COMP40370 Data Mining

Practical 6 : Unsupervised learning - Clustering

IDE: PyCharm

A clustering model tries to group together similar training data points into "Classes". New data belonging to a class C would have similar characteristics like existing class member data points. Also, it shall be closer to those training data points in n dimensional feature space. This "Tendency of cooccurrence" is used in unsupervised classification method of clustering.
When a clustering model is trained on stored data, data distribution and grouping is unknown. Hence, most clustering algorithms start with random partitioning available data into k partitions/ choosing k data points as central tendency of cluster. Typical measure of similarity is "Distance" of data point with other data points (And | Or) with reference point like centroid of cluster.

Question 1: k-means clustering – 2 dimensional input

Question_1.csv has only 16 samples. Feature are "x", " y".; both with float data type. With only 16 data points, plot of data does not show any uniform clustering. 1 significant outlier is seen at (1.3, 22.8). In fact, a linear relation may exist between x and y and a straight line can be drawn between (2.4,6.9) and (19.7,18.5).
Yet, by visual inspection, we may roughly propose 3 clusters, as shown in figure. Also, empirical rule for deciding number of clusters "k" for "n" data samples is → k ≅ $\sqrt[2]{(n/2)}$. For 16 data points, (k ≅ 2.82) → 3

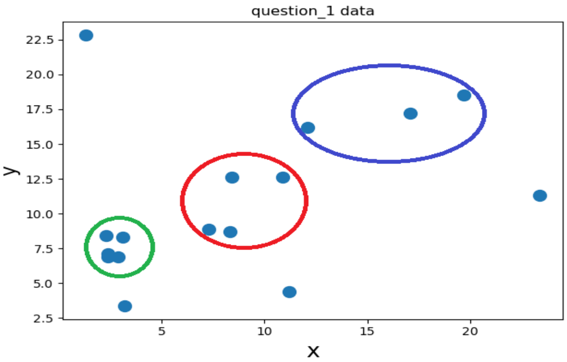

Figure 1 question_1.csv data - No definite visual clustering identified, 3 minor and 1 major outlier found

Q1.1 k-means model is defined with 3 clusters. Classically, when k-means model is fitted, it starts with 3 random centroids. By default, sklearn uses kmeans++ algorithm, which initializes a datapoint as first centroid using random_state. Remaining centroids are synthetically generated so that distance between all clusters is maximized; which shall yield better results [1].
If first centroid is seeded with new random_state integer each time, new set of centroids would be generated each time. So, comparative analysis of models working on same data would be difficult. So random_state is set to 0; which seeds the random generator with same integer "0" and produces same set of centroids during different runs of model repetitively.

Q1.2 Resultant data tuples with class-labels containing [<input : x, y>, <class-labels: cluster >] is saved in output/question_1.csv

Q1.3 Plot in figure 2 shows 3 clusters. Unlike predetermination in Figure 1; 2 major clusters are generated with centroid **1** [5.15, 7.56] and centroid **0** [16.64, 15.16]. A third cluster with centroid **2** [1.3 22.8] and it is formed around single outlier data point (1.3, 22.8).

Inference

- k-means operates on principal of minimizing distances within cluster
- k-means does not have definition for "Unclassified" data. This is observed when single outlier data point is assigned a separate class label **2**. Also, cluster 0 and 1 have some data points which are significantly distinct with respect to other data points.

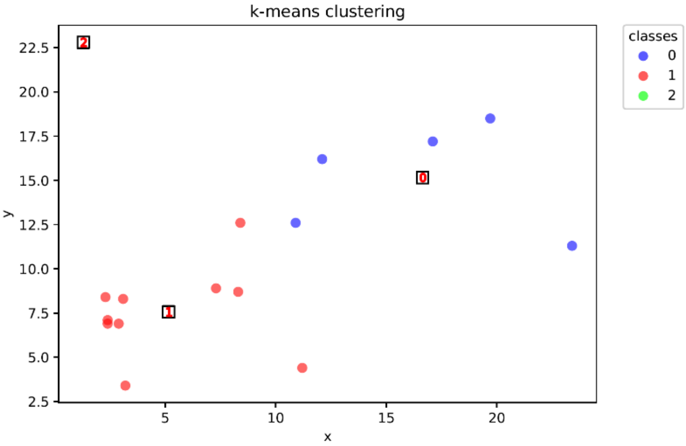

Figure 2 k-means clustering with k=3

COMP40370 Data Mining

Practical 6 : Unsupervised learning - Clustering

IDE: PyCharm

Question 2: k-means clustering – multidimensional input
Typical steps of Execution in k-means model training phase is given bellow. For example: with Number(clusters) [k] = 2, Number(runs) [n_init] = 3, iterations = 10, random_state = <any INT>

