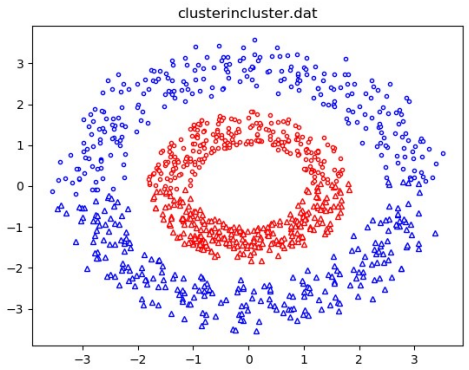
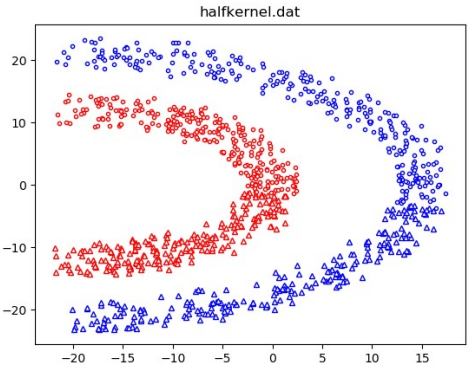
William Briguglio Nov. 16st 2018

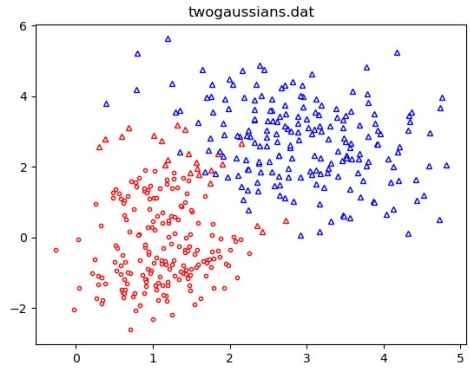
Assignment 3:

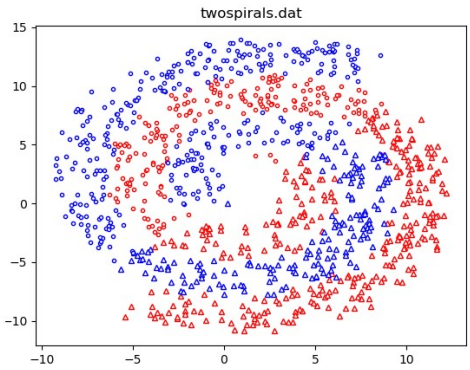
1) Following are the plots obtained from applying K-means and ExpectationMaximization to all four data sets with k=2:

Expectation-Maximization

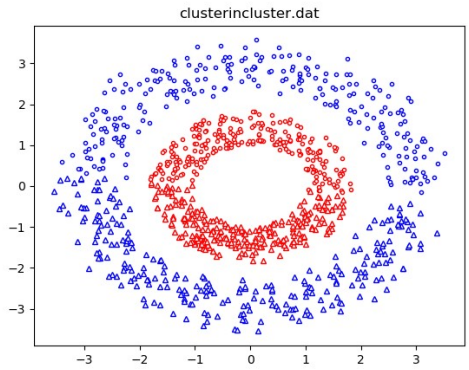


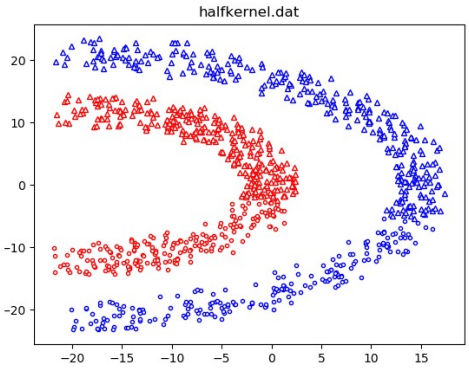


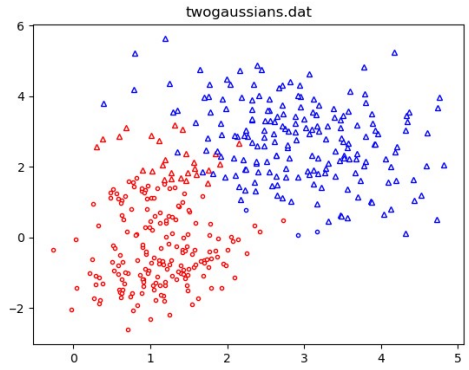


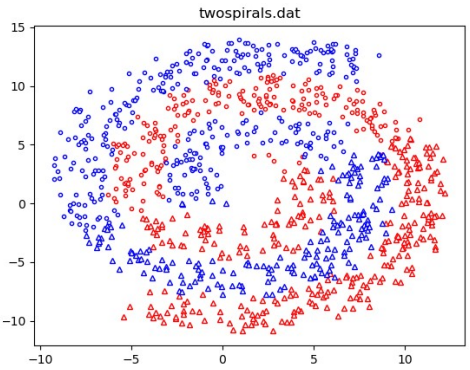


k-Means









2) Following are the best values for k according to the Davies-Bouldin index of validity:

	Expectation Maximization	k-Means
Half Kernel	K = 13 DBI = 0.603	K = 15 DBI = 0.586
Two Spirals	K = 3 DBI = 0.768	K = 17 DBI = 0.768
Two Gaussians	K = 16 DBI = 0.672	K = 17 DMI = 0.656
Cluster in Cluster	K = 2 DBI = 0.695	K = 2 DBI = 0.676

*DBI = Davies-Bouldin Index

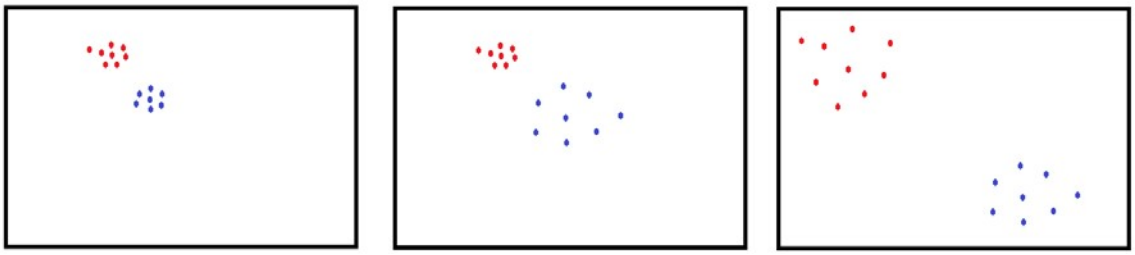
The Davies-Bouldin Index is the average RR value of the kk clusters where:

$$RR_{ii} = \max_{jj,jj \neq ii} \frac{SS_{ii} + SS_{jj}}{dd_{iijj}}$$

Where SS_{xx} is the "Scatter" of cluster xx, i.e. the average distance from the mean of cluster xx to an element of cluster xx. And dd_{xxyy} is the distance between the mean of cluster xx and the mean of cluster yy. Thus, minimizing DBI will minimize the average RR value which in turn will minimize the average max{(Scatter of ii + Scatter of yy)/distance between the means of xx and yy}. And, by minimizing $\frac{SS_{ii}+SS_{jj}}{dd_{iijj}}$, the distance between the means of clusters will be

maximized and the scatter of clusters will be minimized, since the larger the denominator or the smaller the numerator, the smaller the fraction. Thus Davies-Bouldin gives better scores to clustering that are tighter and further apart.

E.g. The following are clustering's that would receive lower(better) R scores.



3) The results obtained from the algorithm show that the EM and k-Means clustering algorithms preform poorly on datasets whose class regions are not convex polygons or ellipses. For this reason, both algorithms found completely inaccurate clusters for the two spirals, cluster in cluster, and half kernel data sets. However, both preformed reasonably well in the two gaussians data set.

This is because the two gaussian data set has class regions which are roughly shaped like ellipses and have a small overlapping area. The EM algorithm preformed better however because the EM algorithm represents clusters with gaussian distributions and finds the distributions that best fit the data, and the two gaussians have two gaussian distributions for class regions. Meanwhile, the k-means algorithm simply splits up the feature space into k-regions that are closest to k-means, like a Voronoi diagram, which can

only make convex polygons. This limitation in shapes for the clusters generated by k-means and EM is the reason why they preform poorly with data sets whose class regions do not approximate ellipses or convex polygons.

Implementation:

Clustering was implemented using scikit learn's `GuassianMixture()` and `KMeans()` functions. The Davies-Bouldin Index was found using scikit learn's `davies_bouldin_score()` function.

```
#_____________________MAIN_____________________________________________________

Data = ['halfkernel.dat',
        'twospirals.dat',
        'clusterincluster.dat',
        'twogaussians.dat'];

print("\n\n\n\tResults will be outputted to Results.txt...")
fp=open("Results.txt", "w")

KM = KMeans(n_clusters=2); #initialize k-Means object with k = 1
EM = GaussianMixture(n_components=2); #initialize Expectation maximization object with k = 1
k = 2;

for d in Data:
    fp.write("\nData Set:"+d+"\n");
    data = np.loadtxt("data/"+d);#Importing dataset
    X = data[:,1:3];#features
    Y = data[:,0];#labels

    for k in range(2,21):
        KM.n_clusters = k;
        EM.n_components = k;

        #plotData(X, Y, d);
        kmeans = KM.fit(X);#calcuate clusters
        KM_Y = kmeans.labels_;
        EMclusters = EM.fit(X);#get sample's clusters membership
        EM_Y = EM.fit_predict(X);#get sample's clusters membership

        fp.write("\tDavies-Bouldin Index(k = "+str(k)+"):\n\t\tk-Means:"+str(dbs(X, KM_Y))+"\tEM: "+str(dbs(X, EM_Y))+"\n");

        #plotData2(X, Y, KM_Y, d);
        #plotData2(X, Y, EM_Y, d);

fp.close()
```