Nathan Onaka

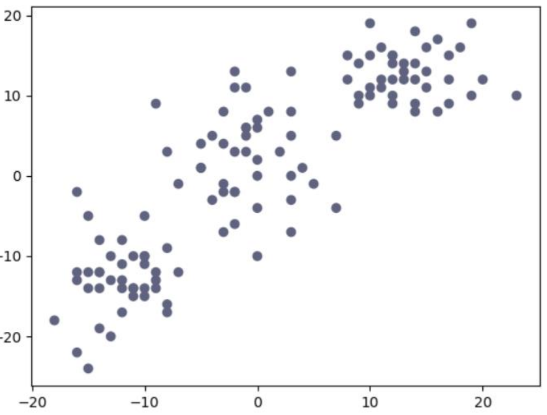
Soft K-Means

Abstract

Soft K-Means is an unsupervised learning algorithm used for classification of data. What sets it apart from the normal K-Means is that it allows for distortion among the classified data points. Each point is weighted heavily towards one cluster center, but also is partially assigned to the rest of the cluster centers. This allows for "fuzzier" classification where the easily deterministic data points are almost entirely classified to one cluster center, and the harder to classify data points are classified as belonging to a mix of cluster centers. Each points level of distortion is determined by the parameter beta(β) found in this formula: exponentiate[-β * d (x, x_c)]. In which we exponentiate – β and multiple by the distance of each point to the cluster centers. This report investigated how different values of β result in higher or lower deterministic classifications for each data point.

It found that a low value of β, .01 resulted in completely unclassified data, in which every data point belonged equally to each clusters center. A high value of β, 10 resulted in almost a Hard K-Means result where each point's classification distribution was 99.1% ± .008% towards a single cluster. And with a value of β where the classification percentage started to converge at .4, where each data point's classification distribution was at 95.73% ± .038% towards a single cluster.

Procedures



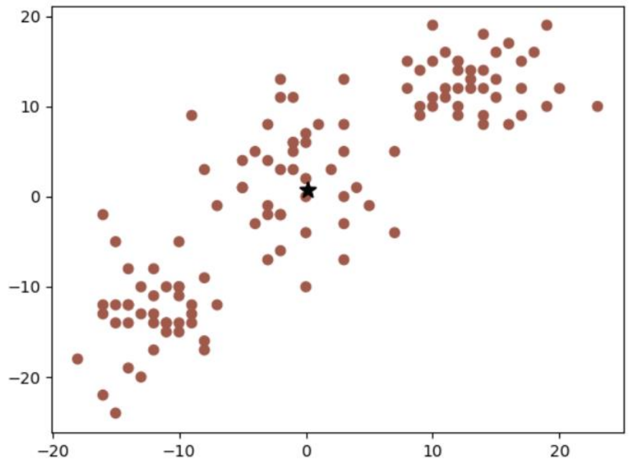
Using Python, a custom data set with 120 two dimensional points was created, with three clusters manually formed, and a few points spread out in between the clusters to help visualization of the classification distribution.

To initialize the centers random points were selected and tested on several trials until more optimal centers were chosen. These initial centers were [0, -20] [10, -10] [20, 0. The initial positions of the centers plays a large role in the classification process, and since this report only seeks to determine the effect of β, the initial center positions were hard coded to remove their influence on the data.

A *betaFormula(centers, points, beta)* method was created to calculate the distortion of each point. A 2D array of the data points, and a blank 2D array of the same size for the distortion values for each point was created. Then using the distortion formula exponentiate[-β * d (x, x_c)], each points distortion value was calculated by exponentiating by -β and then multiplied by the Euclidean distance using numpy's *linalg.norm()* function.