- random_state (RS) triggers PN sequence generator which create Starting centroids
- For a particular RS value, generated initial centroids are reapable
- So here, we get 3 sets of centroids for 3 n_init runs --- [(C_{11},C_{12}), (C_{21},C_{22}), (C_{31},C_{32})]
- For n in n_init<1,2,3>:
    - Choose set (C_{n1},C_{n2})
    - For i in iterations<1,2,3,…10>:
        - Classify all data points with respect to C_{n1},C_{n2} [associate with nearby centroid based on distance from centroid]
        - Recalibrate C_{n1},C_{n2} → $C_{n1}^{(i)}$,$C_{n2}^{(i)}$
        - fit again considering $C_{n1}^{(i)}$,$C_{n2}^{(i)}$ -→C_{n1},C_{n2}
    - Store results [min square distance, class-labels, centroids] for Run 'n'
- Choose results for a Run where root min square distance is minimum as model of choice

Q2.1 Data is read in data frame. To filter out NAME, MANUF, TYPE, and RATING attributes, drop command with inplace=True option is used.
Q2.2 k-means clustering is done with clusters=5, random state=0, Number(runs)=5, optimization steps=100. Results are recorded in data frame column 'config1'.
Q2.3 Again, similar model is created, but Number(runs) is now set to 100. Hence, model tries out and tunes 100 different initial sets of centroids before finding best cluster centroids. This is likely to improve accuracy of clustering. Results are recorded in data frame column 'config2'.
Q2.4 clusters labels recorded in Config1 and config2 are completely different. Only 1 sample retained its cluster. This is not desired and it is effect of distinct best fit centroids.
Q2.5 k-means clustering is done with clusters=3, random state=0, Number(runs)=100, optimization steps=100. Results are recorded in data frame column 'config3'.

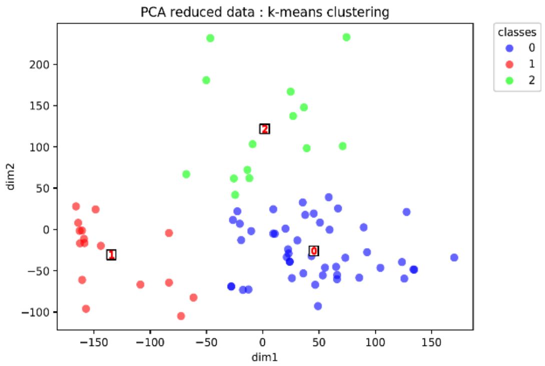

Figure 3 Clustering for PCA reduced data

Q2.6 To evaluate performance of clustering with different set of hyperparameters, 2 approaches are followed –
PCA dimensionality reduction : To visualize the clustering, I used PCA dimensionality reduction technique and reduced data to 2 dimensions; preserving more than 90% of data variance. When clustered with same hyperparameter setting in Q2.5, clusters in adjoined figure 3 were found. As most of data variance is preserved, and data sample size is considerably small, **cluster class-labels for PCA applied data and original data are same.** Refer output/question_2_PCA.csv for details.
Silhouette scores [2] for classification in Q2.2, Q2.3 and Q2.5 are calculated; which is calculated by averaging coefficients of each sample. The silhouette coefficient gives a measure of how close each point in one cluster is to points in the neighbouring clusters. Coefficient -1 indicates that sample lies in multiple clusters, 0 indicates samples on boundary of 2 clusters and +1 indicates best clustering. For a single data sample, silhouette coefficient is given by-

$$\boldsymbol{s = \frac{(b-a)}{max(b-a)}} \;\ldots\ldots\ldots\; \boldsymbol{a = \bar{d}(sample, points\ in\ own\ cluster), b = \bar{d}(sample, points\ in\ nearest\ cluster)}$$

Following results are obtained :

	Input_dim	n_clusters	random_state	n_init	max_iter	silhouette_score
Q2.2	12	5	0	5	100	0.3218
Q2.3	12	5	0	100	100	0.3601
Q2.5	12	3	0	100	100	0.4643
Q2.5PCA	2	3	0	100	100	0.5044

