

Displays for Statistics 5401

Lecture 42

December 16, 2005
(not given)

Christopher Bingham, Instructor
612-625-1024

Class Web Page

<http://www.stat.umn.edu/~kb/classes/5401>

Copyright© Christopher Bingham 2005

K-means clustering

K-means clustering is useful when you are clustering cases from a N by p data matrix \mathbf{X} and have some idea about the number K of clusters to find.

The formal ideal goal is the following

Find clusters $U_1, U_2, U_3, \dots, U_k$ that minimize $\sum_{1 \leq j \leq k} \text{SSE}_j(U_1, U_2, \dots, U_k)$ where $\text{SSE}_j(U_1, U_2, U_3, \dots, U_k)$ is the error SS in an ANOVA of x_j using the clusters as groups.

Another way to state this is:

Find clusters $U_1, U_2, U_3, \dots, U_k$ that minimize $\text{tr}(\mathbf{E}(U_1, U_2, \dots, U_k))$ where \mathbf{E} is the error matrix from a MANOVA using the clusters as groups

Now, if $\mathbf{H}(U_1, U_2, \dots, U_k)$ is the between groups MANOVA matrix, $\mathbf{E}(U_1, U_2, \dots, U_k) + \mathbf{H}(U_1, U_2, \dots, U_k) = \sum (\mathbf{x}_j - \bar{\mathbf{x}})(\mathbf{x}_j - \bar{\mathbf{x}})'$ doesn't depend on the clustering. This means the ideal goal is equivalent to

Find clusters $U_1, U_2, U_3, \dots, U_k$ so as to maximize $\text{tr}(\mathbf{H}(U_1, U_2, \dots, U_k))$

Such an assignment to clusters is the maximum likelihood assignment assuming clusters correspond to populations with $\text{MVN}(\boldsymbol{\mu}_j, \sigma^2 \mathbf{I}_p)$ distributions, $i = 1, \dots, K$.

This suggests the goal is best adapted to spherical clusters which is in fact the case.

A simpler goal, that would be satisfied by a clustering which minimizes $\text{tr}(\mathbf{E}(U_1, U_2, \dots, U_k))$ is the following:

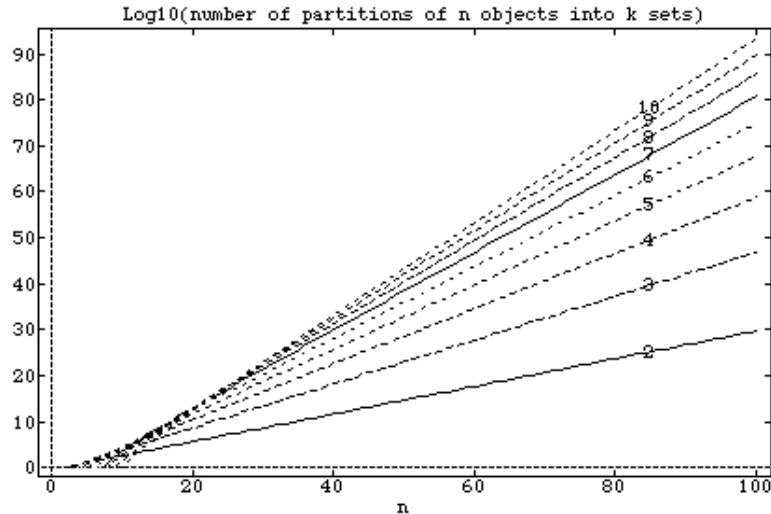
Divide the N data points into k clusters $U_1, U_2, U_3, \dots, U_k$ with means $\bar{\mathbf{x}}_{U_1}, \bar{\mathbf{x}}_{U_2}, \dots, \bar{\mathbf{x}}_{U_k}$ such that

$$U_j = \{\mathbf{x}_\ell \mid \|\mathbf{x}_\ell - \bar{\mathbf{x}}_{U_j}\| = \min_i \|\mathbf{x}_\ell - \bar{\mathbf{x}}_{U_i}\|\}$$

That is, each cluster consists of all the points that are nearest to its centroid. However, a clustering can satisfy this, but not be the clustering that minimizes $\text{tr}(\mathbf{E}(U_1, U_2, \dots, U_k))$.

