Lab 3 Unsupervised Learning Discussion

Michael Tripp
CS 614 Machine Learning

May 2, 2023

1 Introduction

In this lab, we were tasked with programming two unsupervised learning algorithms and analyzing the results. The first is an agglomerative clustering algorithm and the second is a DBSCAN algorithm. Each use euclidean distance as the distance function and models are built given 150 unlabeled data samples of three different, evenly distributed specifications of flowers. In Figure 1 below, this data is visualized. In this paper, you can find analysis of each of the models. Note that while the data is 4-dimensional, we only plot various subsets of two of the four features to provide some visual representation of the data in a 2-dimensional plane. Assume we are using the first and second features as the x and y axes unless otherwise noted.

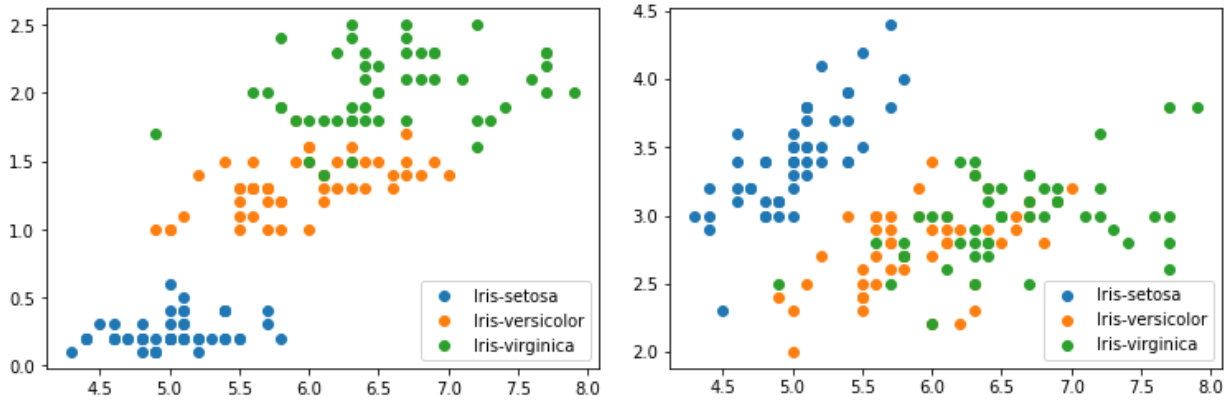

Figure 1 *Plots of the original, labeled data in two different planes. On the left, we use the first and second features as axes while on the right we use the first and fourth features as axes.*

2 Agglomerative Clustering

For the agglomerative clustering model, I took two partitions of cluster groups, one when there were exactly three clusters, and another using the dendrogram approach, or the partition that spends the longest amount of time at any given number of clusters.

In Figure 2 below, we can see the first partition with three clusters, each denoted with a different color. While this may be three clusters, these are clearly not three evenly distributed groups as we would expect from the data. The blue and orange groups are clearly defined, but the green group seems to be more of an

outlier group. Noting the size of the orange cluster, we could speculate two of the flower groups are more closely related than the blue cluster, so the model just combined them into one large cluster.

