Computer Science 145, Homework 4

Michael Wu
UID: 404751542

February 22nd, 2019

Problem 1

For purity and normalized mutual information, we can assign the cluster labels to a ground truth label according to the majority of the ground truth labels in the cluster. This would mean cluster 1 corresponds to ground truth label 2, cluster 2 corresponds to ground truth label 3, cluster 3 corresponds to ground truth label 1, and cluster 4 corresponds to ground truth label 4. We can construct the following table.

Ground \ Cluster	1	2	3	4	Total
2	5	0	0	0	5
3	0	5	0	0	5
1	0	1	4	0	5
4	0	0	1	4	5
Total	5	6	4	4	20

Overall 18 of the data points are matched, so our purity is 0.9. For normalized mutual information, we can use this table to obtain the following expression for information using the base 10 logarithm.

$$\begin{aligned} I(C,\Omega) &= \frac{5}{20}\log\left(\frac{20\times 5}{5\times 5}\right) + \frac{5}{20}\log\left(\frac{20\times 5}{6\times 5}\right) + \frac{1}{20}\log\left(\frac{20\times 1}{6\times 5}\right) \\ &+ \frac{4}{20}\log\left(\frac{20\times 4}{4\times 5}\right) + \frac{1}{20}\log\left(\frac{20\times 1}{4\times 5}\right) + \frac{4}{20}\log\left(\frac{20\times 4}{4\times 5}\right) \approx 0.513254 \end{aligned}$$

Because each ground truth label has 5 data points, the entropy of the ground truth labels is given by the following expression.

$$H(\Omega) = -\log\left(\frac{1}{4}\right) \approx 0.60206$$

The entropy of the clusters is given by the following expression.

$$H(C) = -\frac{1}{4}\log\left(\frac{1}{4}\right) - \frac{3}{10}\log\left(\frac{3}{10}\right) - \frac{2}{5}\log\left(\frac{1}{5}\right) \approx 0.586967$$

Then we have the normalized mean information shown below.

$$NMI(C,\Omega) = \frac{I(C,\Omega)}{\sqrt{H(C)H(\Omega)}} \approx 0.863387$$

In order to calculate the precision, recall, and F-score, we need to know the confusion matrix of this clustering. I obtained this by running the following code.

```
results = [(3,2), (3,2), (1,3), (1,3), (1,3),
  (4,4), (3,2), (3,2), (4,3), (2,1),
  (4,4), (2,1), (1,3), (2,1), (3,2),
  (2,1), (1,2), (2,1), (4,4), (4,4)]


TP=0
FN=0
FP=0
TN=0
for i in range(len(results)):
  for j in range(i+1, len(results)):
    a_ground, a_predicted = results[i];
    b_ground, b_predicted = results[j];
    if a_ground==b_ground:
      if a_predicted==b_predicted:
        TP+=1
      else:
        FN+=1
    elif a_predicted==b_predicted:
      FP+=1
  else:
   TN+=1

print(TP, " ", FN, " ", FP, " ", TN)
```

This told me that there were 32 true positives, 8 false negatives, 9 false positives, and 141 true negatives. The precision is $\frac{32}{32+9} \approx 0.780488$. The recall is $\frac{32}{32+8} = 0.8$. The F-score is then the following.

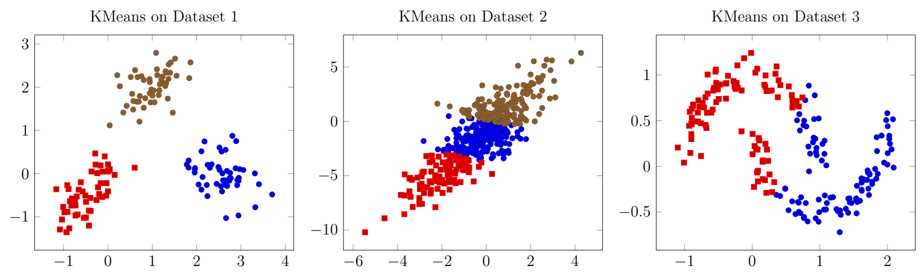
$$\frac{2 \times 0.780488 \times 0.8}{0.780488 + 0.8} \approx 0.790124$$

Overall this is a fairly good clustering since all our metrics have high scores.

Problem 2

a) In order to implement this, I simply assigned points to the closest cluster and calculated the new centroids by averaging the x and y values of the datapoints in the cluster.

b) The following graphs show the results of the clustering.



The following table summarizes the performance of this clustering using purity and normalized mean information.

