CS/ECE/ME532 Warm-up Activity for Additional Clustering Algorithms

*Estimated time: 20 min for P1, 10 min for P2, 15 min for P3*

Scripts are provided to help you visually investigate the k-means, mean-shift, and DBSCAN clustering algorithms using randomly generated two-dimensional data.

1. **K-means algorithm**: clusters are formed by partitioning the data points based upon their closeness to each cluster centroid.
2. *Section 1.1* of the associated script will create some random two-dimensional data points centered around a selectable number of centroids and a selectable amount of noise or variance in the data. Use the following parameters to get started:

|  |  |
| --- | --- |
| Number of points: | 100 |
| Number of centroids: | 3 |
| Noise: | 0.1 |

Using these parameters, select the “Generate points” button until the script produces a data set with three distinct clusters that can be separated, like in figure 1 below:

A screenshot of a cell phone

Description automatically generated

Figure 1: 100 random points in 3 distinct but loose clusters

1. *Section 1.2* of the script will cluster the data using the standard k-means algorithm, and it will show the clusters and overall coherence at each iteration step. For this activity, we will assume that the number of expected clusters is known, so leave the “Number of centroids” slider set to 3. Select the “Random centroids” button several times. The randomly selected centroids will be marked with a black “x”.
2. Are the randomly selected centroids necessarily evenly distributed among the clusters? How do you anticipate this will impact performance? (i.e number of iterations and final assignment of points to clusters)
3. Could input from the user improve this step? Would this be possible with higher-dimensional data that might be difficult to visualize?
4. Now select the “Find clusters” button several times and observe the clusters and centroids evolve as the algorithm iterates. Do the locations of the centroids always follow the same path? Are the final clusters always the same?
5. Now, select new random centroids and find the clusters. Do this several times. Are the final clusters always the same? Does the algorithm always “correctly” identify the clusters? What strategies could be implemented to deal with the problem of local minima?
6. **Mean-shift algorithm**: a centroid-based method that assigns data points to clusters by shifting them toward the local modes, defined as the maxima of the density function.
7. *Section 1.3* of the script will build a kernel density estimation (KDE) using a selectable bandwidth and a gaussian kernel defined as:

where is the bandwidth and is the squared Euclidian distance. The resulting contours will be displayed on the plot of the points. Set the “Bandwidth” slider to 0.5 and select the “Show KDE contours” button. Now try setting the bandwidth to 5.0 and showing the contours. How are they different? How do you anticipate this will impact the final number of clusters?

1. Try different values for the bandwidth, select the “Mean shift” button, and observe as the points climb to a local KDE peak. How important is selecting the appropriate bandwidth to finding the “correct” number of clusters? (i.e. what type of range produces the correct number of clusters?)
2. **Density-based spatial clustering of applications with noise (DBSCAN) algorithm**: a density-based model that attempts to assign data points to clusters based upon densities present in the space.
3. *Section 1.4* of the script will cluster the data using the DBSCAN algorithm. The slider labeled “epsilon” assigns the parameter , which defines the Euclidian distance that comprises a point’s “neighborhood”. The slider labeled “Minimum points” assigns the parameter , which defines the minimum number of neighborhood points (i.e. the density) necessary to start or extend a cluster. Use the following parameters to get started:

|  |  |
| --- | --- |
| Epsilon: | 2.0 |
| Minimum points: | 4 |

Select the “DBSCAN” button. The “Animate” checkbox can be enabled to observe point assignment as the algorithm iterates through all the points.

1. Does the algorithm “correctly” find the clusters?
2. Are there some points that aren’t assigned to a cluster? How are these points labeled?
3. Leaving the “Epsilon” slider at 2.0, try different values for the “Minimum points” slider.
4. What happens to the number of clusters and noise when ?
5. What happens to the number of clusters and noise when ?
6. Reset the “Minimum points” slider to 4 and try different values for the “Epsilon” slider.
7. What happens to the number of clusters and noise when ?
8. What happens to the number of clusters and noise when ?
9. Based upon your observations above, how sensitive is this algorithm to parameter selection?