Concept of Clustering

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.

Question 1 Have a look at the simple k-means program you have been given. The program can work from a randomly generated set of points or a defined set. [Note, it can look for clusters of different k s; but for plotting purposes it fixed at k=3.] Use the init_board method to randomly generate 15 data-points; store this output and set the data variable to it.

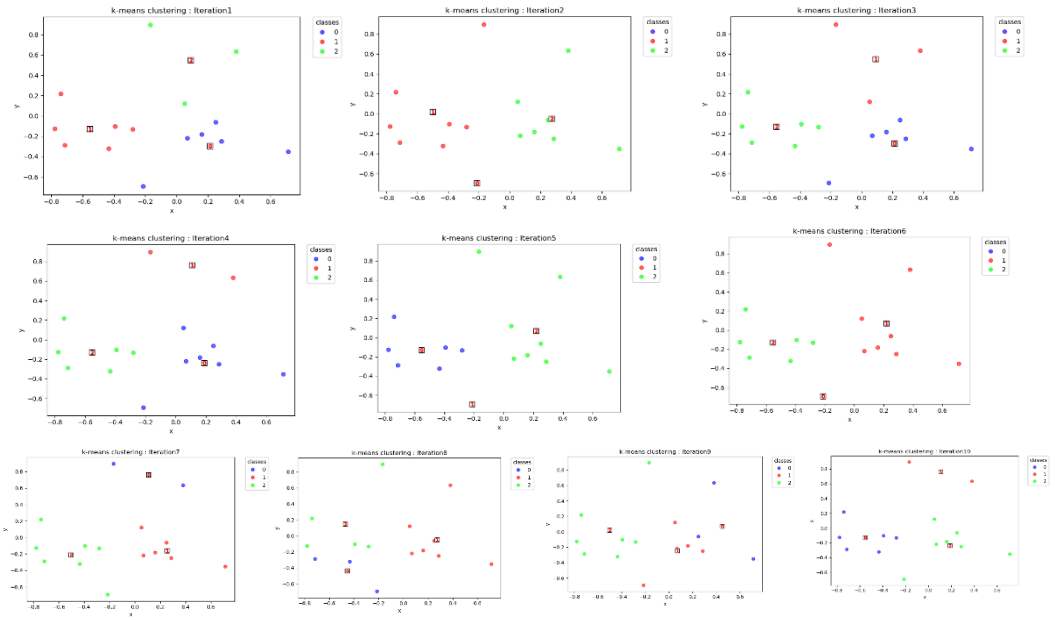
Provided init_board() function is used. Argument "15" is passed to generate 15 random datapoints using normal distribution. Resultant 15 data points are stored in variable "data".

1(a) Now run this dataset 10 times and note the clusters found by k-means. Carefully record how many of these clusters are the same/different (or roughly similar/different, if none are exactly identical).

```
Question 1
********************
Iteration 1
mean euclidean distance between initial centroids: 0.4558
silhouette_score Iteration 1 - kmeans with k=3 : 0.4022
calinski_harabasz_score Iteration 1 - kmeans with k=3 : 12.9802
saving plot of resultant cluster into ../output/question_1_iter_1.png
.
.
.
Iteration 9
mean euclidean distance between initial centroids: 0.2751
silhouette_score Iteration 9 - kmeans with k=3 : 0.1681
calinski_harabasz_score Iteration 9 - kmeans with k=3 : 5.7536
saving plot of resultant cluster into ../output/question_1_iter_9.png
Iteration 10
mean euclidean distance between initial centroids: 0.2442
silhouette_score Iteration 10 - kmeans with k=3 : 0.4495
calinski_harabasz_score Iteration 10 - kmeans with k=3 : 14.4086
saving plot of resultant cluster into ../output/question_1_iter_10.png
```

k-means clustering is partition type of clustering which operates on principal of minimizing mean distances between datapoints within a cluster. Initially, data distribution is unknown. So, when k-means model is fitted, it assigns k randomly choosen data points as centroids representing a cluster each.

For given case, k-means algorithm is ran 10 times with same data generated in previous step. As, k-means model is fit for 3 clusters. Hence, set of 3 new centroids is randomly choosen during each iteration. Following plots shows 3 clusters formed during each iteration. Data points within a cluster are shown with same colour and centroids are numbered. Labels[and hence colours] of cluster would not be consistent as it depends on randomly choosen initial centroid nearest to first datapoint.



Observations

- Across iterations, following clusters have roughly same training datapoints.
- Iteration 7 and 8 have no similarity with any other clustering iteration.