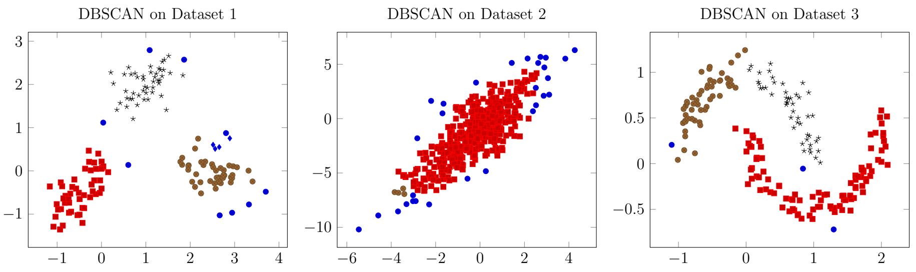
Dataset	Purity	NMI
1	1	1
2	0.764	0.046851365954979234
3	0.76	0.14502499937722388

c) K-Means works well because it is very fast and can handle datasets where clusters are clearly separated like dataset 1. Each iteration is linear with respect to the number of data points. However, it does not handle overlapping clusters with non-circular distributions like in dataset 2 well because it assigns data points to the nearest centroid. It also does not handle data which has irregular boundaries like dataset 3 well. Also, tuning the number of clusters could lead to different results. Luckily in this test we obtained the number of clusters from the number of labels in the training set.

Problem 3

a) I implemented this by assigning points to clusters as they are visited. Due to the way the existing code was structured, I also needed to ensure that reachable non-core points were not included in the noise set.

b) The following graphs show the results of the clustering.



The following table summarizes the performance of this clustering using purity and normalized mean information.

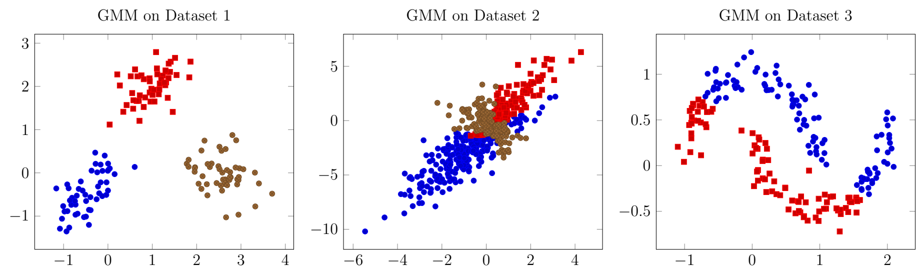
Dataset	Purity	NMI
1	0.94	0.9590647490609898
2	0.714	0.011352036737124684
3	0.985	0.8173489274692755

c) DBSCAN works well on irregular connected distributions such as dataset 3. It is also resistant to noise and is faster than GMM. A disadvantage of DBSCAN is that the correct search radius and the minimum number of points must be tuned correctly so that extra clusters are not created. Dataset 3 shows that an extra cluster is created because the algorithm could not make the jump between two halves of a distribution. There are also small extra clusters in datasets 1 and 2 with only four points. DBSCAN also does not handle overlapping distributions well and will gather nearby data points into one cluster.

Problem 4

a) In order to implement the expectation stage I had to divide the new weights w_{ij} by the sum of the weights for a given data point across all the clusters. In order to implement the maximization stage I set the weights for each cluster to the sum of the weights across all data points for that cluster. I also implemented the covariance formula and the initial means and covariance matrix in `DataPoints.py`.

b) The following graphs show the results of the clustering.



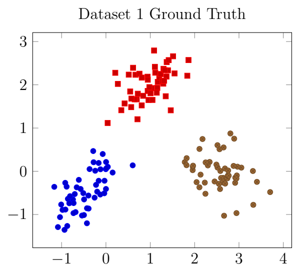
The following table summarizes the performance of this clustering using purity and normalized mean information.

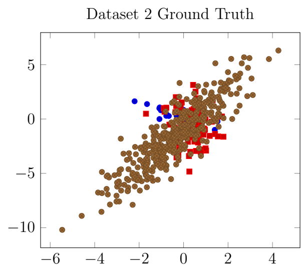
Dataset	Purity	NMI
1	1	1
2	0.764	0.07776497220479159
3	0.69	0.07594783950403936

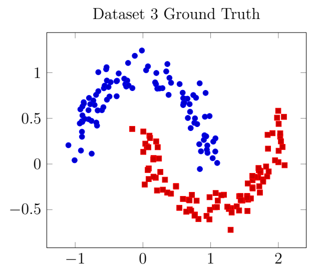
c) GMM works well because it can classify separated distributions like in dataset 1. It can handle varying shapes of distributions as long as they are somewhat elliptical, unlike K-Means which works best on distributions that are circular and have the same spread. Although GMM had some difficulty correctly clustering dataset 2, its clusters more closely resembles the ground truth labels than either K-Means or DBSCAN. Unfortunately GMM is very slow and does not handle irregular distributions such as dataset 3.

Ground Truth Labels

For reference, here is a visualization of the ground truth labels of our datasets.


Dataset 1 Ground Truth


Dataset 2 Ground Truth


Dataset 3 Ground Truth