In-class Lab 9b: Unsupervised Learning - Clustering

Submit Assignment

Due Monday by 11:59pm **Points** 1 **Submitting** a file upload **File Types** pdf

Read the line-by-line explanation of today's lab. Then answer these questions:

1) Is the following True or False? Explain why in either case.
In the clustering example of today's lab we *predict* the color of the data points. Then the plot of line 21 shows which points are *misclassified*.

2) After looking at lines 19, 20, 21, can you tell what does the "col" parameter specify?

3) For how many points in the plot does K-means disagree with our initial cluster assignment?

Explanation of today's lab.

Clustering works in all dimensions. But in this lab we show it in 2 dimensions, in order to look at the classes and see how well it is done. We make some fake data for purposes of demonstration.

Line 7: We set the seed using an arbitrary number.

Line 8: We make a matrix of random normals (100 observations, 2 columns). Try dim(x) after line 8. That creates a cloud of Gaussians.
What we want to do, is to create 4 clusters. So we generate some means and displace some of the Gaussians by shifting their means around for 4 clusters.

Line 9: How to do that? We make another random normal matrix, xmean, with 8 elements (2 means for each cluster, since x has two columns, x1 and x2). Also, the st.deviation of the random normals is set to be 4, so that these means are shifted around much more than what the actual data did.

Line 10: Then we decide which of the rows get which means (cluster centroids). So we pick a random sample from 1 to 4, 100 times for 100 rows.
Why do we want to do that? Because in clustering, there is no response variable and we cannot judge how well the clustering algorithm will do. Hence, in this code we will manually assign some default clusters to the points and claim that this is how the data should be clustered by the algorithm.

Line 11: Then we add the appropriate mean to the appropriate rows. This is a typical R vector operation that is nice and tricky, and effective.

xmean is a 4-row matrix. We are indexing it with 100 indices through "which". That will produce a 100 row matrix with 2 columns. It will pick out the right mean for each row. And we just add it to the original x.

Line 12: We make a plot for all points, after we have displaced them as above. We set their color to match the "which" cluster that was manually assigned to each point. Colors tell us which cluster it belongs to (it uses colors 1-4: black, red, green, blue). You can see the data and the clusters that we have imposed.

We know the clusters but now we will hand in this data to K-means and not tell it what the clusters are, it is supposed to find them automatically.

Line 17: We tell it we have 4 clusters here. We also tell it to do 15 random starts. Random starts are cheap. K-means clustering starts with randomly selecting initial clusters centroids, and of course if we pick a bad random start, we will end up with a bad solution. (It is rather unlikely, though, in this toy data example!)

Line 18: We print out the result. The cluster vector shows which cluster each data point belongs to, according to the findings of the K-means algorithm that was just run. We can also see the within-cluster sums of squares.

The between (sum of squares / total sum of squares) is the R^2 for clustering; it shows how much variance is explained. It has done a good job here.

Line 19: We plot the data with big circles for each point. The coloring here is not the original clustering that we had imposed when we generated the data. These are cluster assignments found by the k-means clustering algorithm.

Line 20: To see how well it did in terms of the original known clusters, we can include these points and put the original points inside.

We can see we have a color mismatch. We named the clusters 1-4 in some order.

K-means clustering algorithm found out 4 clusters and it named them between 1-4 but the ordering is arbitrary. So it picked a different order.

Line 21: We can look at the clusters here and re-assign the colors.

Now we see a more useful plot: the outer circles show the k-means cluster id and the inner points show the original cluster id.

You can see the mismatches; a black point was assigned the green cluster and a blue point was assigned the black cluster. Quite reasonably of course, as the data points look closer to the wrongly assigned clusters.

The final plot should be **this** .

Troubleshooting! If you are getting different color assignment, make sure that you start your session over by removing all objects. Type ls(), then rm(object), where object can be x, xmean, etc.