Iterations with similar data	Iterations with same data
(1,3,4,7,10)	(1,3), (4,10)
(2,5,6)	(5,6)

- **Visual inspection would observe that Iteration 4 & 10 produce best in class clustering.**
- **By visual inspection, we can als osay that datapoints (-0.2, -0.65), (-0.2, 0.9) and (0.75, -0.3) which are potential outliers. But, k-means does not have definition for "Unclassified" data and**

these data points are always accomodated in a cluster. It can reduce clustering accuracy as centroid would be biased towards this outlier and tend to shift towards outlier in feature space.

Corrolary can be given as such : If an NLP algorithm wants to cluster 3 sets of names from dataset containing Vegetables, Birds and Fruit names. BUT, accidently, dataset contains an outlier word ELEPHANT. If k-means algorithm is used, it forces one of the clusters to accept this outlier as part of cluster. Possibly, Elephant would be associated with Vegetables which has similar word "EGGPLANT". But, centroid of cluster shall be shifted towards this outlier word "ELEPHANT" and in future accommodate irrelvent words like "INFANT".

1(b) Discuss and comment on the variations you find across these runs.

Unsupervised clustering does not have predefined labels, hence evaluation is tricky. Often, these clustering methods use ground truths as reference points. In given case, No ground truth exists. Hence I used **Silhouette scores** for evaluating classification; which is calculated by averaging Silhouette coefficients of each sample; which is 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-

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

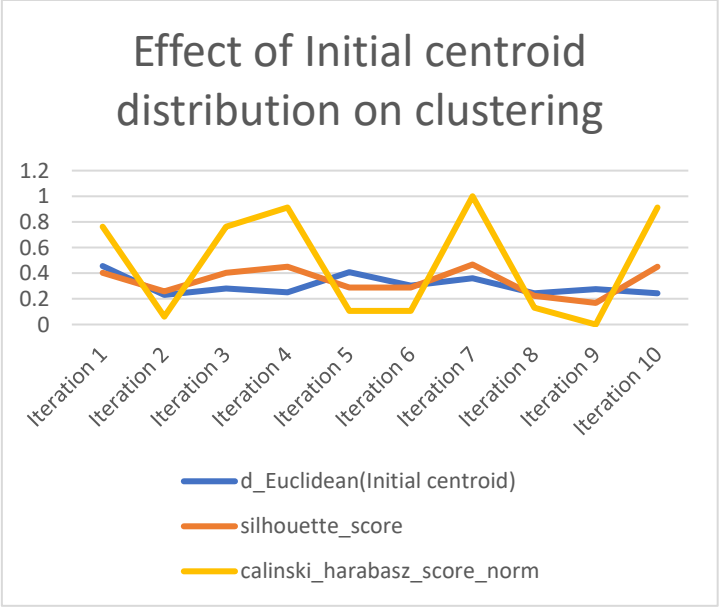
As seen in table below, scores for iterations 1,3,4,10 are better (~0.4) AND scores for 8,9 are poor(~0.2). This is consistent with visual inspection of clusters.

Additionally, **calinski_harabasz score** is used, which is defined as ratio between the within-cluster dispersion and the between-cluster dispersion. Minimum score shows good clustering. Score of kmeans is typically large, since intraclass variance is large due to inclusion of potential outliers. For plotting purpose, I have normalized calinski_harabasz score in range [0,1]

Iter	d_Euclidean (Initial centroid)	Silhouette score	calinski_harabasz score
1	0.4558	0.4022	12.9802
2	0.2306	0.2579	6.3346
3	0.2801	0.4022	12.9802
4	0.2497	0.4495	14.4086
5	0.4092	0.2889	6.7605
6	0.3031	0.2889	6.7605
7	0.3611	0.4677	15.2416
8	0.2439	0.2223	7.0055
9	0.2751	0.1681	5.7536
10	0.2442	0.4495	14.4086

Root cause analysis of variations in each iteration:

- As stated previously, k-means starts by assigning k random data points as initial centroid. Then, using distance of datapoint from centroid, clusters are formed and centroids are revaluated with respect to newly found cluster members. Algorithm exhaustively keeps updating centroid till set of datapoints in cluster do not change. So, **iterative process of fitting and updating centroid highly depends upon First set of randomly chosen centroids.**
- **As initial centroids change, initial clusters change.** When potential outliers alike in this case are present, final centroids and cluster members are not repetitive.