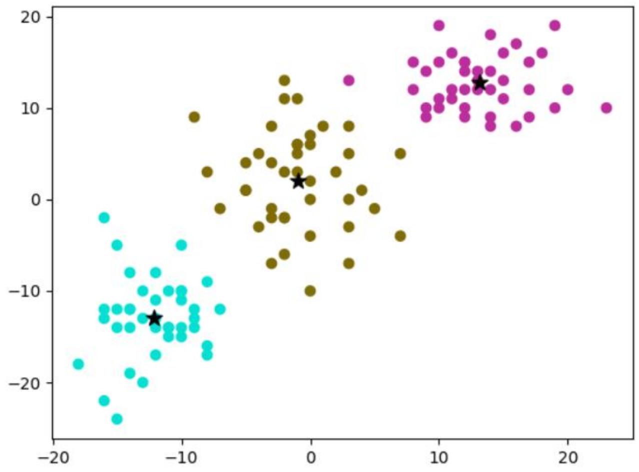
Then an *updateCenters(points, distortion, K)* method was created to update the clusters centers on each loop. The centers were made in an empty 2D array and the distortion values previously calculated were applied by taking the dot product. The updated center positions were then returned.

Each cluster of points had their average distributions calculated and printed to the console so that they could be analyzed and compared to the β values that would be tested. The points, and centers were plotted using *matplotlib.* Random colors were assigned to the clusters and was dot product multiplied by the distortion values to get the visual representation of points assigned to mix cluster centers.



To find the values of β that would be best to test, the outer extremes were first tested. First with a very low value of β, .01, resulting in all the centers beings updated to the same position, and every point being assigned equally to every cluster.

The average distribution for each point was .33% equally assigned to all 3 centers. And all 3 center positions were [0.133, .758].



Next a very high value of β, 10 was selected, resulting in an almost hard KMeans classification. Each point was almost 100% assigned to a single cluster center, and the visual representation looks like 3 totally separate clusters.

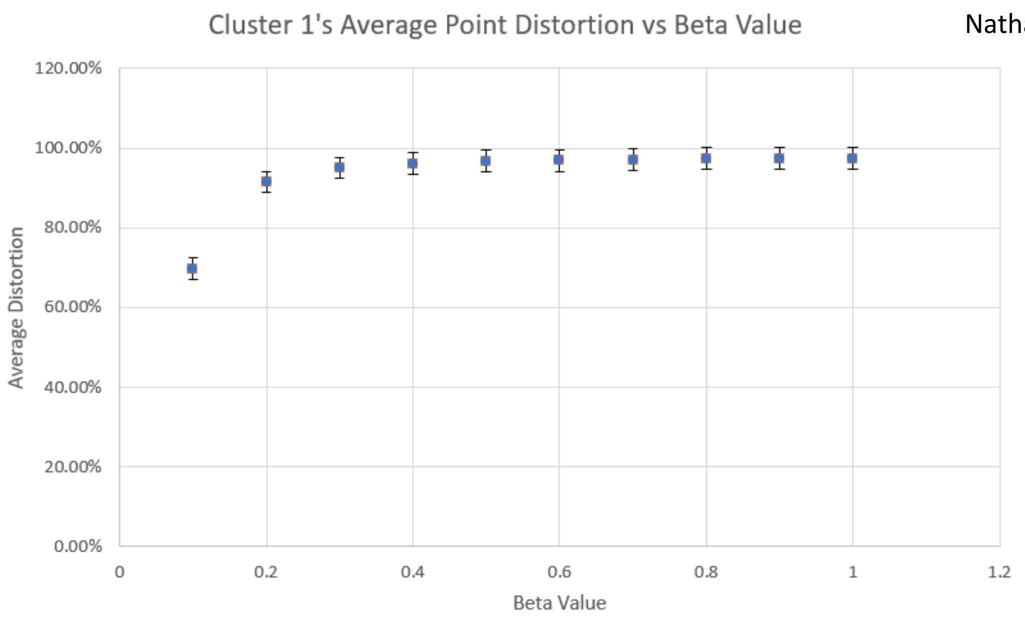
The first center was at [-12.1, -12.9] and the points assigned to it had an average distribution of 97.5%. The second center was at [-0.93, 2.08] and the points assigned to it had an average distribution of 99.9%. The third center was at [13.2, 12.7] and the points assigned to it had an average distribution of 99.9%.

