CS5011: Introduction to Machine Learning
Programming Assignment 3

Abhishek Narwekar
EE11B129

November 2015

1 Clustering

Question 1

The code to write an .arff file has two main components:

- The header
- The data

In the header, the first two attributes are numeric, while the third attribute is categorical. It was observed that the ranges of values in the first two attributes of the data are disparate, making clustering based on Euclidean distance difficult. This problem was overcome by **standardizing** the data before writing to the .arff file. This resulted in an improvement in performance.

On Cluster Purity

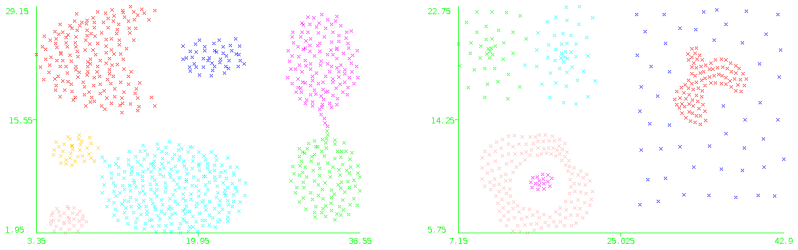
We define cluster purity, p, as:

$$p = \frac{1}{N} \sum_{c \in \mathbf{C}} n_c^{max} \tag{1}$$

where n_c^{max} is the number of points belonging to the majority in terms of class labels. A purity close to 1 indicates that all the clusters have a skewed class distribution. We note that this can happen if the number of clusters is very large (equal to the total data points, for instance). A good clustering algorithm achieves good purity for a small number of clusters. Ideally, when the number of clusters is equal to the number of classes in the data, the purity should be 1.

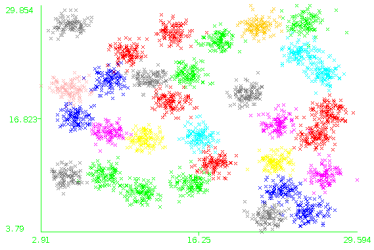
Question 2

The datasets have been visualized in figure 1.

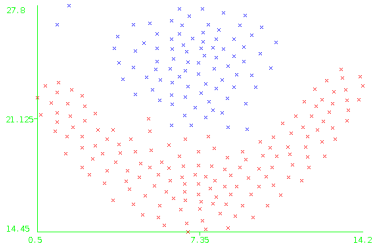


(a) Aggregation

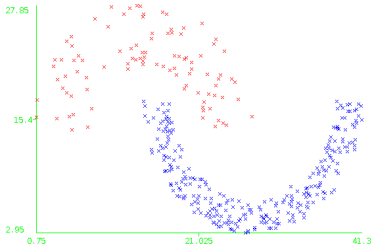
(b) Compound



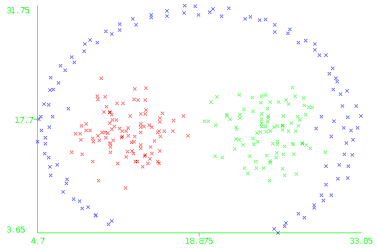
(c) D31



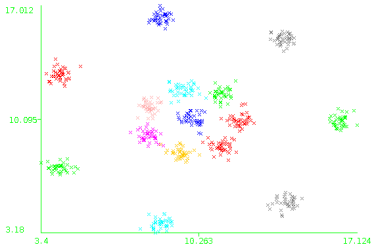
(d) Flames



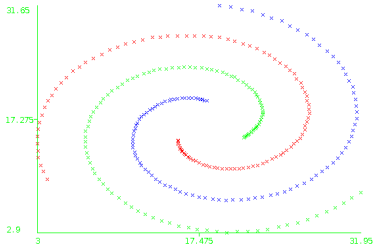
(e) Jain



(f) Path-based



(g) R15



(h) Spiral

Figure 1: A visualization of all the datasets

Which algorithm works well?

K-means clustering works well for datasets with convex clusters. Hierarchical clustering with compound link (hereby referred to as $HC_{complete}$ forces spherical clusters, and this works well for datasets with convex clusters. DBSCAN works

for non-convex clusters, but requires good separation between two clusters. Hierarchical clustering with single link (hereby referred to as HC_{single}) works also works well for non-convex clusters, with the added advantage over DBSCAN of allowing a user-defined number of clusters, and thus being more robust to noise points between clusters.

Aggregate: Since the compounds are clearly convex, k-means and $HC_{complete}$ will work well. HC_{single} will work well except for the region between the clusters, which may get misclassified. DBSCAN will work well if *minpoints* are set to be sufficiently high, so that the points on the boundary of two clusters will get classified as outliers.

Compound: K-means and $HC_{complete}$ will not work well due to the non-convex nature of some of the clusters. DBSCAN will work for particular values of *minpoints*, but won't classify the cluster in blue in the right of the dataset image. HC_{single} will also work well except for the blue cluster on the right half, which it may merge with the red cluster.

D31: The convex nature of the clusters implies that this dataset is ideally suited for k-means and $HC_{complete}$, $HC_{complete}$ possibly being more robust to noise points. DBSCAN will suffer from the noise points, and so will the HC_{single} algorithm. The latter may perform better since it shall output 32 clusters certainly, which may resemble the actual clusters.

Flames: This dataset is ideal for HC_{single} and DBSCAN with appropriate *minpoints*, given the considerable density of boundary points. However, k-means and $HC_{complete}$ will perform poorly

Jain: The two clusters are clearly separated here: even more than the flames dataset. Hence, just as before, HC_{single} and DBSCAN will work well. In fact, it will work even better than the Flames dataset. However, $HC_{complete}$ and K-means will not work well, as the clusters are non-convex.

Path-based: Owing to the non-convex nature of the clusters, $HC_{complete}$ and K-means won't work well. HC_{single} work better than DBSCAN, in which we shall need to be careful of the *minpoints*, an incorrect assignment of which may misclassify the points at the boundary.

R15: This is the only dataset with well separated convex clusters. Hence, all algorithms will work well on this dataset.

Spiral: This dataset has clearly separated non-convex clusters. Hence, HC_{single} and DBSCAN will work very well for this dataset. However, k-means and $HC_{complete}$ won't work well.

Question 3: The R15 Dataset

For $k = 8$, the cluster purity is 0.533. As the cluster size increases, the purity increases as well, as seen in figure 2. The increase in purity is justified, since an increase in the number of clusters results in a finer granularity, thereby resulting in a more skewed distribution in each cluster.

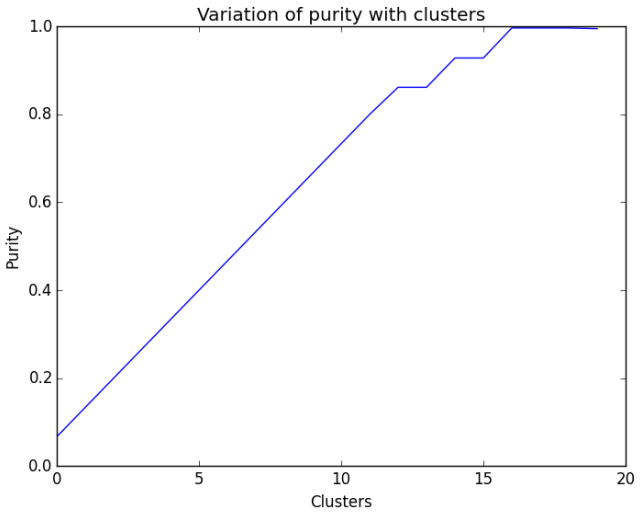


Figure 2: Variation of purity vs the number of clusters for the R15 dataset

Question 4 - The Jain Dataset

I wrote a java code to access the weka functions. This makes it easy to perform a grid search over the parameters. I varied *epsilon* from 0 to 1 in intervals of 0.05, and *minpts* from 1 to 20 in intervals of 1. I observed that for a large portion of the search space, DBSCAN resulted in a **single cluster**, for which the purity was around **0.74** - the fraction of points belonging to the dominant class. A reason for this observation could be that for **large values of** *epsilon*, the density based scanner identifies a relatively **large number of core points**, and classifies the entire data into a single cluster. On the other hand, **increasing the number of** *minpoints* identifies a relatively **fewer number of core points**. Thus, for large *minpoints*, more points are classified as **outliers**, thus getting the label "NOISE".
A right balance is obtained around $epsilon = 0.1$, *minpoints* taking values between 11 and 17. In terms of **purity alone**, the best performance was obtained when $epsilon = 0.05, minpoints = 2$, for which the purity was **0.973**. However, these parameters gave **12 clusters**. Among those values of parameters which gave **2 clusters**, the best purity, **0.858**, was obtained for

Performance of DBSCAN for the Path-based, Spiral and Flames datasets						
Dataset	*epsilon*	*minpoints*	*purity*	Clusters	Classes	Comments
Path-based	0.05	1	1	30	3	Too many clusters
Path-based	0.05	7	0.533	3	3	Best for 3 clusters
Flames	0.05	1	1	57	2	Too many clusters
Flames	0.1	9	0.975	2	2	Best for 2 clusters
Spiral	0.1	2	1	3	3	Best for 3 clusters

Table 1: Comparison of the performance of DBSCAN for 3 datasets

$minpoints = 17, epsilon = 0.1$.

Question 5: Path-based, Spiral and Flames Datasets

DBSCAN: The performance of DBSCAN for the Path-based, Spiral and Flames datasets is shown in table 1. For the Path-based and Flames datasets, a purity of 1 was obtained for some parameter combinations. However, the clusters identified were far larger than the number of classes in the datasets. For such cases, I have also reported the best purity for those parameters which resulted in clusters equal to the number of classes. In the Path-based dataset, the best purity for 2 clusters was 0.627, which isn't as good as for the Jain dataset. For the Flames dataset, DBSCAN performed better - the best purity was 0.975. For Spiral, however, a purity of 1 was obtained with exactly 3 clusters, indicating excellent performance.

Hierarchical: The best performance for various datasets has been summarized in table 2. For the spiral dataset, the **single link** clustering based on the linkage gave the best performance. The purity was **1**. For the flames dataset, clustering with **ward** linkage gave the best performance, again with a purity of **1**. For the path-based dataset, ward gave the best performance in terms of purity, which was **0.753** - worse than the other two datasets.

Dataset	**Clusters**	**Best Linkage**	**Purity**
Spiral	3	Ward	1
Flames	2	Ward	1
Path-based	3	Single	0.753

Table 2: Performance of Hierarchical Clustering for Path-based, Spiral and Flames Datasets

Question 6: The D31 Dataset

k	Purity
32	0.879
40	0.897
48	0.957
56	0.965
64	0.967
128	0.966

Table 3: Variation of performance of K-means for various values of k

K-means: The performance of K-means is shown in table 3. We observe that the k-means successfully identifies 32 clusters for $k = 32$ with a purity score of 0.879. The purity increases as we increase k. But it saturates for large values of k, which can be seen from the nearly constant value of purity at around 0.967 as k goes from 64 to 128.

Hierarchical Clustering: Using the Ward linkage with 32 clusters gives the best purity, 0.9632.

DBSCAN: Since the clusters are not well separated, **DBSCAN performs poorly**. I varied *epsilon* from 0 to 1 intervals of 0.05 and *minpoints* from 1 to 20 in intervals of 1 respectively. The number of clusters in the output **never exceeds 5**. The best purity was **0.1613**, obtained for *epsilon* = 0.05 and *minpoints* taking values 11 to 19 (all give the same purity).

2 Decision Trees

Question 1

The following arff files have been included: $agaricus - lepiota\_train.arff$ and $agaricus - lepiota\_test.arff$, which contain the train-test split of 7000 and 1124 points over the entire dataset.

Question 2

For $MinNumObj = 2$ (the default), the J48 classifier classifies **all points correctly**, implying that the classwise measures of precision, recall and F1-measure are **all 1**. Upon increasing the value of $MinNumObj$ by small amounts, the performance remains unchanged. But at around $MinNumObj = 100$, some points from class p get misclassified as class

	Class p	Class e
Precision	1	0.985
Recall	0.989	1
F1	0.995	0.993

Table 4: Performance of J48 for $MinNumObj = 100$

e, as seen in table 4. The results **remain unchanged** even after activating $reducedErrorPruning$

Question 3

Looking at the decision tree, we see that the splits occur at the following attributes:

1. Feature 5: odor
2. Feature 10: stalk-shape
3. Feature 20: spore-print-odor
4. Feature 8: gill size
5. Feature 7: gill spacing
6. Feature 21: population

Thus, these features are clearly important in classifying the edibility of the mushroom.

Question 4

The tree model has been submitted under the file name: $J48model.model$.