Homework 4 Report

Rajiv Anisetti, UID: 904801422

November 18, 2018

1 Clustering Evaluation

1.1 Purity

After grouping by cluster, we can find the majority element of each cluster and add them up in order to find an average of purity. Using this technique, we get a purity of $\frac{5+5+4+4}{20} = \mathbf{0.9}$.

1.2 Precision

In order to calculate precision, we first need to calculate TP, FP, TN, and FN. First, we can calculate TP + FP as the number of distinct pairs *within our clusters*. This is equal to $C(5,2) + C(6,2) + C(5,2) + C(4,2) = 41$. Next, we can calculate TP as the number of pairs of same-class elements within the same cluster. This is simply $C(5,2) + C(5,2) + C(4,2) + C(4,2) = 32$. From here, $FP = TP + FP - TP = 41 - 32 = 9$.

We can count FN manually, and see that within clusters 2 and 3, there are false negatives, 8 to be exact. TN can be calculated in a simple counting manner, making sure not to double count pairs. There are 141 true negatives.

From here, our precision $= \frac{TP}{TP+FP} = \frac{32}{41} = \mathbf{0.78}$.

1.3 Recall

Using our previous calculations, the recall $= \frac{TP}{TP+FN} = \frac{32}{32+8} = \mathbf{0.8}$

1.4 F-measure

Our F-measure $= \frac{2*0.78*0.8}{0.78+0.8} = \mathbf{0.79}$.

1.5 Normalized Mutual Information

In order to calculate NMI, we first need to calculate the entropies of the clusters and classes followed by the mutual information of the clustering.

The entropy of the classes, $H(C)$ is simple, as we have an even distribution of classes. Because we have 5 instances of each class, $H(C) = 4 * (-0.25 * log(0.25)) =$ **2**.

The entropy of the clusterings, $H(\Omega)$, requires some more calculation. This is equivalent to $H(\Omega) = -0.25 * log(0.25) - 0.3 * log(0.3) - 0.25 * log(0.25) - 0.2 * log(0.2) =$ **1.985**.

Finally, mutual information, $I(C, \Omega)$, is calculated by looking at each cluster and the classes present within them, and using a log summation. The following equation represents the mutual information.

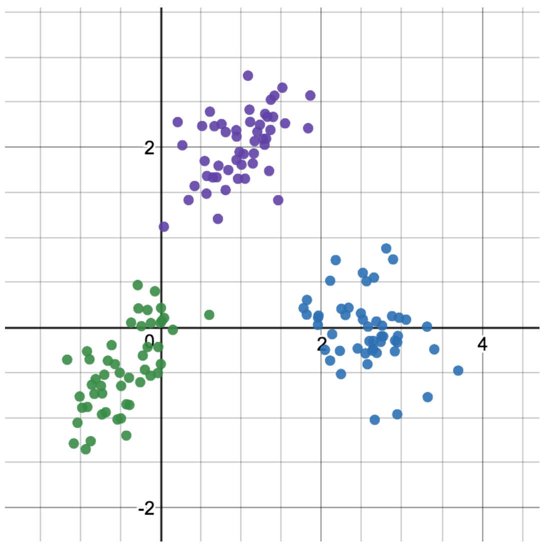
$\frac{1}{4} * log(\frac{100}{25}) + \frac{1}{20} * log(\frac{20}{30}) + \frac{1}{4} * log(\frac{100}{30}) + \frac{1}{5} * log(\frac{80}{25}) + \frac{1}{20} * log(\frac{20}{25}) + \frac{1}{5} * log(\frac{80}{20}) =$ **1.62**

Finally, our NMI is $\frac{I(C,\Omega)}{H(\Omega)H(C)^{\frac{1}{2}}} = \frac{1.62}{(2*1.985)^{\frac{1}{2}}} =$ **0.815**

