
[198913.]
Cluster of: 34
[198788.]
Cluster of: 35
[199061. 199068. 1991112.]
Cluster of: 36
[198776.]
Cluster of: 37
[198783.]
Cluster of: 38
[199162. 199255.]
Cluster of: 39
[199244.]
Cluster of: 40
[198947. 199139.]
Cluster of: 41
[19926.]
Cluster of: 42
[198934. 199156.]
Cluster of: 43
[19871. 198710. 198725. 198730. 198731. 198734. 198781.
198820. 198859. 198879. 198982. 198988. 199033. 1990120.
1990127. 1990143. 199127. 1991105. 1991111. 199216. 199227.
199260. 199264. 199269. 199270. 199271. 199273.]
Cluster of: 44
[198741.]
Cluster of: 45
[198933. 199040.]
Cluster of: 46
[198956.]
Cluster of: 47
[198772. 198812. 198829. 198830. 198862. 199077. 1990104.

1990105. 199126. 199196. 1991120. 1991121. 199214. 1992107.]
Cluster of: 48
[198766.]
Cluster of: 49
[199280.]
Cluster of: 50
[198745. 198787. 198811. 198835. 198838. 198841. 198869.
198872. 198877. 198935. 198937. 198948. 198981. 199041.
199045. 199048. 199050. 199084. 199148. 199149. 199152.
199153. 199157. 199163. 199232. 199248. 199250. 199290.
1992102. 1992105. 1992111.]
Cluster of: 51
[198836. 199058. 199235.]
Cluster of: 52
[198986.]
Cluster of: 53
[19886.]
Cluster of: 54
[198746. 198889.]
Cluster of: 55
[198715. 198815. 198824. 198825. 198839. 198858. 198922. 198923.
198925. 198926. 198931. 199020. 199021. 199029. 199032. 199034.
199036. 199074. 199083. 199119. 199120. 199121. 199122. 199130.
199160. 199161. 199223. 199284. 199285. 199287. 199289.]
Cluster of: 56
[198983.]
Cluster of: 57
[198726. 198729. 198743. 198748. 198749. 198771. 198786.
198790. 198834. 198847. 198848. 198852. 198855. 198856.
198871. 198895. 19897. 19898. 19899. 198911. 198915.
198916. 198917. 19903. 19906. 199014. 199017. 199018.
199019. 199049. 199051. 199053. 199069. 19912. 199110.
199113. 199143. 199194. 199247. 199251. 199254. 199268.
1992112. 1992116. 1992120. 1992123. 1992124. 1992125. 1992127.]
Cluster of: 58
[198764.]
Cluster of: 59
[198828.]
Cluster of: 60
[198747.]
Cluster of: 61
[198755. 198866.]
Cluster of: 62
[19927.]
Cluster of: 63
[198753.]
Cluster of: 64
[198844. 1992121.]
Cluster of: 65
[19873. 198711. 198714. 198723. 198756. 198758. 198777.
198832. 198833. 198837. 198857. 198861. 198863. 198874.
198930. 198944. 198951. 198955. 198961. 198969. 198973.
1989100. 1989101. 19909. 199037. 199039. 199042. 199044.
199060. 199071. 199072. 199078. 199085. 199093. 199098.
1990101. 1990110. 1990129. 199137. 199168. 199189. 199190.
1991108. 1991114. 1991127. 19921. 199275. 199279. 199283.

199293. 199299.]
Cluster of: 66
[198718. 198727. 198739. 198740. 198785. 198846. 198885.
198939. 198941. 198942. 198958. 198967. 199056. 199067.
199092. 1990103. 1990139. 1991141. 199240. 199241. 199242.]
Cluster of: 67
[198760.]
Cluster of: 68
[198950.]
Cluster of: 69
[199218.]
Cluster of: 70
[198736.]
Cluster of: 71
[199030. 199125.]
Cluster of: 72
[1991101.]
Cluster of: 73
[198713.]
Cluster of: 74
[199065. 199131. 199234.]
Cluster of: 75
[1992115.]
Cluster of: 76
[19889. 198963. 198990.]
Cluster of: 77
[19891.]
Cluster of: 78
[198744.]
Cluster of: 79
[198940. 199169. 199170. 199238.]
Cluster of: 80
[198919.]
Cluster of: 81
[198945.]
Cluster of: 82
[199150.]
Cluster of: 83
[199231.]
Cluster of: 84
[199167.]
Cluster of: 85
[199079.]
Cluster of: 86
[199239.]
Cluster of: 87
[199158. 199219.]
Cluster of: 88
[198759. 1990141. 199159. 199195. 199298.]
Cluster of: 89
[199028.]
Cluster of: 90
[198721. 198754. 198762. 198840. 198878. 198881. 198883.
198886. 198888. 198943. 198996. 199013. 1990116. 1990136.
1990142. 199155. 199191. 199192. 199199. 199230. 199253.
199294. 199295. 1992101. 1992117.]

Cluster of: 91
[199129.]
Cluster of: 92
[19874. 198716. 198720. 198732. 198774. 198778. 198822. 198823.
198867. 198876. 19895. 19896. 198987. 19907. 199016. 199022.
199024. 199070. 199088. 19911. 19914. 19915. 199114. 199116.
199135. 199136. 199144. 199243.]
Cluster of: 93
[1992104.]
Cluster of: 94
[1991119.]
Cluster of: 95
[199025. 1991113.]
Cluster of: 96
[1991118. 19922.]
Cluster of: 97
[198912. 19901. 199112.]
Cluster of: 98
[199267.]
Cluster of: 99
[198882.]

Observations:

The k-means algorithm was run for the nips dataset whose columns represented the (scaled) number of times a specified word appears in the different documents. The first column specified a document id for each paper of the dataset.

The k-means algorithm was run for different values of k in the range of 10-700 and the corresponding elbow plot and silhouette score were computed. These plots helped in calculating the optimum value of the number of clusters. As seen from the silhouette score plot, there's a dip at the value of k=100. Hence, I chose 100 to be the optimum value of k. After finding the optimum value of k, the document id's for each cluster of the best value of k was obtained. The plots of silhouette score as well as elbow plot along with the document id's for the best k have been shown in the figures above.