Andrew Elenbogen and Quang Tran

CS 324: Data Mining

Final Project Write-up

For our final project, we implemented the DBScan algorithm, a density base clustering approach. This algorithm involves the user specifying two values: radius and MinPoints. The algorithm finds all points that have at least MinPoints data points within radius of them. It designates these points Core Points. Then, all points which did not have sufficient surrounding points but are within radius of a Core Point are designated Border points. Finally, the remaining points are assumed to be noise and are removed from the dataset completely. To actually generate the clusters, the algorithm iterates through all Core Points in the data set and places any point that is radius or less from another Core Point into the same cluster as that point. To achieve this simply and quickly, we drew an edge from each Core Point to each other sufficiently close core point, then employed the Floyd–Warshall algorithm on the resulting adjacency matrix.

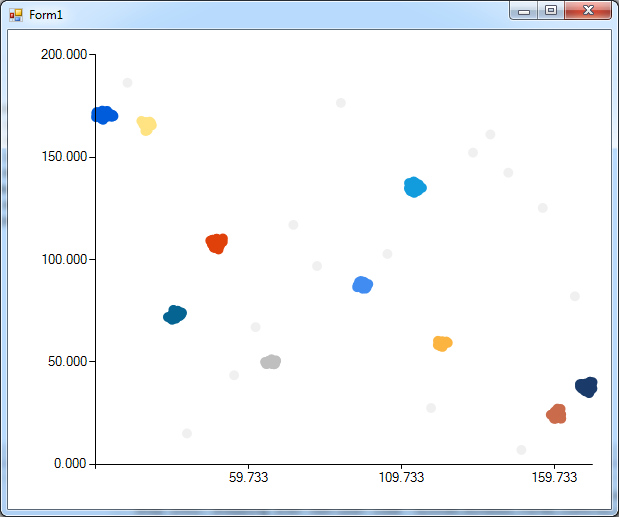
The final step of the algorithm is to assign Border Points to clusters. We do this by calculating the distance from each Border Point to the closest point in each existing cluster. The border point is then placed in the cluster with the lowest such distance.

We employed Euclidian distance as our data metric, as our sources stated that this general practice. We did some testing with Euclidian squared but it lead to a drop in performance in virtually all cases.

We began by testing our algorithm on the writing portfolio data set. On this dataset. as per the linked text book, we graphed the distance from each point to it’s kth nearest neighbor, for a reasonable k, sorting those distances in increasing order before graphing. We then picked the value located at the heel of that graph for the radius. In our case, that meant radius = 3.68 for k=9.

Some experimentation showed that min points should be around 10-20 to avoid either classifying all of the data as noise or making many 1-2 item clusters. Doing this results in clusters of the following Sizes:

Cluster 1: 56, Cluster 2: 17, Cluster 3: 6, Cluster 4: 10, Cluster 5: 9, Cluster 6: 11, Cluster 7: 6, Cluster 8: 12, Cluster 9: 5  
This seemed to us to be superior to agglomerative clustering, as that method throws almost all the points into a single giant cluster. However, to verify our algorithm was working, we then tried it on a dataset which made the clusters obvious.



Using the same k=9 method as before, we determined the optimal radius for this data to be around 2.88.

We again found the optimal minimum points to be about 15. On this graph, our implementation DBSCan finds almost all of the colored clusters and misses almost all of the graphed out noise. Specifically, it finds the following centers:

X:96.83566 Y:87.56144

X:49.113903 Y:108.00138

X:35.545044 Y:73.31861

X:124.11486 Y:49.72649

X:26.434252 Y:165.55272

X:114.091774 Y:135.3985

X:113.317856 Y:137.68613

X:70.55266 Y:123.40223

X:74.05511 Y:111.54327

Sources:

<http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf>, p528

<http://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm>

<http://en.wikipedia.org/wiki/DBSCAN>

Test Dataset From:

<http://www.vbforums.com/showthread.php?755167-Density-Based-Spatial-Clustering-of-Applications-with-Noise-%28DBSCAN%29>