Even this condition is almost impossible to solve by brute force because there are just too many ways to split into clusters.

When N is even moderately large, to find the "best" of all clusterings, using any criterion, is a tall order since there are approximately $N^k/k!$ such sets.



The ***K-means algorithm*** is an iterative method for, you hope, coming close to the optimal.

In the following, for any set of cases V , \bar{x}_V is the sample mean of the cases in V

- You start with K *initial trial* clusters U_1, \dots, U_k , chosen in some way, possibly randomly, and compute $\bar{x}_{U_j}^{\pm i}$, $j = 1, \dots, k$.

When x_i is not in U_j , $\bar{x}_{U_j}^{+i} \equiv \bar{x}_{\{U_j, i\}}$
When x_i is in U_j , $\bar{x}_{U_j}^{-i} \equiv \bar{x}_{\{U_j - i\}}$

Then, repeat the following until there is no change.

- Examine x_1, x_2, \dots, x_N sequentially.
If $x_i \in U_\ell$, compute the distances $d_j = \|x_i - \bar{x}_{U_j}^{+i}\|$, $j \neq \ell$, $d_\ell = \|x_i - \bar{x}_{U_\ell}^{-i}\|$
- Define J by $d_J = \min\{d_j\}$. If $J \neq \ell$, reallocate x_i to U_J and update means.

Eventually you will complete a cycle through $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ without reallocating any points. Then you stop.

That is, reallocating any point would put it closer to another cluster's centroid than to the centroid of the other points in its cluster.

This differs from the description in Johnson & Wichern. They suggest computing distances

$$d_j = \|\mathbf{x}_i - \bar{\mathbf{x}}_{U_j}\|, \quad j = 1, \dots, K$$

and reallocating \mathbf{x}_i to the nearest group, the group with d_j . Their method performs worse as measured by $\text{tr } \mathbf{E}(U_1, \dots, U_K)$.

It's easy to use the distances to the unadjusted means $\bar{\mathbf{x}}_{U_\ell}$ and $\bar{\mathbf{x}}_{U_j}$ to compute the distances to the adjusted means $\bar{\mathbf{x}}_{U_\ell}^{-i}$ and $\bar{\mathbf{x}}_{U_j}^{+i}$.

When $\mathbf{x}_i \in U_\ell$,

$$\|\mathbf{x}_i - \bar{\mathbf{x}}_{U_\ell}^{-i}\|^2 = (n_\ell / (n_\ell - 1)) \|\mathbf{x}_i - \bar{\mathbf{x}}_{U_\ell}\|^2$$

and for $j \neq \ell$

$$\|\mathbf{x}_i - \bar{\mathbf{x}}_{U_j}^{+i}\|^2 = (n_j / (n_j + 1)) \|\mathbf{x}_i - \bar{\mathbf{x}}_{U_j}\|^2$$

Here $n_j, j = 1, \dots, K$ are the cluster sizes at the point in the algorithm when you examining \mathbf{x}_i .

