CS 145 - Homework 2

Zooey Nguyen
zooeyn@ucla.edu
May 3, 2021

Question 1.

Setup. For future iterations we choose the sets of points, assigning based on which mean μ_k is closer.

$$\mu_1 = (1, 1)$$
$$\mu_2 = (4, 4)$$

Iteration 1. Point (2,3) is equidistant from first means, arbitrarily assign it to cluster S_1.

$$S_1 = \{(1, 1), (1, 2), (2, 0), (2, 2), (2, 3)\}$$
$$S_2 = \{(2, 4), (3, 3), (4, 2), (4, 4)\}$$
$$\mu_1 = (1.6, 1.6)$$
$$\mu_2 = (3.25, 3.25)$$

Iteration 2.

$$S_1 = \{(1, 1), (1, 2), (2, 0), (2, 2)\}$$
$$S_2 = \{(2, 3), (2, 4), (3, 3), (4, 2), (4, 4)\}$$
$$\mu_1 = (1.5, 1.25)$$
$$\mu_2 = (3, 3.2)$$

Iteration 3. Looks like it converged already.

$$S_1 = \{(1, 1), (1, 2), (2, 0), (2, 2)\}$$
$$S_2 = \{(2, 3), (2, 4), (3, 3), (4, 2), (4, 4)\}$$
$$\mu_1 = (1.5, 1.25)$$
$$\mu_2 = (3, 3.2)$$

We aim to prove that K-means algorithm converges in finite iterations.

Proof. Say we have k clusters and we calculate their means μ_i. The configuration will change if there is at least one point in a cluster that is closer to another mean μ_n than its own cluster's mean μ_m. Thus on reassignment it will be clustered with μ_m, which will strictly decrease overall summed distances from their means, because we would be replacing the term in the sum with a shorter distance. So we can track the summed distances:

$$E = \sum_k \sum_{p_i \in S_k} \|\mu_k - p_i\|^2$$

We showed that this will strictly decrease until the configuration cannot find a better mean to reassign a point to. There are a finite number of combinations of points into k clusters. After starting with a certain selection of k means, the average summed distance must strictly decrease with every iteration, and cannot return to a previously explored configuration because that would mean increasing the summed distances, which k-means will not do. Therefore, if the algorithm has finite configurations, strictly decreases configuration cost, and cannot cycle through its configurations, it must converge to a lowest convergence cost in finite time. □

Question 2.

K-Means.

- Pro: It is guaranteed to converge, likely efficiently.
- Con: You have to specify the number of clusters, so you may have to try many values of k to get the lowest cost or most reasonable configuration.
- Con: It cannot handle non-convex groupings, that is, ones that may be locally dense but not radially.

DBSCAN.

- Pro: Automatically finds the number of clusters.
- Pro: Good with non-convexity and arbitrary shapes since it focuses on local densities about points.
- Con: The order of the data in processing can affect the exact groupings.
- Con: If different clusters have very different densities or you need to capture subclusters, DBSCAN cannot capture them well, since it relies on a constant neighborhood size.

BIRCH.

- Pro: Efficient, only has to pass over the data once to group them into their appropriate subclusters by pushing them further and further down into a clustering tree with each step rather than reassigning them into different clusters entirely.
- Con: Clustering features are only defined on numeric data, which means this algorithm can't compactly save categorical data to be considered in the hierarchy.
- Con: Sensitive to the order of the data put in, the branching of the tree depends on the point that came before it.

Algorithm 1 is K-Means while Algorithm 2 is DBSCAN. Algorithm 1 clearly bases clusters on absolute distance to some centroid, as the shapes of the clusters it creates are all somewhat round, even if there are gaps in the local density of the cluster it settles on as seen in Dataset 2. In Algorithm 2 we can see that points that are tightly packed together are most likely to be clustered, seeing it clustered arbitrarily-shaped clusters of similar density in Dataset 2. Algorithm 2 performed well generally on Dataset 1, but we can see that some of the points in between the clusters have random categorisations, which may be due to the fact that they have been cast as edge or outlier points, more arbitrarily assigned because it is assumed that the different clusters are very separable, which is not as much the case for the edge points.

There may be a way to fix it by hand. One is to increase the neighborhood size slightly so that the points between the green and yellow cluster are density reachable to each other and count as a single cluster, but you have to be careful to not increase it so much that the blue group also gets reached and the whole thing gets clumped into one. A more hacky fix would be to insert a dummy datapoint right in between that little gap between the green and yellow clusters to ensure that we don't have to change the neighborhood size, keeping the blue group as separated as it is now while using my own inference to assume that it's likely that there'd be a point populating the top of the green/yellow arc. Yet another hacky way is to change which point that clustering starts on – it may have been the case that the top three green points were explored left-to-right, each being counted as edge points until the second point simply couldn't reach enough, but if we started at the very top point, we might be able to reach a yellow point and include that in the neighborhood directly. I don't think this is likely to fix it, but when margins are this small you can try.

Question 3.

Entropy of each node.

$$\begin{aligned}
H_{11} &= -\frac{300}{400}\log\frac{300}{400} - \frac{100}{400}\log\frac{100}{400} \\
&= 0.811 \\
H_{12} &= -\frac{100}{400}\log\frac{100}{400} - \frac{300}{400}\log\frac{300}{400} \\
&= 0.811 \\
H_{21} &= -\frac{400}{600}\log\frac{400}{600} - \frac{200}{600}\log\frac{200}{600} \\
&= 0.918 \\
H_{22} &= -\frac{1}{1}\log\frac{1}{1} \\
&= 0
\end{aligned}$$

Information gain for split on each attribute. The information gain on the second attribute is higher, so you would prefer the split on the $\boxed{\text{second attribute}}$ first.

$$\begin{aligned}
H(Gender) &= -\frac{400}{800}\log\frac{400}{800} - \frac{400}{800}\log\frac{400}{800} \\
&= 1 \\
H(Gender|Attr_1) &= H_{11}P_{11} + H_{12}P_{12} \\
&= (0.811)\frac{300+100}{800} + (0.811)\frac{100+300}{800} \\
&= 0.811 \\
I(Attr_1) &= H(Gender) - H(Gender|Attr_1) \\
&= 0.189 \\
H(Gender|Attr_2) &= H_{21}P_{21} + H_{22}P_{22} \\
&= (0.918)\frac{400+200}{800} + (0)\frac{200}{800} \\
&= 0.688 \\
I(Attr_2) &= H(Gender) - H(Gender|Attr_2) \\
&= 0.312
\end{aligned}$$

Gain ratio for split on each attribute. The gain ratio on the second attribute is higher, so you would prefer the split on the $\boxed{\text{second attribute, yet again}}$.

$$\begin{aligned}
G(Attr_1) &= H(Gender)/H(Gender|Attr_1) \\
&= 1.23 \\
G(Attr_2) &= H(Gender)/H(Gender|Attr_2) \\
&= 1.45
\end{aligned}$$

Question 4.

Naive Bayes values. Assume the given event for each feature is Y where the event $notF_i$ is N since they all take on only two values.

$$\begin{aligned}
P(F_1|A) &= \frac{4}{9} \\
P(F_2|A) &= \frac{6}{9} \\
P(F_3|A) &= \frac{8}{9} \\
P(F_1|B) &= \frac{6}{10} \\
P(F_2|B) &= \frac{2}{10} \\
P(F_3|B) &= \frac{2}{10} \\
P(A) &= \frac{9}{19} \\
P(B) &= \frac{10}{19}
\end{aligned}$$

This testing sample would be classified to maximise the score on each class S_{class}. The score for class B is higher so we would classify this point as $\boxed{\text{class B}}$.

$$\begin{aligned}
S_{class} &= \sum_i P(F_i = given|class)P(class) \\
S_A &= P(F_1|A)P(A) + P(F_2|A)P(A) + P(notF_3|A)P(A) \\
&= \frac{9}{19}\left(\frac{4}{9} + \frac{6}{9} + \frac{1}{9}\right) \\
&= \frac{11}{19} \\
S_B &= P(F_1|B)P(B) + P(F_2|B)P(B) + P(notF_3|B)P(B) \\
&= \frac{10}{19}\left(\frac{6}{10} + \frac{2}{10} + \frac{8}{10}\right) \\
&= \frac{16}{19}
\end{aligned}$$