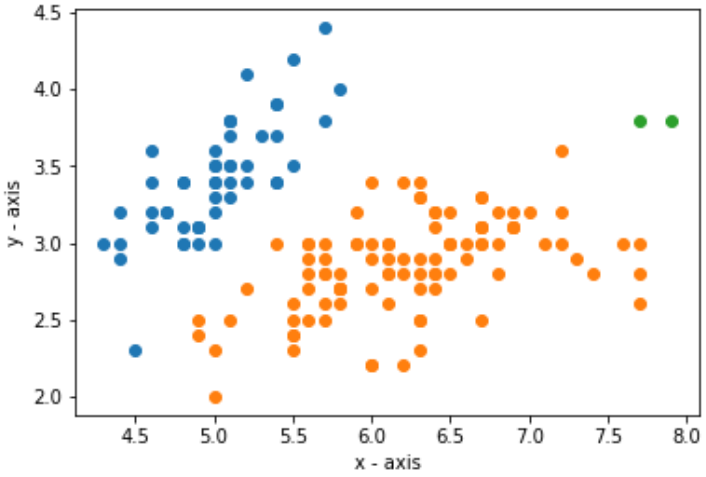

Figure 2 *Partition when stopped at three clusters.*

In Figure 3, we can see the second partition. As we can see, this partition only contains two clusters. Since this partition was selected using the dendrogram approach, this means it took the greatest amount of time to merge these last two clusters than any other cluster merge. This makes sense considering the distance between the two clusters, although, again, not very helpful if we wanted three somewhat evenly distributed groups as seen in our plots above.

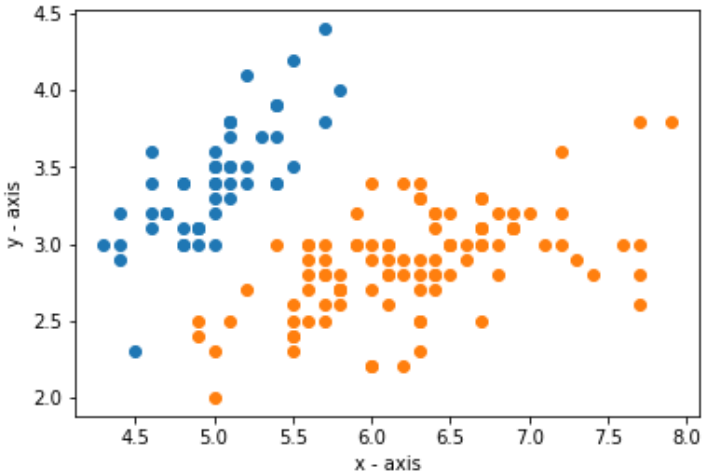

Figure 3 *Partition made using dendrogram approach.*

3 DBSCAN

Next, we will look at the DBSCAN model. DBSCAN is nice because it not only helps us find clusters, but also outliers! For this model, we will look at results across various parameter specifications. The two primary parameters we will tweak for DBSCAN are *minimum features per cluster* (minPts) and *search distance* (ε).

For my first test, I set minPts = 5 and $\varepsilon = 0.6$, seen in Figure 4 below. As we can see, we get two main cluster groups and in grey a whole bunch of outliers! Visually, these outliers made sense, even across various different feature axes.

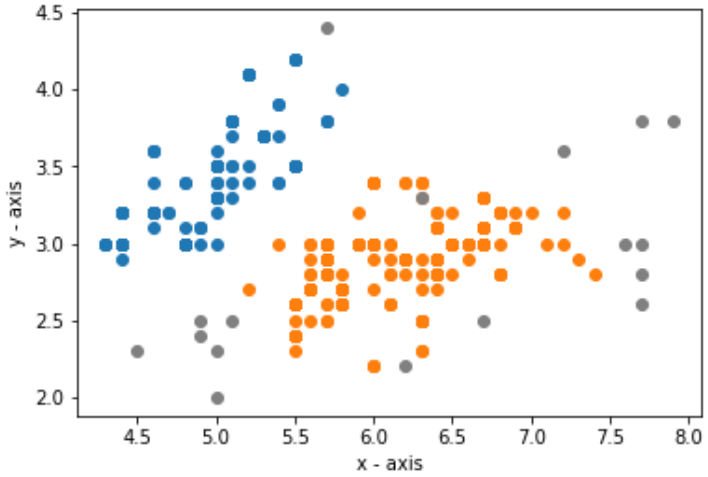

Figure 4 *Model built with minPts = 5 and $\varepsilon = 0.6$.*

Next, I tried decreasing the search distance to see what effect it would have on the clusters. In Figure 5a, we can see that decreasing ε to 0.5 really breaks up the orange cluster as the search distance is not great enough to bridge the gap between the smaller clusters. We also get more outliers as many of the outer points are now no longer close enough to be joined into any cluster at all.