2 KMeans

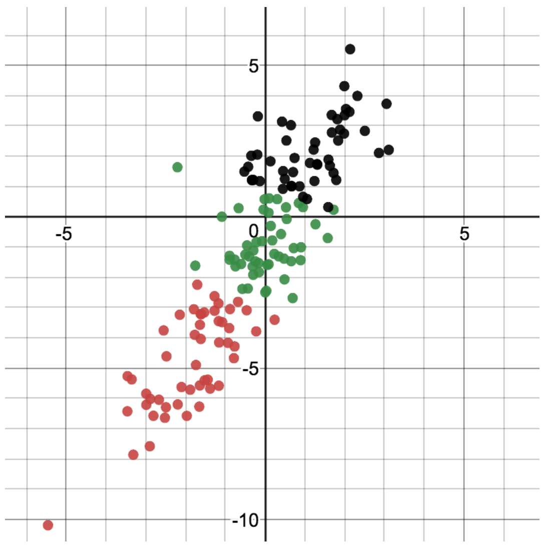
2.1 Plots

Dataset 1



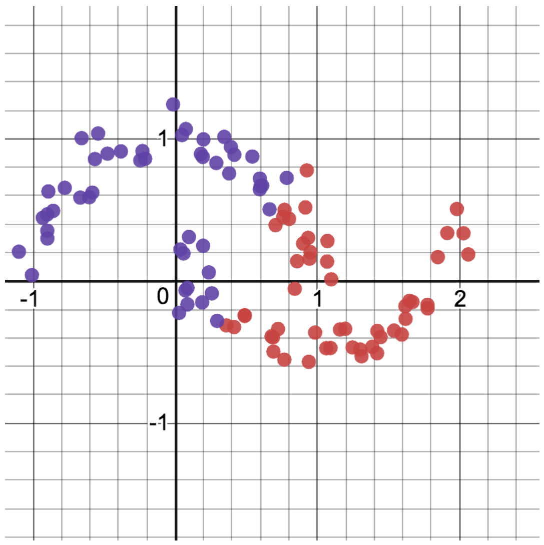
For this dataset, we get both a purity and NMI of 1.0, which is very good. This makes sense, as our three clusters are very separable by centroid, so the KMeans algorithm does a good job.

Dataset 2



In this dataset, we obtain a purity of 0.764 and an NMI of 0.0469. We see a much smaller margin between our clusters and they are also not as spherically-shaped. We know that KMeans tends to perform worse on non-spherical clusters, so this may be a reason for our decreased accuracy measures.

Dataset 3



For the last dataset, we get a purity of 0.76 and an NMI of 0.145. Looking at the plot, the clusters are much more intertwined than the previous datasets. However, KMeans was still able to distinguish the clusters better than the previous dataset. This may be due to the increased uniformity in terms of cluster density that we see in this final dataset. The previous dataset was not only non-spherical but variably dense for each cluster. This may be a reason for its worse relative performance to the final dataset, as we know that KMeans does not perform well on clusters of variable densities.

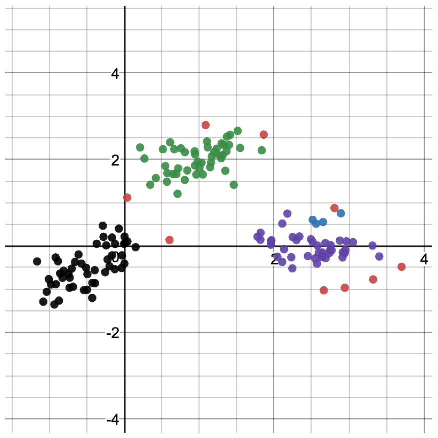
2.2 Strengths/Weaknesses of KMeans

- **Strengths:**
  - *Efficiency*: KMeans is relatively efficient, having a linear runtime relationship with respect to number of data objects
  - *Simplicity:* KMeans is a pretty simple algorithm to implement
  - *Interpretability:* KMeans clustering results are easy to interpret
- **Weaknesses:**
  - *Dimensionality:* KMeans does not work well when not dealing with hyper-spherical clusters
  - *Requires Prior Knowledge of K:* KMeans requires that the number of clusters, K, already be known before its execution
  - *Outlier Sensitivity:* KMeans is pretty sensitive to outliers
  - *Invariable Density:* KMeans does not work well on clusters of different density

3 DBSCAN

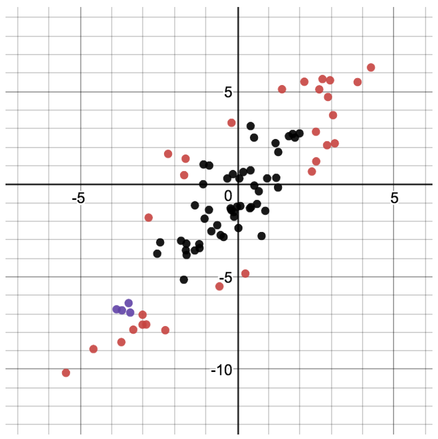
3.1 Plots

Dataset 1



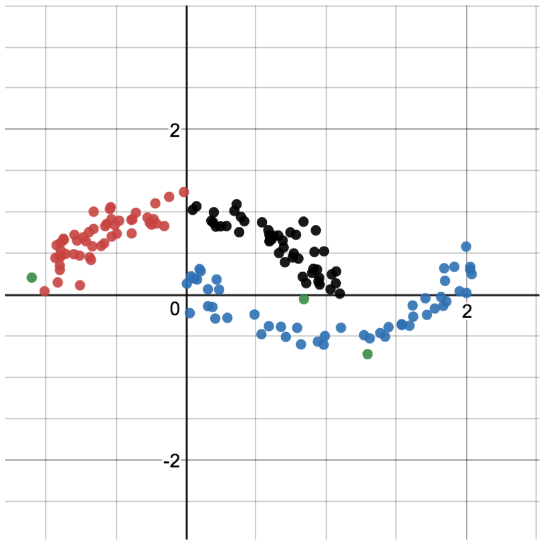
For this dataset, we get a purity of 0.94 and an NMI of 0.959. DBSCAN performs relatively well here, and it denotes the noise points in red. One thing to notice is the fourth output cluster on the right in blue. Perhaps with some tuning of parameters, we could eliminate this error and join in it with the purple cluster.

