K-means cluster

Algorithm description and implementation details:

First, we need to decide how many clusters we want to get (k) and initialize k initial cluster centers.

Then, for each data point, we calculate the distance from each points to the k cluster centers and find the which cluster center is the most close to the data point, then we can assign this data point to the nearest cluster center. After we finish assigning all the data points to a specific cluster, we can recalculate the cluster center by taking the average of the data points belonging to the same cluster.

We will carry out the process again and again until the centroid begin to converge.

In the implementation on the dataset cho.txt and iyer.txt, I randomly select the initial center

Pro:

It is easy to implement

For the round shape data point, the K-means can work well.

Con:

It is sensitive to the initial centroid setting

If the data point has different size、densities or shapes，the K-means can’t work well.

It is sensitive to the outliers.

DBSCAN cluster

Algorithm description and implementation details:

To implement this algorithm, first we need to determine two parameters in algorithm, which are  and MinPts. is the threshold to determine whether a border point belong to a specific cluster. And a point is a core point if it has more than MinPts within . Then, we have the border point, it is doesn’t have up to MinPts number of point within , but it has the distance less than from a core point. At last is the noise point, which is neither a core point nor a border point.

When I implement this algorithm, I first build a unvisited list which include all the data point and a null visited list. Then , I randomly choose the unvisited point P, get a list of its neighbors which has a distance less than from P. if the number of the list is less than the MinPts, I will mark the P as -1(NOISE).

Else, I will arrange a new cluster C for the point P. Then , we will traverse all the neighbor points (P’) from P(dis<). If P’ is not visited, we will mark P as visited and query the P’ neighbors. If P’’s the number of neighbors >= MinPts, we can append P’’s neighbors to the P’s neighbors. And if P’’s cluster is belonging to -1, we will add the the P’ to cluster C.

Pro:

1. DBSCAN can handle different shapes and sizes cluster.
2. DBSCAN can resistant to the dataset with noise

Con:

1. DBSCAN can’t handle data with varying densities.
2. It’s hard to determine the right parameters for DBSCAN

DBSCAN