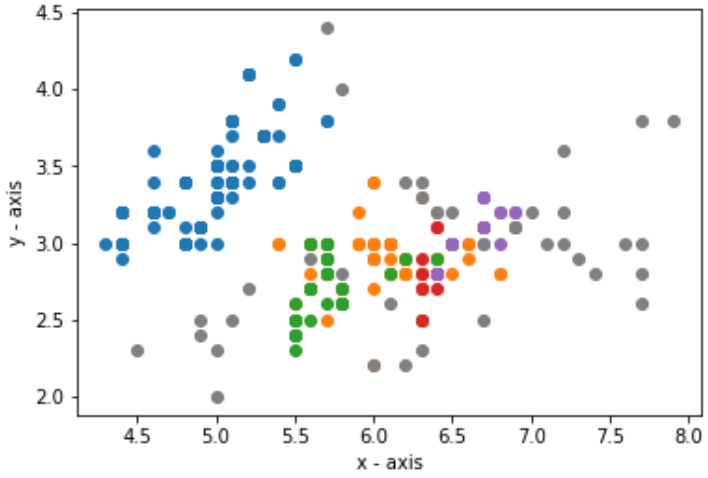

Figure 5a *Model built with minPts = 5 and $\varepsilon = 0.5$.*

We can see these smaller clusters split up a bit better visually in Figure 5b using the first and fourth features as axes instead.

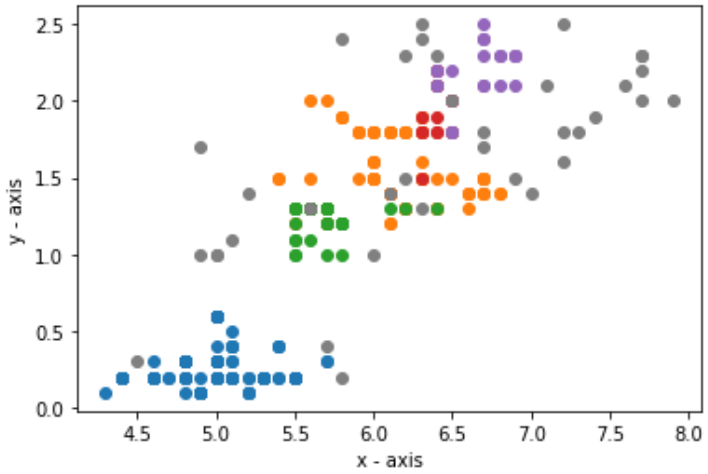

Figure 5b *Model built with same parameters as Figure 5a, but using the first and fourth features to plot the axes*

Whereas decreasing ε in Figure 5 resulted in more clusters, we can see in Figure 6 that increasing it too far results in all those non-blue smaller clusters to be merged back into one large cluster. Even the outliers were now close enough to be included. Ideally, however, we'd like three clusters, not two, and could benefit from finding some outliers, so this distance is probably too high.