Dataset 2



For the second dataset, we get a purity of 0.714 and an NMI of 0.0114. As we can see, DBSCAN does not perform very well on this dataset, and there are many noise points as well. Once again, perhaps with some tuning of EPS and minpts, we can obtain a better clustering, but it is possible that DBSCAN is not a good clustering method for this dataset.

Dataset 3



For the last dataset, DBSCAN actually performs relatively well. With a purity of 0.985 and an NMI of 0.8173, DBSCAN actually performs better than KMeans and GMM. This may be due to the fact that the clusters are dense and relatively separated.

3.2 Strengths/Weaknesses of DBSCAN

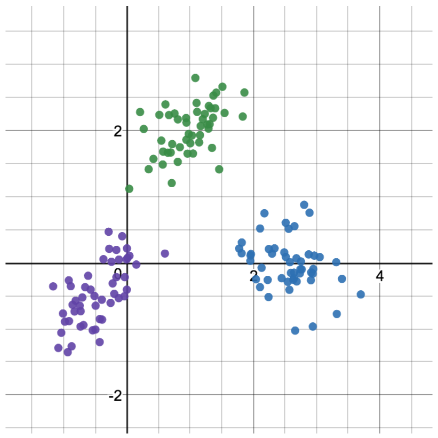
- **Strengths:**
  - *Noise/Outlier Sensitivity*: DBSCAN has a notion of noise points that other clustering algorithms do not take into account. Thus, it is robust to outliers in datasets.
  - *Cluster Specification:* DBSCAN doesn't require specification of the number of clusters, as KMeans does.
  - *Cluster Shape:* DBSCAN can find arbitrarily shaped clusters, while algorithms like KMeans are only good at detecting spherically shaped clusters
- **Weaknesses:**
  - *Parameter Sensitivity:* DBSCAN's output is heavily dependent on its parameters, EPS and minpts. It is difficult to find the correct tuning of these parameters.

- *Invariable Density:* DBSCAN does not work well on clusters of different density
- *Sensitivity to Dimensionality:* DBSCAN does not work well on datasets with high dimensionalities, due to its density-based nature

4 GMM

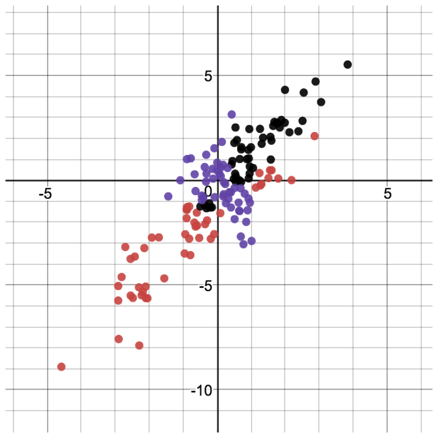
4.1 Plots

Dataset 1



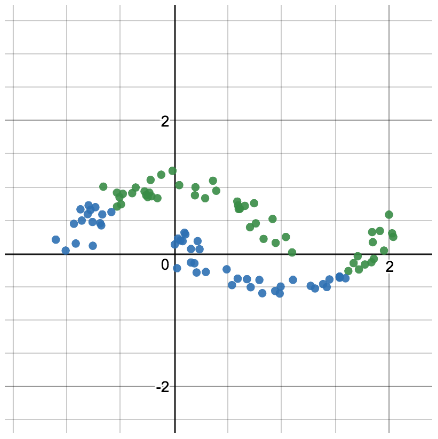
For the first dataset, GMM performs very robustly. We obtain both a purity and NMI of 1.0, showing that the clustering is very accurate. In this case, it is clear that the EM algorithm was able to find optimal gaussian distributions to cluster the dataset.

Dataset 2



For our second dataset, GMM performs more poorly. We obtain a purity of 0.764 and an NMI of 0.0777. The clustering becomes relatively vague near the center of the graph, which may be a result of our gaussian distribution optimization. However, this is more of an interspersed dataset than our other two, so a degree of error is expected.

Dataset 3



For our final dataset, we obtain a purity of 0.69 and an NMI of 0.0759. We get a pretty different clustering than the previous methods, and it almost looks as if the clusters are separated by a horizontal separator. It is important to note that GMM does not perform well on non-convex shapes, so this may be a reason for our reduced accuracy from KMeans.

4.2 Strengths/Weaknesses of GMM

- **Strengths:**
  - *Generality*: GMM can find more general clusters (with different densities and sizes), which general partitioning does not perform well on
  - *Parameter Characterization:* For GMM, clusters can be characterized by a small number of parameters
  - *Statistical Assumptions:* The clustering results of GMM may satisfy the statistical assumptions we make of the generative models
- **Weaknesses:**
  - *Local Convergence:* GMM has a tendency to converge to local optimals, but this can be combated by running the model multiple times with random initialization.

- *Runtime/Computation:* If the number of distributions is relatively large, GMM can be very computationally expensive
- *Cluster Estimation:* It is difficult the estimate the number of clusters for general mixture models
- *Invariable Shape:* GMM can only deal with spherical clusters