Example of use of `kmeans()` for doing k-means clustering

Try to cluster the utility company data using K-means. Keywords `kmax` and `kmin` specify that clustering will first be done with $K = 8$, followed by $K = 7, 6, \dots, 3$.

```
Cmd> stuff <- kmeans(data,kmax:8,kmin:3)
Cluster analysis by reallocation of objects using Trace W
criterion
Variables are standardized before clustering
Initial allocation is random
  k   Initial   Final Reallocations
  8   112.13    48.275         13
  8    48.275    45.473          2
  8    45.473    45.21          1
  8    45.21     43.191         2
  8    43.191    43.191          0   Criterion for K = 8
Merging clusters 3 and 7; criterion = 49.35
  k   Initial   Final Reallocations
  7    49.35    48.98          1
  7    48.98    48.98          0   Criterion for K = 7
Merging clusters 1 and 5; criterion = 58.154
  k   Initial   Final Reallocations
  6    58.154    58.154          0   Criterion for K = 6
Merging clusters 2 and 5; criterion = 67.406
  k   Initial   Final Reallocations
  5    67.406    67.406          0   Criterion for K = 5
Merging clusters 2 and 4; criterion = 80.383
  k   Initial   Final Reallocations
  4    80.383    80.383          0   Criterion for K = 4
Merging clusters 1 and 3; criterion = 101.71
  k   Initial   Final Reallocations
  3   101.71    101.71          0   Criterion for K = 3
```

Later clusters start by merging two clusters. As you see, they converge to a solution with less effort than with $K = 8$.

`kmeans()` first uses the k-means algorithm for $K = 8$. Then it merges the two closest clusters using the weighted distances $\{n_i n_j / (n_i + n_j)\} \|\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j\|^2$. This ensures that $\text{tr}(E(U_1, U_2, \dots, U_{k-1}))$ is minimized over the $k(k-1)/2$ clusterings obtainable by merging two clusters.

```
Cmd> compnames(stuff) # the result is a structure
(1) "classes"      N by kmax-kmin+1 matrix
(2) "criterion"    vector of length k
```

```
Cmd> kmeansclass <- stuff$classes
```

```
Cmd> print(kmeansclass,format:"4.0f") # classes
MATRIX:      8      7      6      5      4      3  Clusters
(1,1)        5      5      1      1      1      1
(2,1)        7      6      5      2      2      2
(3,1)         1      1      1      1      1      1
(4,1)         3      3      3      3      3      1
(5,1)         4      4      4      4      2      2
(6,1)         1      1      1      1      1      1
(7,1)         6      6      5      2      2      2
(8,1)         8      7      6      5      4      3
(9,1)         1      1      1      1      1      1
(10,1)        3      3      3      3      3      1
(11,1)         8      7      6      5      4      3
(12,1)         6      6      5      2      2      2
(13,1)         3      3      3      3      3      1
(14,1)         5      5      1      1      1      1
(15,1)         6      6      5      2      2      2
(16,1)         8      7      6      5      4      3
(17,1)         2      2      2      2      2      2
(18,1)         5      5      1      1      1      1
(19,1)         5      5      1      1      1      1
(20,1)         3      3      3      3      3      1
(21,1)         6      6      5      2      2      2
(22,1)         7      3      3      3      3      1
```