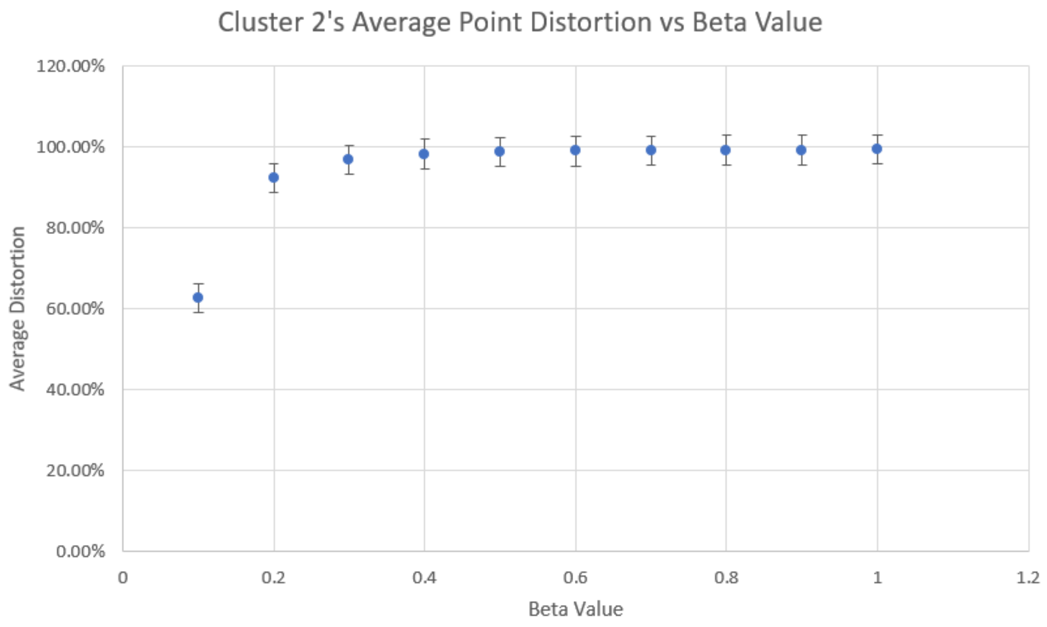
With this information the best values of β for this data set can be found, values as low as .01 and as high as 10 are useless for this test, so the values .1 to 1 will be tested. The best result will be one that has most of the data clearly classified, with just a few points being classified to multiple clusters. The same data is used every time, the initial centers positions are hard coded, and the number of KMeans iterations is constant so only the value of β will affect the results.