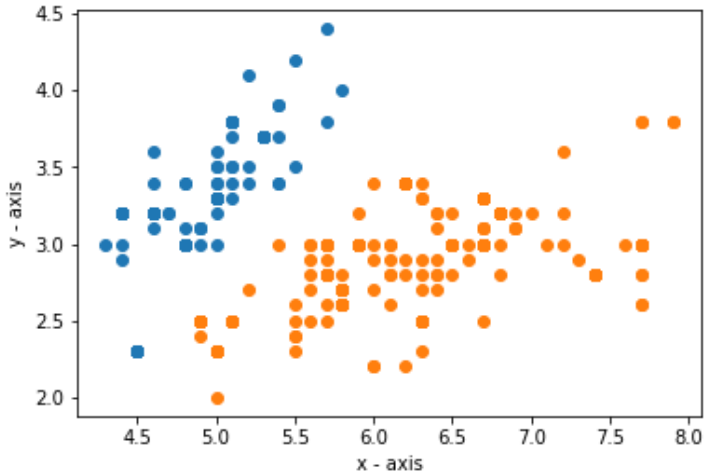

Figure 6 *Model built with minPts = 5 and ε = 1.*

As a search distance of 0.6 seemed to give pretty good results, I decided to go back to $\varepsilon = 6$ and try different minPts values. In this next model, I decided to try 10 minimum points per cluster, and as we can see in Figure 7, we get very nice results! We can observe that the large orange group seen prior is now split up into two primary clusters which are somewhat evenly distributed — exactly what we wanted! We do lose a good few points as outliers, but I think visually we can see this is a fair conclusion.

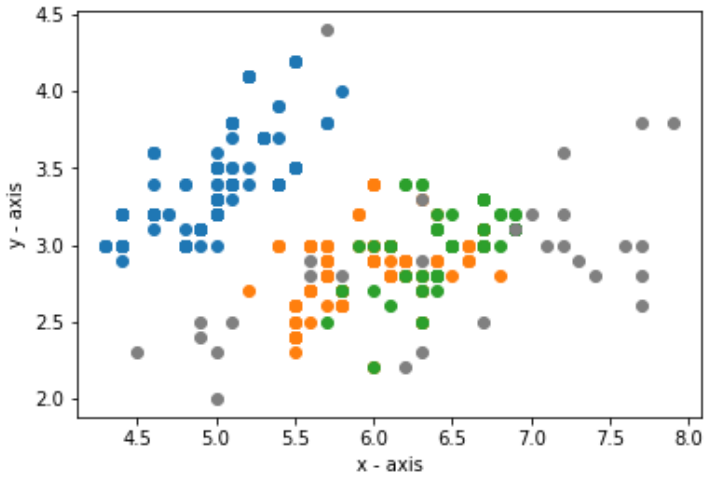

Figure 7a *Model built with minPts = 10 and $\varepsilon = 0.6$.*

Plotting this with the first and fourth features instead in Figure 7b, we can see these clusters even clearer. Furthermore, we can verify these are reasonable clusters as well since besides some outliers, these are pretty much the same clusters as our labeled plot above!

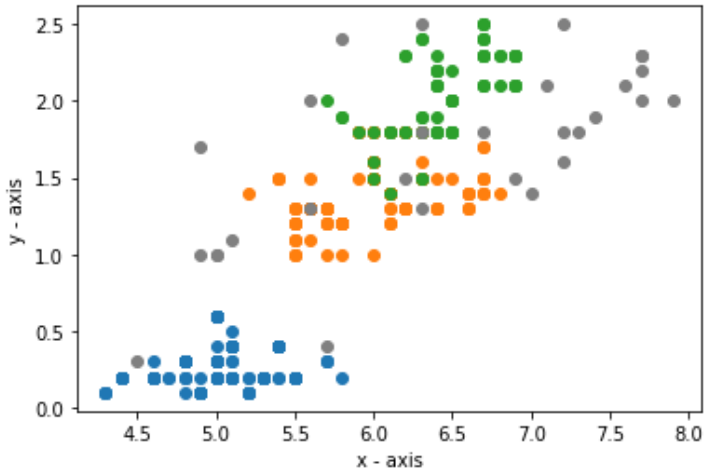

Figure 7b *Model built with same parameters as Figure 7a, but using the first and fourth features to plot the axes.*

We can see in Figure 8, however, that increasing the mininmum number of points per cluster by too much yields much poorer results. In this model, much of the non-blue colored points become outliers because there are not enough clusters of points within the search distance to label very many core points outside of the center. Increasing minPts to 20 continues this trend, unsurprisingly, forcing all the non-blue colored points to become outliers.