Note K decreases from left to right.

```

Cmd> split(run(22), kmeansclass[,5]) # 4 cluster membership
component: comp1  Cases in cluster 1
(1) 1 3 6 9 14
(6) 18 19
component: comp2  Cases in cluster 2
(1) 2 5 7 12 15
(6) 17 21
component: comp3  Cases in cluster 3
(1) 4 10 13 20 22
component: comp4  Cases in cluster 4
(1) 8 11 16

Cmd> split(run(22), avelnkclass[,3]) # compare w/ aver linkage
component: comp1  Cases in cluster 1
(1) 1 3 4 6 9
(6) 10 13 14 18 19
(11) 20 22
component: comp2  Cases in cluster 2
(1) 8 11 16
component: comp3  Case in cluster 3
(1) 5
component: comp4  Cases in cluster 4
(1) 2 7 12 15 17
(6) 21

Cmd> tabs(,avelnkclass[,3],kmeansclass[,5]) # confusion matrix
(1,1) 0 7 5 0 Ave Link 1
(2,1) 0 0 0 3 Ave Link 2
(3,1) 1 0 0 0 Ave Link 3
(4,1) 6 0 0 0 Ave Link 4

Cmd> tabs(,avelnkclass[,4],kmeansclass[,4]) # same, K = 5
(1,1) 7 0 5 0 0
(2,1) 0 0 0 0 3
(3,1) 0 0 0 1 0
(4,1) 0 5 0 0 0
(5,1) 0 1 0 0 0

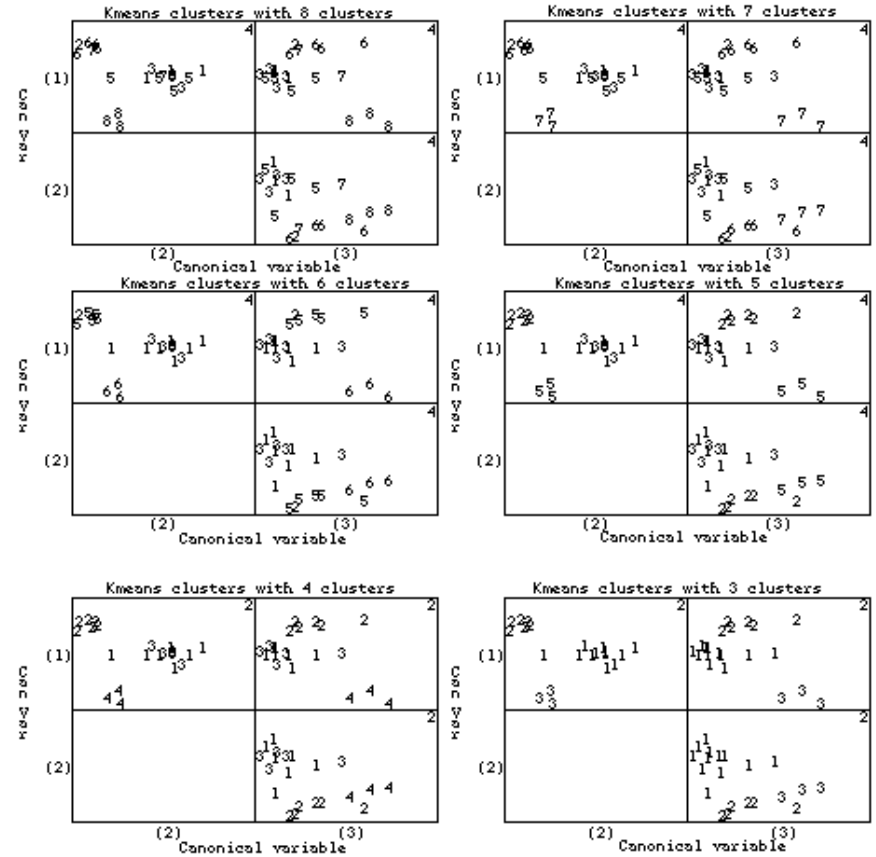
Cmd> junk <- kmeans(data,avelnkclass[,3],start:"class")
Cluster analysis by reallocation of objects using Trace W
criterion
Variables are standardized before clustering
Initial allocation is predefined
k Initial Final Reallocations
4 88.734 88.734 0
    
```

No reallocations, but $tr E > K$ -means $tr E$

Plots of clusters using canonical variables

```

Cmd> for(i,run(6)){plotmatrix(z,symbols:kmeansclass[,i],\
upper:T,title:paste("Kmeans clusters with",9-i,"clusters"),\
xlab:"Canonical variable",ylab:"Can
Var",xaxis:F,yaxis:F,wind:i)}
    
```



By default `kmeans()` uses random starting clusters.

I did 200 `kmeans()` clustering with $K = 4$, each using a different random start.

```
Cmd> M <- 200; CRITERION <- rep(0,M)

Cmd> for(i,run(M)){ # cluster and save criterion
  CRITERION[i] <- kmeans(data,kmax:4)$criterion;;}

Cmd> unique(round(CRITERION,3)) # 8 different criterion found
(1)      80.383      92.01      90.883      91.781      96.11
(6)      88.734      99.207      92.52 Different values found

Cmd> sum(round(CRITERION,3) == unique(round(CRITERION,3)))
(1,1)      184      3      3      5      2
(1,6)      1      1      1 Counts of each value
```

Most of the time, it found the identical clustering. 184 times it hit 80.383, and never was greater than 92.52.