COMP40370 Data Mining

Practical 6 : Unsupervised learning - Clustering

IDE: PyCharm

Inference

- As noticed with class-labels generated in Q2.2 and Q2.3, **clustering with higher value of n_init gives better Silhouette scores**. This is not surprising as, model considers 20x more sets initial centroids before finalizing classes.
- Silhouette score improved with clusters generated in Q2.5 compared to Q2.3. This indicates that **actual clusters present in dataset are likely to be less than or equal to 3. Figure 3 supplements this conclusion as it displays 3 clusters with reduced dimensionality.**
- Silhouette score for PCA reduced clustering is moreover improved. So, **if contribution of individual feature in clustering is immaterial towards business decision, reducing dimensionality before clustering is a good option.**

Q2.7 Resultant data tuples with class-labels containing [<input : 12 features>, <class-labels: config1, config2, config3 >] are saved in output/question_2.csv

Question 3: k-means clustering Vs DBSCAN clustering– 2 dimensional input

k-means clustering is a partition based algorithm. It fits algorithm to find centroids which define an “area of coverage”. Any new data point lying in area for cluster C shall be nearest to centroid of cluster C, and would be part of C. Refer link for details.

DBSCAN clustering [3], in contrast does not define a central tendency which defines range of cluster. DBSCAN clusters areas of high density separated by areas of low density. It defines distance regulator **epsilon**, which is maximum distance between data samples to be considered as neighbours; hence part of cluster. Dense part of cluster is formed by core-samples; which have more than **min_samples** within distance less than epsilon. **DBSCAN allows to declare outliers. Any data point which are not within eps distance of a cluster data point is marked as outlier.** As, distance for all samples is checked pairwise with each other, **DBSCAN is memory expensive algorithm.**

From figure 4, We observe that 7 potential clusters of irregular shapes are present in data. Also, presence of outliers all over the feature space is challenging for algorithms.

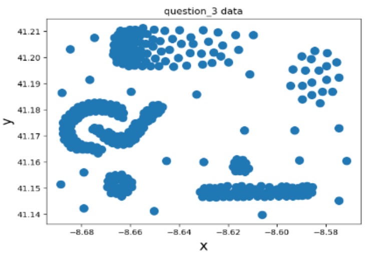

Figure 4 question_3.csv data - 7 potential clusters with multiple outliers identified with visual inspection

Q3.1, Q3.2 K-Means model with 7 clusters is defined and fitted. Resultant clusters are shown in Figure 5.

- Most of bundled data is correctly clustered. However, since circular boundaries formed due to centroid mechanism produce wrong clustering. In case of cluster **2** and **3**, we clearly see that original data has 2 distinct shapes resembling “C” and “✓”; and their edge points are misclustered.
- Also, all outliers are made a part of nearby clusters.