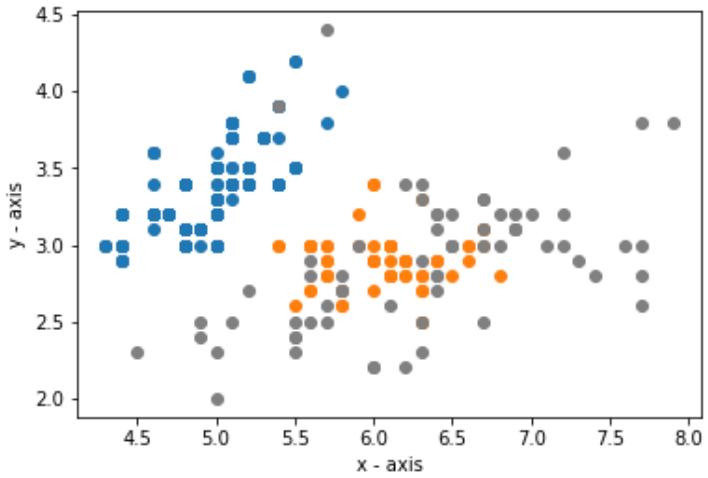

Figure 8 *Model built with minPts = 15 and $\varepsilon = 0.6$.*

After this, I was curious to see if I could get better results using a high value for minPts by changing my search distance. This experimentation led me to the model seen in Figure 9 below, plotted with the first and fourth features again. Interestingly enough, I was actually able to get very similarly looking results to Figure 7 (and very similar to our original labeled data!). Once again, we are able to get three somewhat evenly distributed clusters, although the orange cluster is noticeably more dominant than before, causing it to not fit quite as well to our original data clusters. This result is particularly interesting, however, because it is sort of unstable. Any higher or lower than 0.65 and we do not see a third cluster show up at all, whereas our margins were a bit larger previously.

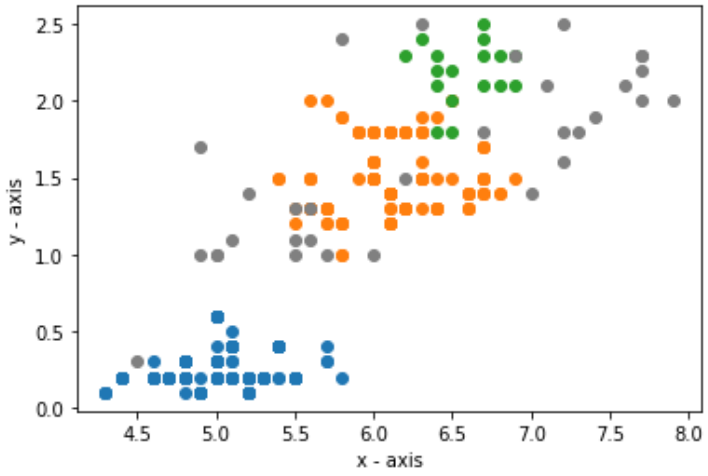

Figure 9 *Model built with minPts = 15 and $\varepsilon = 0.65$.*

To build on this idea, I decided to revist a search distance value I found to yield poor results in the past and see if I could find a minPts value that would work relatively well with it. As a search distance of 0.5 previously gave me too many clusters, I decided to reuse that value. The best results I could find resulted from a minPts value of three, as seen in Figure 10. While there may still be one too many clusters in this model, we actually do get fairly nice results! We still have three somewhat evenly distributed clusters (in

blue, orange, and green), despite the additional cluster. As such, it may not be as good as some of the previous parameter combinations, but it proves there are many ways to get decent results!

