Clustering

Clustering is an unsupervised learning method, meaning that clustering is not used to predict an outcome, or dependent variable. The main goal is to segment a set of observations into similar groups, based on the available data. However, although clustering is not designed to predict anything, clustering can be useful

- to improve the accuracy of predictive models,
- as a stand-alone tool to get insight into data distribution,
- as a preprocessing (or intermediate) step for other algorithms

The data can be clustered into similar groups, which may add structure to the input data, e.g. in the form of new derived attributes. Clusters can also be used to build a predictive model for each group. We saw an example of this in the Patterns of Heart Attacks lecture.

Before a clustering algorithm can be applied, the distance metric that will be used to define the distance between two points needs to be selected. A typical choice is euclidean distance. This is discussed more in relation to the specific clustering algorithms below. Additionally, the data often needs to be normalized, to make sure that some variables will not dominate others in the distance calculation. This can be done by subtracting the mean and dividing by the standard deviation for each variable.
There are many different algorithms for clustering, which differ in how the clusters are built and the properties of the clusters produced from the method. In this class, we cover Hierarchical clustering and K-means clustering, two of the most popular clustering methods.

Hierarchical Clustering

Hierarchical clustering starts with each observation in its own cluster. So if you have n data points, or observations, you start with n clusters. Then the distance between each pair of clusters (or points) is computed, and the two closest clusters are combined (resulting in n-1 clusters). This is repeated until all of the data points are in a single cluster.

There are many different ways to define point distance and cluster distance in hierarchical clustering. A common point distance is euclidean distance, and a common cluster distance is centroid distance.

Hierarchical clustering can be visualized with a dendrogram. In a dendrogram, each of the points is shown on the bottom, and the process by which they are combined is shown with lines in the plot. The height of the lines approximately shows how far apart the two clusters were when they were combined.

Hierarchical clustering is done without first selecting the number of clusters desired. This is nice because you can run the algorithm without having to know the appropriate number of clusters for your problem. However, in many applications, you need to ultimately have a specific number of clusters. This can be done after the hierarchical process is complete by looking at the dendrogram. You should imagine a horizontal line cutting across the dendrogram. The number of vertical lines of the dendrogram that your horizontal line crosses shows how many clusters would be selected. That number of clusters tends to be a good choice if the horizontal line has a lot of "wiggle room", meaning that the horizontal line can move up and down without running into a horizontal line of the dendrogram. However, when picking the number of clusters for a particular problem, you should also consider how many clusters make sense for the particular application.

K-Means Clustering

Instead of forming the clusters in a hierarchical way, k-means clustering iteratively re-assigns data points to clusters. First, the number of clusters desired has to be selected. It is often useful to run hierarchical clustering first to better understand which number of clusters would be good, or to think about the number of clusters that would be appropriate for the given problem. You can also run the clustering several times with a different choices for the number of clusters. The results can be visualized with a scree plot, which plots the number of clusters along the x-axis, and the sum of squares within each cluster (or another variance metric) on the y-axis. This shows for different choices of the number of clusters how much variance there is within the clusters. There is often an "elbow" in the line, which indicates a good choice for the number of clusters.

Once the number of clusters has been selected, each data point is randomly assigned to one of the clusters and each cluster centroid is computed. Then each data point is re-assigned to the closest cluster centroid, and the cluster centroids are re-computed. This process repeats until no improvement can be made, or no data point needs to be re-assigned. It can also be stopped by defining a maximum number of iterations.

The largest benefit of the k-means clustering algorithm is that it is very fast, and works with small and large datasets. This is not true for hierarchical clustering, because of the distance computations required for hierarchical clustering. Hierarchical clustering will not work if the dataset is too large, so in some situations, hierarchical clustering might not be an option.

Regardless of the clustering algorithm you use, you should always analyze and explore your resulting clusters to see if they are meaningful. This can be done by looking at basic statistics in each cluster, like the mean, minimum, and maximum values in each cluster and each variable. You can also check to see if the clusters have a feature in common that was not used in the cluster, like an outcome variable. If they do, then this often indicates that your clusters might help improve a predictive model.

Clustering in R

Suppose your dataset is called "DataFrame", which consists of the following three independent variables: "IndependentVar1", "IndependentVar2", and "IndependentVar3".

These variables can be normalized using the "caret" package and the following commands:

```
preproc = preProcess(DataFrame)
DataFrameNorm = predict(preproc, DataFrame)
```

Then the normalized dataset is called DataFrameNorm, which we will use for clustering.

Then you can construct a hierarchical clustering model based on these three variables with the following commands:

```
HierDistances = dist(DataFrameNorm, method = "euclidean")
HierClustering = hclust(HierDistances, method = "ward.D")
```

We used euclidean distance for the points, and Ward's method for the clustering, but other methods can be used as well. You can plot the cluster dendrogram using the plot function:

```
plot(HierClustering)
```

Once you have decided on a number of clusters to use (we'll use five clusters here), you can assign observations to cluster groups using the cutree function:

```
HierClusterGroups = cutree(HierClustering, k = 5)
```

And then using these assignments, you can analyze the cluster centroids using the tapply function:

```
tapply(DataFrameNorm$IndependentVar1, HierClusterGroups, mean)
tapply(DataFrameNorm$IndependentVar2, HierClusterGroups, mean)
tapply(DataFrameNorm$IndependentVar3, HierClusterGroups, mean)
```

Or, you can analyze the cluster centroids with the following more advanced single command:

```
lapply(split(DataFrameNorm, HierClusterGroups), summary)
```

To create a k-means clustering model, you can use the kmeans function (we'll create five clusters):

```
KMeansClustering = kmeans(DataFrameNorm, centers = 5)
```

Then the assignment of observations to cluster groups is an attribute of the KMeansClustering object:

```
KMeansClusterGroups = KMeansClustering$cluster
```

And then just like for hierarchical clustering, you can analyze the cluster centroids using the tapply function:

```
tapply(DataFrameNorm$IndependentVar1, KMeansClusterGroups, mean)
tapply(DataFrameNorm$IndependentVar2, KMeansClusterGroups, mean)
tapply(DataFrameNorm$IndependentVar3, KMeansClusterGroups, mean)
```