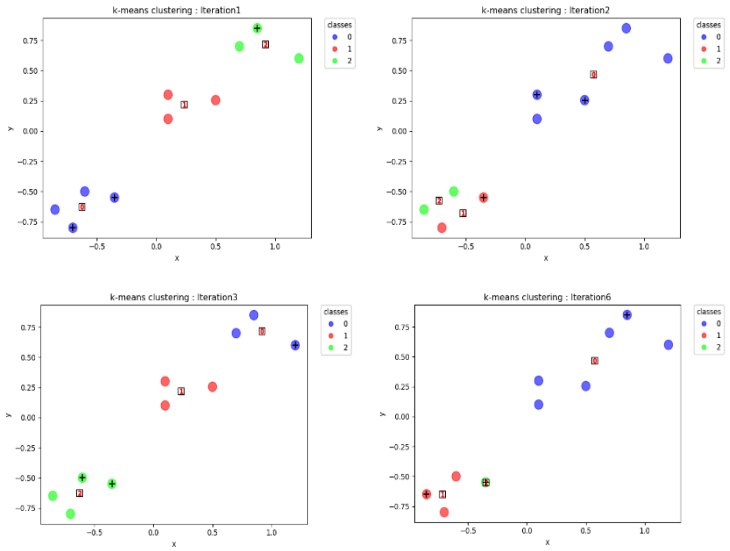
I calculated mean of Euclidean distances between initial centroids as an independent variable for assessing performance; as statistically, null hypothesis for data distribution is : normal shape. Expected result was direct proportion between mean distance between initial centroids and evaluation score. Whereas, I obtained roughly inverse relation between mean Euclidean distance in initial centroids and final silhouette score. **Hence, for this dataset : closer the initial centroids, better the final results.**

This effect seems dataset specific and against general notion, as initial centroids spread farther have better chance of approximating normal distributed data.

Question 2 Now, create your own dataset, by hand, with 10 data-points. You should construct this data-set to have three very clear clusters (a bit like the simple 6-point example shown in the program). Now run this set 10 times and note the clusters found by k-means. Report the results of these runs and the extent to which the same clusters are found. Modify some of the points in the dataset to ensure that most of the time the k-means finds the same 3 clusters. Report the changes you had to make to the dataset to achieve this consistent clustering.

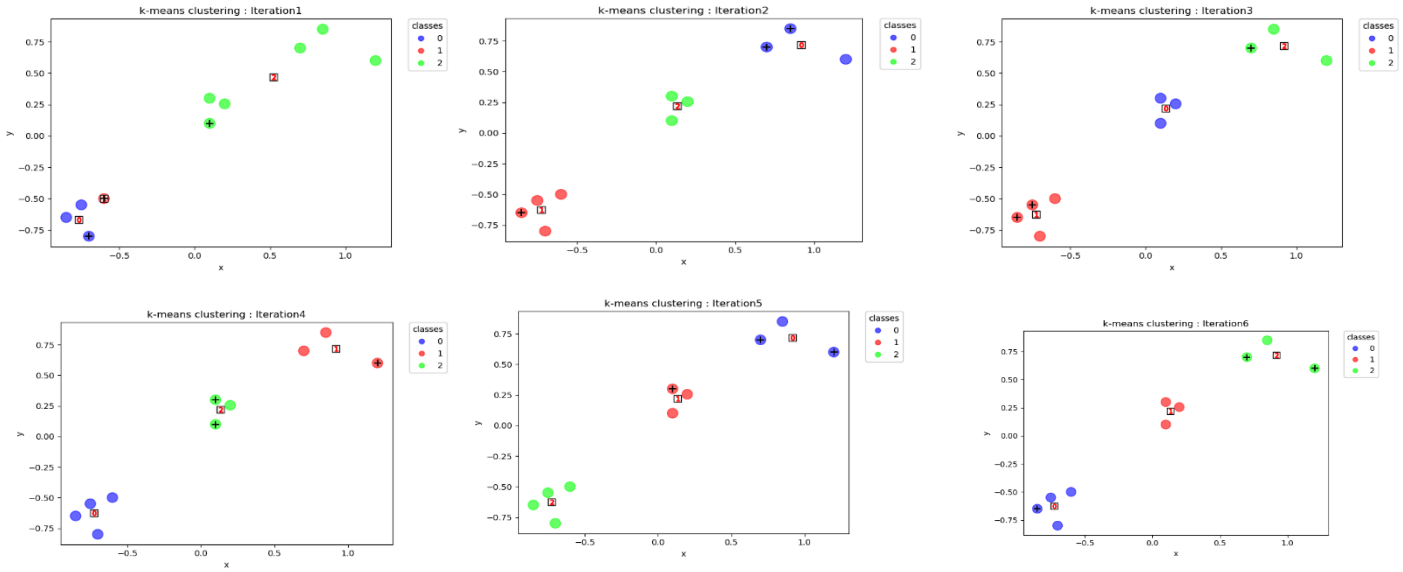
datapoint	1	2	3	4	5	6	7	8	9	10
x	0.7	-0.35	0.85	-0.6	1.2	-0.7	0.1	-0.85	0.5	0.1
y	0.7	-0.55	0.85	-0.5	0.6	-0.8	0.3	-0.65	0.255	0.1