Resultant class-labels are store under feature “kmeans” in output/question_3.csv

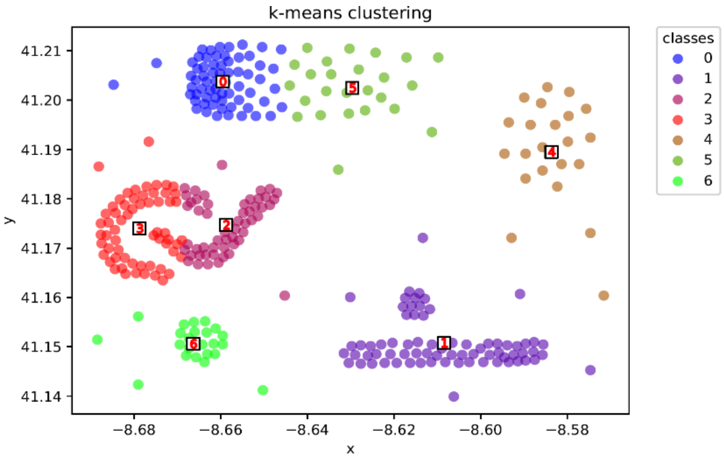

Figure 5 k-means clustering

COMP40370 Data Mining

Practical 6 : Unsupervised learning - Clustering

IDE: PyCharm

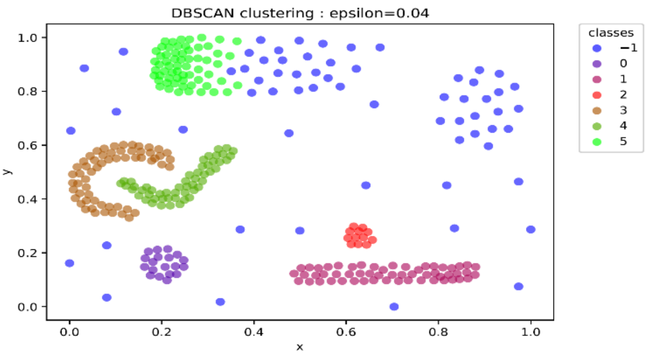

Figure 6 DBSCAN eps =0.04, min_samples = 4

Q3.3 Before applying DBSCAN, data is normalized using minmax scaling in range [0,1]. This is mandatory since DBSCAN operates on physical Euclidean distance between data points. When eps =0.04 is set as model parameter, with 4 min_samples, clustering as shown in Figure 6 is obtained. **Smaller eps dictate stringent clustering. Hence, highly dense data points are part of cluster. Whereas, less dense points which formed cluster 4 and 5 in case of kmeans are defined as outliers.** Resultant class-labels are store under feature “dbscan1” in output/question_3.csv

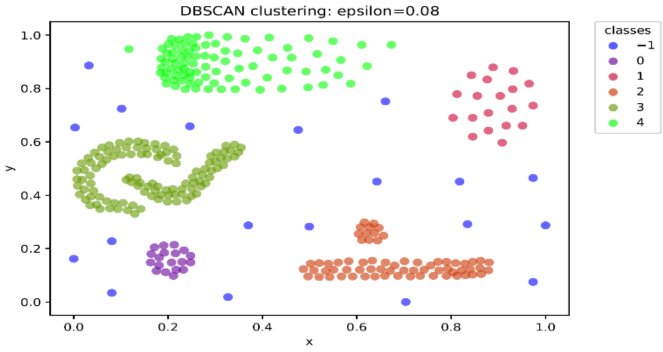

Figure 7 DBSCAN eps=0.08, min_samples = 4

Q3.4 Non-core samples are in proximity of a cluster, but do not satisfy min_samples criteria. When eps are made less stringent i.e. 0.08, more non-core samples are incorporated in cluster. Hence, data points classified as outliers in Q3.3 form a new cluster **1;** and **incorporated as part of cluster 4** as shown in Figure 7. Also, status of all outliers is maintained.
But choice of eps=0.8 has lost distinct cluster 3&4 obtained in Q3.3. Both are fused into cluster **3**, as shown in Figure 7. Same thing happens with cluster 1&2 in Q3.3, which fuse together into cluster **2**. Resultant class-labels are store under feature “dbscan2” in output/question_3.csv

Q3.5 Resultant data tuples with class-labels containing [<input : x, y>, <class-labels: kmeans., dbscan1, dbscan2>] are saved in output/question_2.csv

Q3.6 Performance of all algorithms is evaluated with silhouette_score. Kmeans and dbscan2 have good scores. But in case of dbscan1, silhouette score for DBSCAN is heavily penalized given presence of outliers, which essentially gave no cluster. So, it is not powerful metric to assess DBSCAN. Hence, **calinski_harabasz score** [4] is used, which is defined as ratio between the within-cluster dispersion and the between-cluster dispersion. Minimum score shows good clustering. As observed, dbscan1 gives best in class score; since it has maximum level of details [i.e. number of clusters]. Score of kmeans is very large, since intraclass variance is large due to inclusion of potential outliers.
HENCE, dbscan2 model with eps=0.8 and min_samples=4 is model of choice for given dataset. But, results can be finetuned by trying out values for eps in range 0.04<eps<0.08 to get optimal classification which leaves out outliers and disintegrate fused clusters discussed in Q3.4

Algorithm and hyperparameters						silhouette_score	calinski_harabasz_score
Q3.1	Kmeans	n_clusters=5	random_state=0	n_init=5	max_iter=100	0.5127	628.34
Q3.3	Dbscan1	Epsilon = 0.04	Min_neighbors = 4	-	-	0.3266	175.97
Q3.4	Dbscan2	Epsilon = 0.08	Min_neighbors = 4	-	-	0.5273	324.86

References

[1] k-means++ algorithm
[2] Clustering algorithm performance evaluations
[3] DBSCAN
[4] calinski_harabasz score