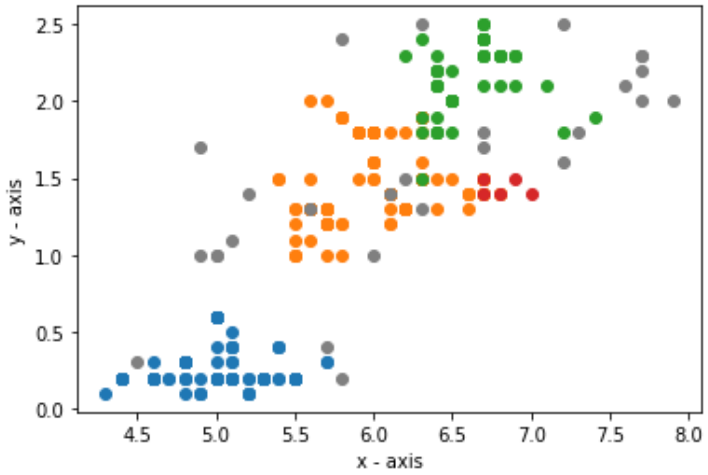

Figure 10 *Model built with minPts = 3 and $\varepsilon = 0.5$.*

For these last two models, I wanted to try some very different parameter values just to see what sorts of different output I can get, even if they aren't good results. For the first, I tried setting minPts to 1 and epsilon to 0.4 so that clusters are essentially fully dependent on search distance, not number of points. This can be seen below in Figure 11. This model clearly yields poor results, but it is interesting to see the various small clusters that result. Even the blue-colored cluster split off into one or two other small clusters.

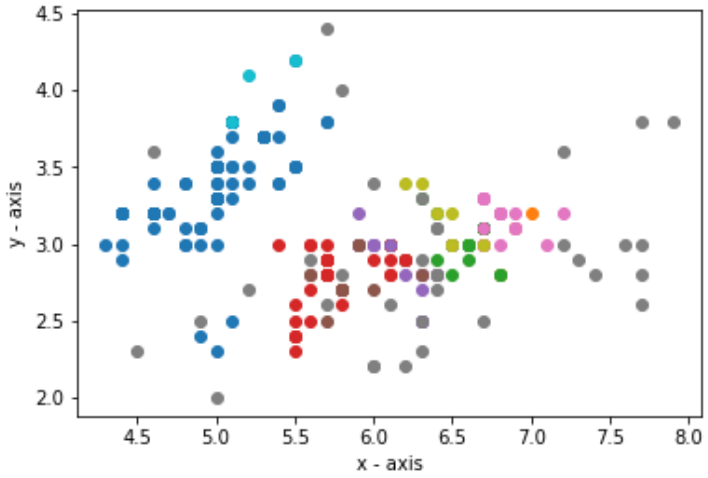

Figure 11 *Model built with minPts = 1 and $\varepsilon = 0.4$.*

This last one in Figure 12 is not particularly important, but I thought it was interesting to see definitively how dense the blue-colored cluster is, as it is the only set of points that contains enough data points within a small distance of 0.4 to be called a cluster.

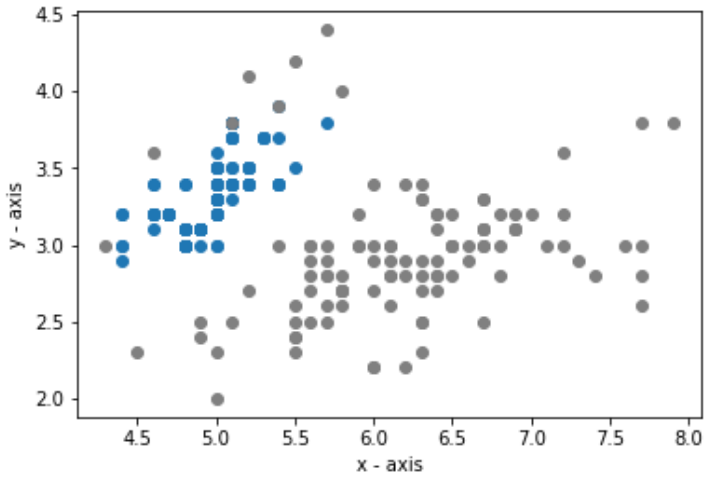

Figure 12 *Model built with minPts = 10 and $\varepsilon = 0.4$.*

4 Time Complexity

The time complexities of each of my algorithms in terms of number of observations n is given as follows:

Agglomerative clustering: $\mathcal{O}(n^3)$

DBSCAN: $\mathcal{O}(nlogn)$