With initial data distribution shown in above table, adjoined results were obtained. 4 representative iterations are shown. Iteration **1** is possible best fit, but 2, and 6 represent erroneous clustering. Initial centroid placement is also shown with a "+". For such small dataset, it is unreasonable to blame initial placement for wrong clustering.



By visual inspection, datapoint 2 and 9 are potential BOUNDRY points, which confuses the algorithm. When said datapoints are adjusted towards cluster centroids , better and consistent classification is obtained. Representative iterations display better clustering except iteration 1.

datapoint	1	2	3	4	5	6	7	8	9	10
x	0.7	-0.75	0.85	-0.6	1.2	-0.7	0.1	-0.85	0.2	0.1
y	0.7	-0.55	0.85	-0.5	0.6	-0.8	0.3	-0.65	0.255	0.1



Question 3 Do some web searches (in Google and Google Scholar) on methods that have been developed to improve the clusters found by k-means (especially, with respect to the issue of it finding different clusters on different runs). Describe one improvement with reference to the literature you read on the topic (provide full citations to the relevant papers found).

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. Typical steps of Execution in k-means model training phase is given bellow

For example: with Number(clusters) [k] = 2, iterations = 10

- So here, we get set of centroids for run--- [(C_1,C_1)]
  - For i in iterations<1,2,3,…10>:
    - Classify all data points with respect to C_1,C_2 [associate with nearby centroid based on distance from centroid]
    - Recalibrate C_1,$C_2 \rightarrow C_1^{(i)}$,$C_2^{(i)}$
    - fit again considering $C_1^{(i)}$,$C_2^{(i)}$ -→C_1,C_2
  - Store results [min square distance, class-labels, centroids]

Problems associated with k-means clustering

- Clustering is unstable and highly dependent on initial random choice of centroids. As highlighted in above example, 1st iteration with initial set of centroids (C1,C2) governs the result. When initial centroids change, final clustering is also changed.
- K-means clustering has high computational complexity; since distances between datapoints and cluster are iteratively calculated.

Approach1 Repeated k-means

This is nest in class approach, where we restart k-means several times from different initial solution to produce several candidate solutions, and then keeping the best result found as the final solution. Different sets of centroids are seeded with a PRN generator. Hence, above example becomes –

with Number(clusters) [k] = 2, Number(independent runs) [n_init] = 3, iterations = 10

- 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},$C_{n2} \rightarrow C_{n1}^{(i)}$,$C_{n2}^{(i)}$
    - fit again considering $C_{n1}^{(i)}$,$C_{n2}^{(i)}$ -→C_{n1},C_{n2}
  - Store results [min square distance, class-labels, centroids] for Run 'n'

Best results for a Run where root min square distance is minimum; is chosen as model of choice

This solution yields best clustering, but further increases computational complexity, and does not solve centroid problem.

Approach2 Better Initialization of Centroids : kmeans++

Kmeans++ proposes different Initialization of the centroids. Algorithm is as follows:
1. Choose first centre c1, chosen uniformly at random from dataset.
2a. Compute D(x) for each data point : it is distance between DP and nearest known centroid
2b.Choose one new data point at random as a new centre, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$
2c. Repeat 2b till k centres are found.
3. Proceed as with the standard k-means algorithm.
In the paper, Authors did show that Kmeans++ initializer outperforms random selection of centroids. This algorithm is widely accepted and incorporated in standard libraries and products like Scikit-learn, TensorFlow, MATLAB.

Reference: Arthur, D. and Vasilevskiy, S., 2007, January. k-means++: The advantages of careful seeding. 18th annual ACM-SIAM symposium on Discrete algorithms (pp. 1027-1035). Society for Industrial and Applied Mathematics. http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf