K-means Clustering

-Applied Multivariate Analysis & Statistical Learning-

MMA Chapter 15.4.1, ISLR 10.3.1

Lecturer: Darren Homrighausen, PhD

Preamble:

- State K-means
- Discuss how it can be computed
- Define a way of choosing the number of clusters

Clustering Introduction

We will focus on two particular clustering algorithms

- K-means: Seeks to partition the the observations into a pre-specified number of clusters.
- Hierarchical: Produces a tree-like representation of the observations, known as a dendrogram.

There are advantages (disadvantages) to both approaches.

We can cluster observations on the basis of the features in order to find subgroups of observations.

Just as easily, we can find clusters of features based on the observations to find subgroups in the features.

We will focus on clustering the observations. You can cluster features by transposing $\mathbb{X}$ (that is, clustering on $\mathbb{X}^\top$).

K-means

1. Select a number of clusters K.
2. Let $C_1, \ldots, C_K$ partition $\{1, 2, 3, \ldots, n\}$ such that
   - All observations belong to some set C_j.
   - No observation belongs to more than one set.
3. K-means attempts to form these sets by making within-cluster variation, $W(C_k)$, as small as possible.

$$\min_{C_1, \ldots, C_K} \sum_{k=1}^{K} W(C_k).$$

4. To Define W, we need a concept of distance. By far the most common is Euclidean

$$W(C_k) = \frac{1}{|C_k|} \sum_{i, i' \in C_k} ||X_i - X_{i'}||_2^2.$$

That is, the average (Euclidean) distance between all cluster members.

K-MEANS

It turns out

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k) \tag{1}$$

is too hard of a problem to solve computationally (K^n partitions!).

So, we make a greedy approximation:

1. Randomly assign observations to the K clusters
2. Iterate until the cluster assignments stop changing:
   - For each of K clusters, compute the centroid, which is the p-length vector of the means in that cluster.
   - Assign each observation to the cluster whose centroid is closest (in Euclidean distance).

This procedure is guaranteed to decrease (1) at each step.

Warning: It finds only a local minimum, not necessarily the global one. Which local min depends on step 1.

K-means: A Summary

To fit K-means, you need to

1. Pick K (inherent in the method)
2. Convince yourself you have found a good solution (due to the randomized approach to the algorithm).

It turns out that 1. is difficult to do in a principled way. We will discuss this shortly

For 2., a commonly used approach is to run K-means many times with different starting points. Pick the solution that has the smallest value for

$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k)$$

As an aside, why can't we use the approach for picking K?

K-means: A Summary

To fit K-means, you need to

1. Pick K (inherent in the method)
2. Convince yourself you have found a good solution (due to the randomized approach to the algorithm).

It turns out that 1. is difficult to do in a principled way. We will discuss this shortly

For 2., a commonly used approach is to run K-means many times with different starting points. Pick the solution that has the smallest value for

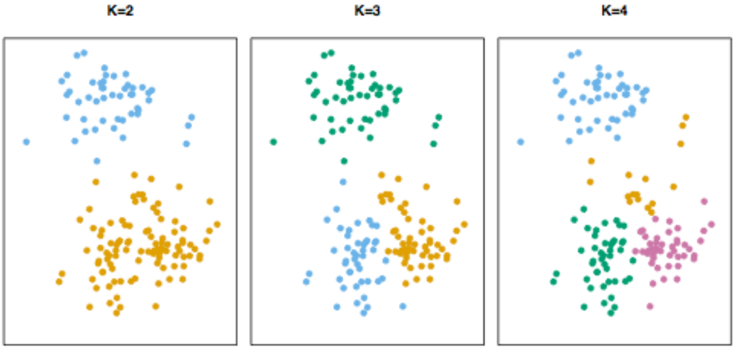
$$\min_{C_1,\ldots,C_K} \sum_{k=1}^{K} W(C_k)$$

As an aside, why can't we use the approach for picking K?

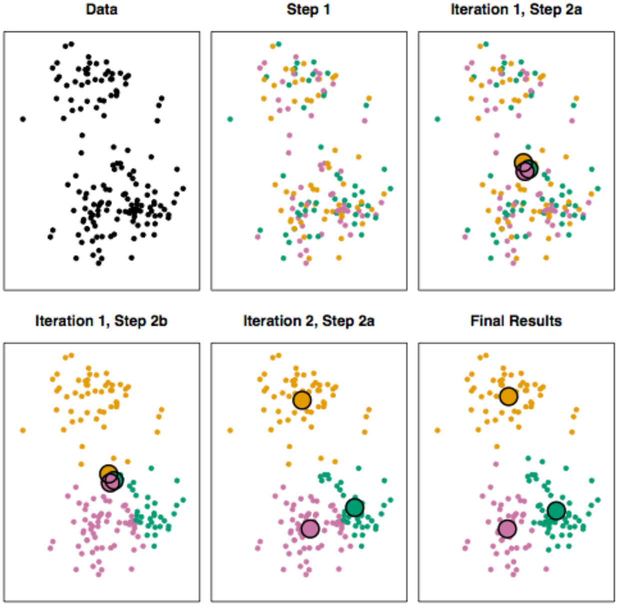
(We would choose $K = n$)

K-means

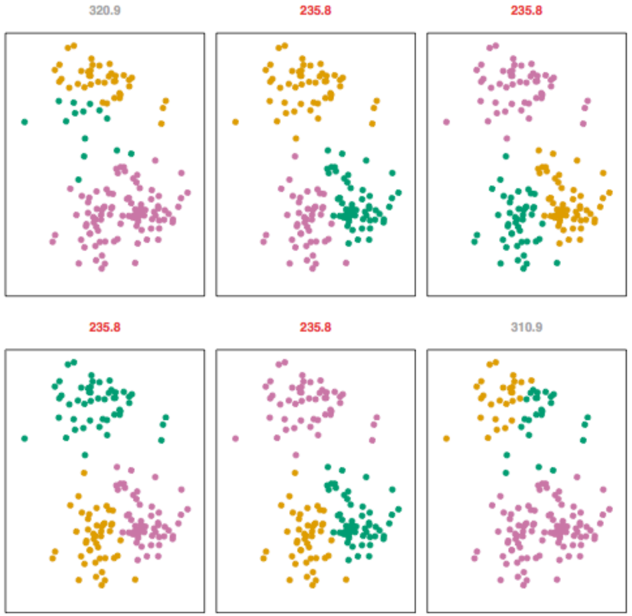
K-means: Various K's



K-means: Algorithm at Work



K-means: Finding Good Local Minimum



K-means in R

Like usual, the interface with R is very basic

```
n   = 30
X1 = rnorm(n)
X2 = rnorm(n)
X   = cbind(X1,X2)
K = 3
kmeans.out = kmeans(X, centers=K)
> names(kmeans.out)
[1] "cluster"      "centers"      "totss"       "withinss"
[5] "tot.withinss" "betweenss"    "size"
> kmeans.out$cluster
 [1] 2 2 2 2 2 2 1 1 2 2 3 1 2 1 2 2 2
 3 1 2 2 1 3 2 1 3 3 1 2 3
```

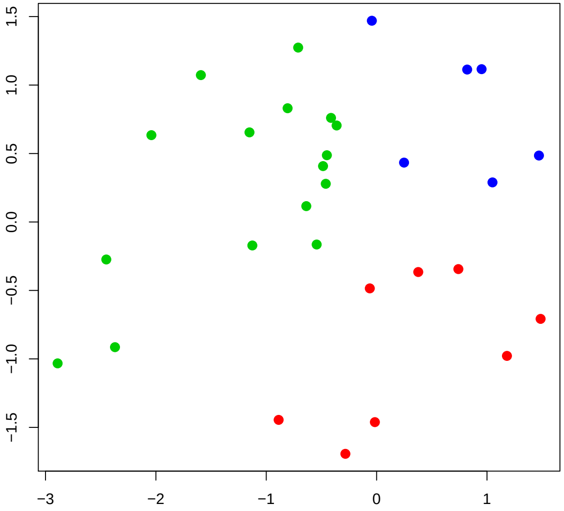
K-means in R

Like usual, the interface with R is very basic

```
n   = 30
X1 = rnorm(n)
X2 = rnorm(n)
X   = cbind(X1,X2)
K = 3
kmeans.out = kmeans(X, centers=K)
> names(kmeans.out)
[1] "cluster"      "centers"      "totss"       "withinss"
[5] "tot.withinss" "betweenss"    "size"
> kmeans.out$cluster
 [1] 2 2 2 2 2 2 1 1 2 2 3 1 2 1 2 2 2
 3 1 2 2 1 3 2 1 3 3 1 2 3

plot(X, col=(kmeans.out$cluster+1), xlab="", ylab="",
           pch=20, cex=2)
```

K-means in R



K-means in R

Another example

```
x = matrix(rnorm(50*2),ncol=2)
x[1:25,1] = x[1:25,1] + 3
x[1:25,2] = x[1:25,2] -4

kmeans.out = kmeans(x,centers=2,nstart=20)
```

K-means in R

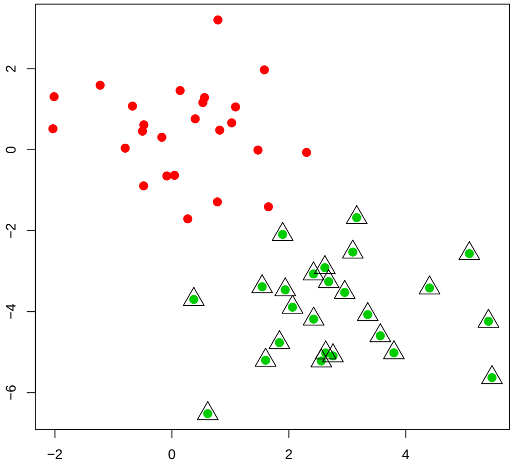


Figure: Two clusters (which is the true number)

K-means in R

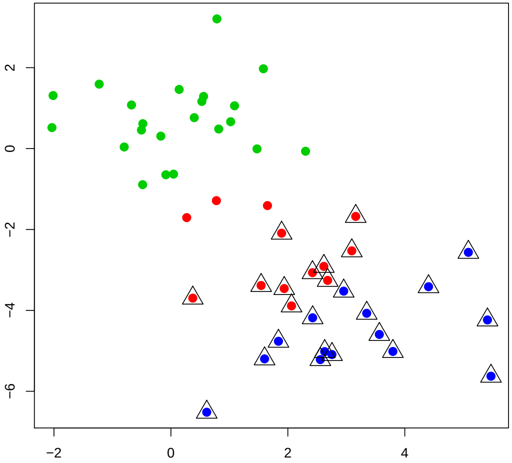


Figure: Three clusters

K-means in R: Comparison

R provides several objects in the kmeans output.

$W(C_k)$ is the same as: kmeans(x,centers=K)\$withinss

$\sum_{k=1}^{K} W(C_k)$ is the same as: kmeans(x,centers=K)\$tot.withinss

```
> kmeans(x,centers=4,nstart=1)$tot.withinss
[1] 19.12722
> kmeans(x,centers=4,nstart=20)$tot.withinss
[1] 18.5526
> kmeans(x,centers=5,nstart=20)$tot.withinss
[1] 12.01893
```

Choosing K

Choosing the number of clusters

Why is it important?

- It might make a big difference (concluding there are $K = 2$ cancer sub-types versus $K = 3$)
- One of the major goals of statistical learning is automatic inference. A good way of choosing K is certainly a part of this

Choosing the number of clusters

Sometimes, the number of clusters is fixed ahead of time:

- Segmenting a client database into K clusters for K salesmen
- Compressing an image using vector quantization (K is the compression rate)

Most of the time, it isn't so straight forward. Why is this a hard problem?

- Determining the number of clusters is hard (for humans) unless the data is low dimensional
- It is just as hard to explain what we are looking for
  (ie: in classification, we want a classifier that predicts well. In clustering, we want a clusterer to ... what?)

Reminder: What does K-means do?

Given a number of clusters K, we (approximately) minimize:

$$\sum_{k=1}^{K} W(C_k) = \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} ||X_i - X_{i'}||_2^2.$$

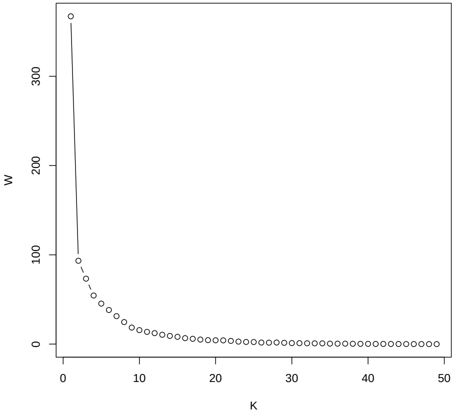
We can rewrite this in terms of the centroids as

$$W(K) = 2 \sum_{k=1}^{K} \sum_{i \in C_k} ||X_i - \overline{X}_k||_2^2,$$

Minimizing W in K

Of course, a lower value of W is better. Why not minimize W?

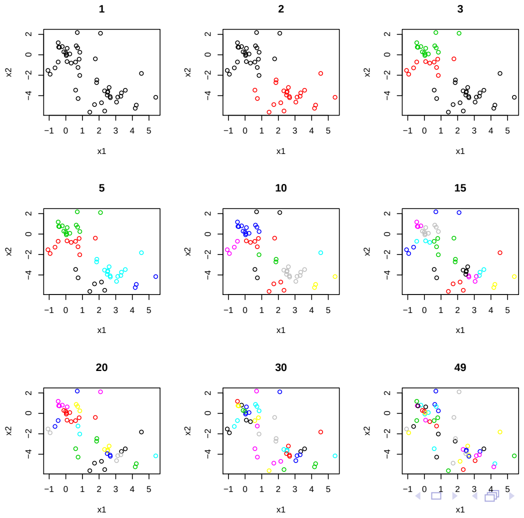
```
plotW = rep(0,49)
for(K in 1:49){
  plotW[K] =  kmeans(x,centers=K,nstart=20)$tot.withinss
}
```



Minimizing W in K

Of course, a lower value of W is better. Why not minimize W?

A look at the cluster solution



Between-cluster variation

Within-cluster variation measures how tightly grouped the clusters are. As we increase K, this will always decrease

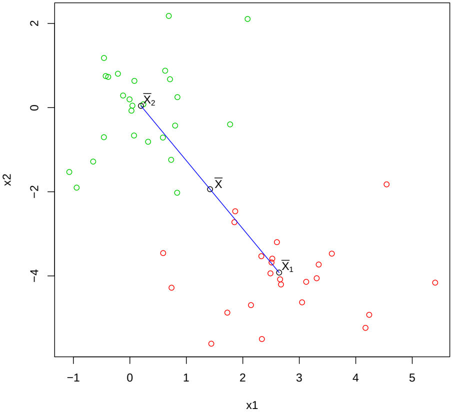
What we are missing is between-cluster variation, ie: how spread apart the groups are

$$B = \sum_{k=1}^{K} |C_k| ||\overline{X}_k - \overline{X}||_2^2,$$

where $|C_k|$ is the number of points in C_k, and $\overline{X}$ is the grand mean of all observations:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i$$

Between and within cluster variation example

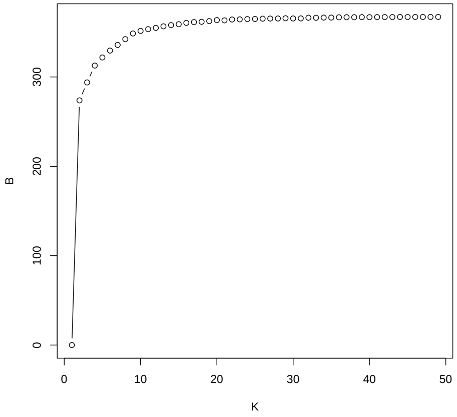


$$B = |C_1| ||\overline{X}_1 - \overline{X}||_2^2 + |C_2| ||\overline{X}_2 - \overline{X}||_2^2$$

$$W = \sum_{i \in C_1} |C_1| ||\overline{X}_1 - X_i||_2^2 + \sum_{i \in C_2} |C_2| ||\overline{X}_2 - X_i||_2^2$$

Can we just maximize B?

Sadly, no. Just like W can be made arbitrarily small, B will always be increasing with increasing K.



CH INDEX

Ideally, we would like our cluster assignment to simultaneously have small W and large B.

This is the idea behind *CH* index. For clustering assignments coming from K clusters, we record CH score:

$$CH(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)}$$

To choose K, pick some maximum number of clusters to be considered ($K_{\max} = 20$, for example) and choose the value of K that

$$\hat{K} = \underset{K \in \{2,\ldots,K_{\max}\}}{\arg\max} \; CH(K).$$

Note: CH is undefined for $K = 1$.

MMA equation (15.37)

CH INDEX

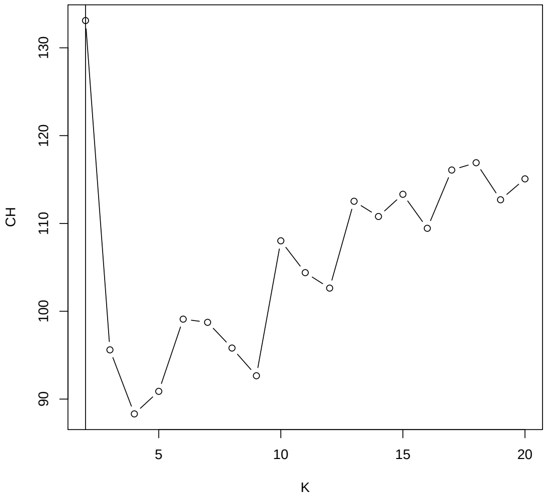
```
ch.index = function(x,kmax,iter.max=100,nstart=10,
                      algorithm="Lloyd")
{
  ch = numeric(length=kmax-1)
  n = nrow(x)
  for (k in 2:kmax) {
    a = kmeans(x,k,iter.max=iter.max,nstart=nstart,
               algorithm=algorithm)
    w = a$tot.withinss
    b = a$betweenss
   ch[k-1] = (b/(k-1))/(w/(n-k))
  }
  return(list(k=2:kmax,ch=ch))
}
```

A simulated example

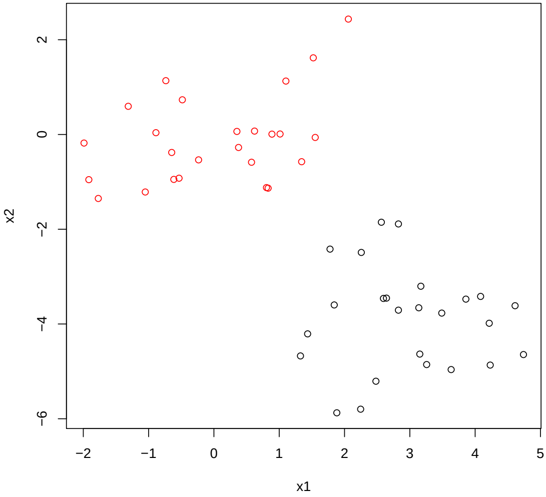
```
x = matrix(rnorm(50*2),ncol=2)
x[1:25,1] = x[1:25,1] + 3
x[1:25,2] = x[1:25,2] -4
```

We want to cluster this data set using K-means with K chosen via CH index.

CH plot



Corresponding solution



Postamble:

- State K-means

  (K-means partitions the feature space, creating clusters)

- Discuss how it can be computed

  (K-means can be computed by randomly setting centroids and iteratively re-assigning points to clusters and recomputing centroids)

- Define a way of choosing the number of clusters

  (The CH index is a way to choose the number of clusters)