**Clustering**

**What is Clustering?**

* Clustering refers to a very broad set of techniques for finding subgroups, or clusters, in a data set.
* We seek a partition of the data into distinct groups so that the observations within each group are quite similar to each other.

**Applications of Clustering** Market segmentation, social network analysis

**Example**

* Suppose we have access to a large number of measurements (e.g. median household income, occupation, distance from nearest urban area, and so forth) for a large number of people.
* Our goal is to perform market segmentation by identifying subgroups of people who might be more receptive to a particular form of advertising, or more likely to purchase a particular product.
* The task of performing market segmentation amounts to clustering the people in the data set.

**Methods**

1. **K-means clustering:** we seek to partition the observations into a pre-specied number of clusters.
2. **Hierarchical Clustering:** we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a dendrogram, that allows us to view at once the clustering obtained for each possible number of clusters, from 1 to n.

**K-Means Algorithm**

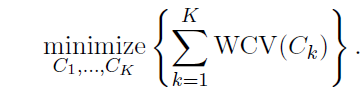
**Input**:

-­‐ K (number of clusters)

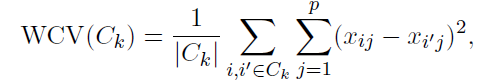
-­‐ Training set

**Details of K-means clustering**

* The idea behind K-means clustering is that a good clustering is one for which the within-cluster variation (WCV) is as small as possible.
* The within-cluster variation for cluster is a measure WCV() of the amount by which the observations within a cluster differ from each other.
* Hence we want to solve the problem



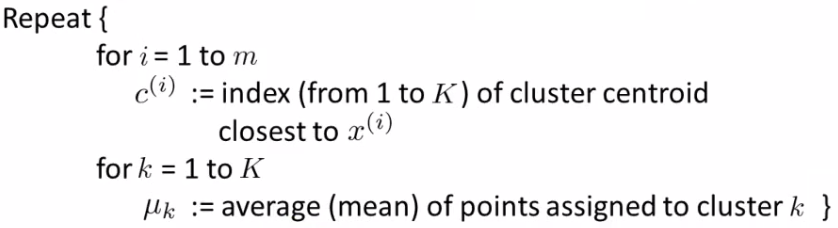
* In words, this formula says that we want to partition the observations into K clusters such that the total within-cluster variation, summed over all K clusters, is as small as possible.
* Typically we use Euclidean distance



where denotes the number of observations in the kth cluster.

**Algorithm**

Randomly initialize K cluster centroids



Repeat the procedure until the variation within each cluster stops decreasing significantly or cluster remains same.

Loop 1

This inner loop repeatedly sets the  variable to be the index of the closest cluster centroid

i.e. take  example, measure Euclidean distance to each cluster centroid, assign c(i)to the cluster closest, suppose it is closest to cluster 1 then:

Loop 2

Loops over each centroid calculate the average mean based on all the points associated with each centroid from c(i)

#choose optimal K

**R Code**

Exploring the data

The iris dataset contains data about sepal length, sepal width, petal length, and petal width of flowers of different species. Let us see what it looks like:

library(datasets)

head(iris)

*Sepal.Length Sepal.Width Petal.Length Petal.Width Species*

*1 5.1 3.5 1.4 0.2 setosa*

*2 4.9 3.0 1.4 0.2 setosa*

*3 4.7 3.2 1.3 0.2 setosa*

*4 4.6 3.1 1.5 0.2 setosa*

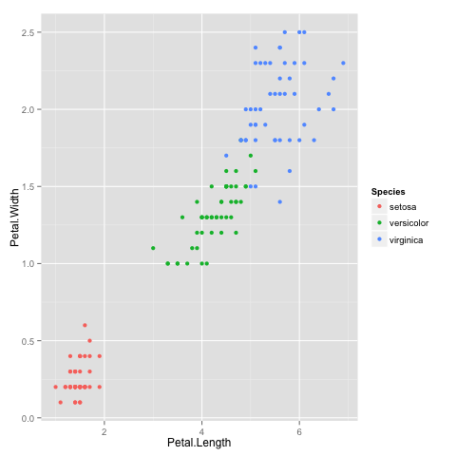
*5 5.0 3.6 1.4 0.2 setosa*

*6 5.4 3.9 1.7 0.4 setosa*

After a little bit of exploration, I found that Petal.Length and Petal.Widthwere similar among the same species but varied considerably between different species, as demonstrated below:

library(ggplot2)

ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom\_point()

Here is the plot:  
[](http://i2.wp.com/datascienceplus.com/wp-content/uploads/2015/12/plot1.png)

Clustering

Okay, now that we have seen the data, let us try to cluster it. Since the initial cluster assignments are random, let us set the seed to ensure reproducibility.

set.seed(20)

irisCluster <- kmeans(iris[, 3:4], 3, nstart = 20)

irisCluster

*K-means clustering with 3 clusters of sizes 46, 54, 50*

*Cluster means:*

*Petal.Length Petal.Width*

*1 5.626087 2.047826*

*2 4.292593 1.359259*

*3 1.462000 0.246000*

*Clustering vector:*

*[1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3*

*[35] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2*

*[69] 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1*

*[103] 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 2 2 1 1 1 1 1 1 1 1*

*[137] 1 1 2 1 1 1 1 1 1 1 1 1 1 1*

*Within cluster sum of squares by cluster:*

*[1] 15.16348 14.22741 2.02200*

*(between\_SS / total\_SS = 94.3 %)*

*Available components:*

*[1] "cluster" "centers" "totss" "withinss"*

*[5] "tot.withinss" "betweenss" "size" "iter"*

*[9] "ifault"*

Since we know that there are 3 species involved, we ask the algorithm to group the data into 3 clusters, and since the starting assignments are random, we specify nstart = 20. This means that R will try 20 different random starting assignments and then select the one with the lowest within cluster variation.  
We can see the cluster centroids, the clusters that each data point was assigned to, and the within cluster variation.

Let us compare the clusters with the species.

table(irisCluster$cluster, iris$Species)

*setosa versicolor virginica*

*1 0 2 44*

*2 0 48 6*

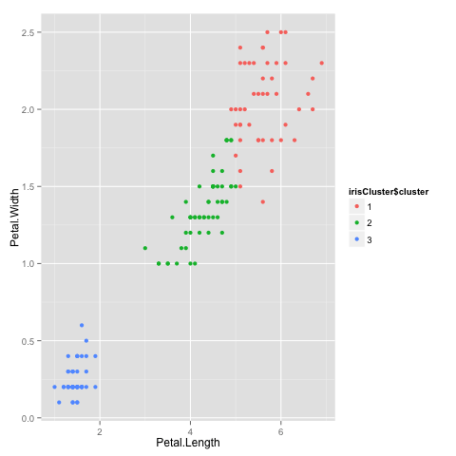
*3 50 0 0*

As we can see, the data belonging to the setosa species got grouped into cluster 3, versicolor into cluster 2, and virginica into cluster 1. The algorithm wrongly classified two data points belonging to versicolor and six data points belonging to virginica.

We can also plot the data to see the clusters:

irisCluster$cluster <- as.factor(irisCluster$cluster)

ggplot(iris, aes(Petal.Length, Petal.Width, color = iris$cluster)) + geom\_point()

Here is the plot:  
[](http://i2.wp.com/datascienceplus.com/wp-content/uploads/2015/12/plot2.png)