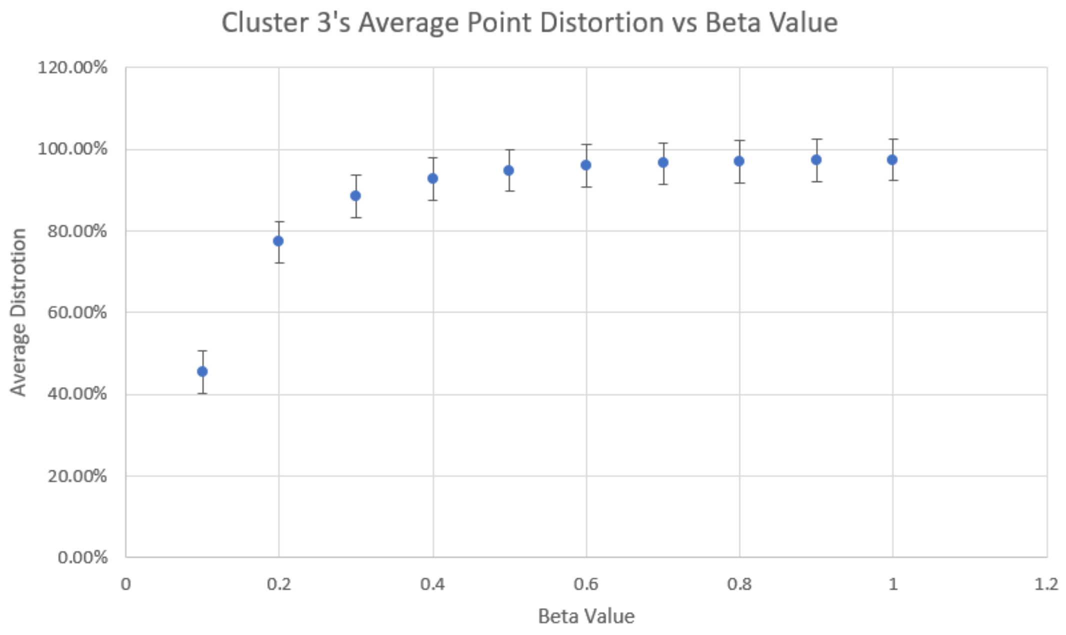
Final Results

The average distortion distribution values for each cluster were printed out on every trial so that they could be analyzed and graphed in comparison to the β value that was set. Here are the graphs for each cluster.

Nathan Onaka

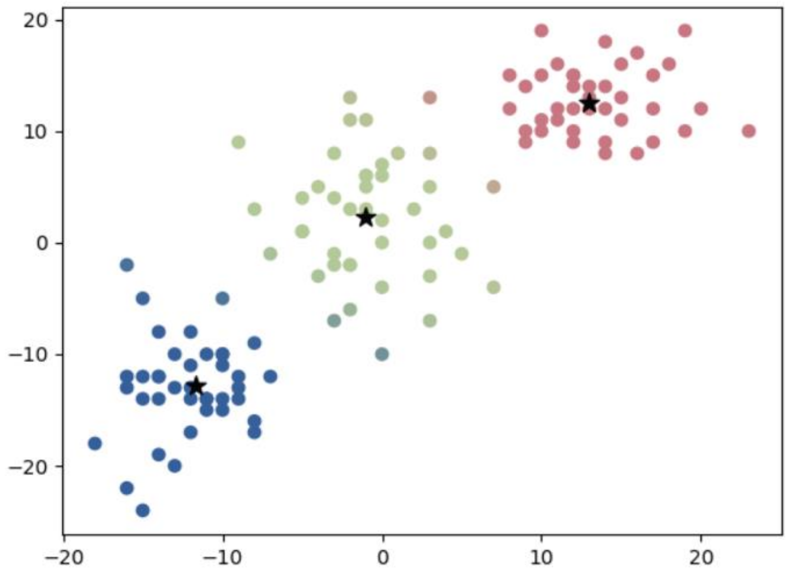






These graphs show that the closer β gets to 1, the closer the distribution of each point to a single cluster approach 100%. For all 3 cluster graphs at the .4 β mark the data starts to normalize and converge with not much of an increase past that point. For the data set used in this test this appears to be the optimal value of β, where most of the data points are clearly classified, and only a few of the hard data points are equally assigned to multiple clusters. At a value of .4 β the average distortion distribution was 95.73% ± .038%.

Here is the visual graph at .4 β.



What I Learned

Soft K-Means is a much more versatile cluster classification algorithm, as you can set its parameters to be just as hard classifying as the normal K-Means, but also as weak as you need. This allows you to fit the data you are testing on more precisely and get a better result. I learned the most while researching how numpy works in regards to 2D arrays and dot product multiplication, using the numpy *.shape* to get the dimensions of the arrays was confusing at first. Numpy also provided the Euclidean distance and exponential functions which were essential. Just like the other classifying algorithms we learned the level of success relies heavily on the data, using random data points or too noisy of data make the soft K-Means not as meaningful. It suffers from the same drawbacks as the normal K-Means if it can’t identify good cluster centers.