J&W don't say how they did the clustering in their example, but it is not optimal. `kmeans()` can improve on it

```
Cmd> junk <- kmeans(data,jwclass,start:"class",kmax:4)
Cluster analysis by reallocation of objects using Trace W
criterion
Variables are standardized before clustering
Initial allocation is predefined
  k   Initial   Final Reallocations
  4    85.84    80.383         4
  4    80.383    80.383         0
```

The starting value for the criterion is 85.84, worse than what `kmeans()` accomplished.

I wrote a macro to do J&W type K-means clustering. It does worse than `kmeans()`.

```
Cmd> dataS <- standardize(data) # macro does not standardize

Cmd> CRITERION1 <- rep(0, M)

Cmd> for(i,run(M)){
  CRITERION1[i] <- \
    reverse(kmeansmac1(dataS,k:4,silent:T)$criterion)[1];;}

Cmd> describe(hconcat(CRITERION,CRITERION1), \
  mean:T, stddev:T, min:T, max:T, median:T)
component: min      Minima from kmeans() and and J&W kmeans
(1)      80.383      80.383
component: median Medians from kmeans() and and J&W kmeans
(1)      80.383      93.752
component: max      Max from kmeans() and and J&W kmeans
(1)      99.207      123.7
component: mean     Maxima from kmeans() and and J&W kmeans
(1)      81.354      93.566
component: stddev
(1)      3.3726      9.3496

Cmd> min(abs(CRITERION1-85.84)) # never hit 85.84
(1)      0.375
```

Not once did I get 85.84 and the mean and median are worse than from `kmeans()`. Now do K-means clustering of can. vars.

```
Cmd> zclasses <- kmeans(z,kmax:4)$classes
Cluster analysis by reallocation of objects using Trace W
criterion
Variables are standardized before clustering
Initial allocation is random
  k   Initial   Final Reallocations
  4    57.628    8.8961        15
  4     8.8961    8.8961         0

Cmd> @junk <- kmeans(data,zclasses,start:"class")
  k   Initial   Final Reallocations
  4    88.734    88.734         0
```

Example with artificial data set with 4 known "clusters".

```

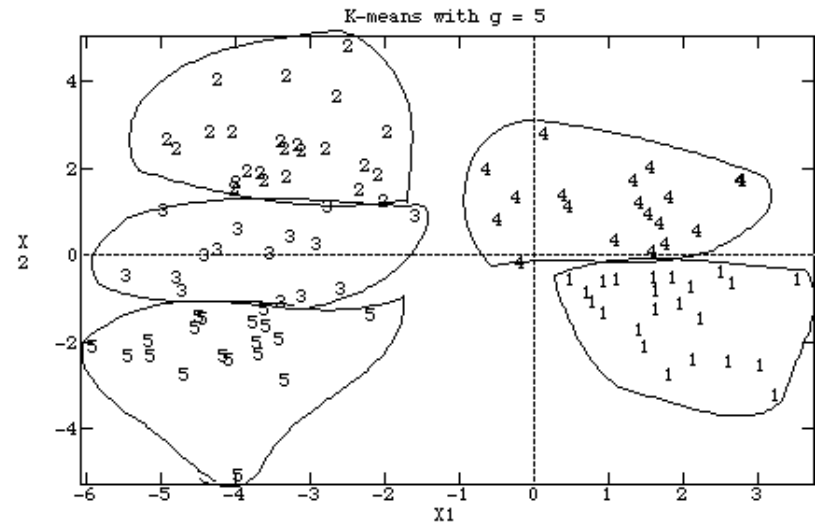
Cmd> results <- kmeans(x1,kmax:5,kmin:3) # K-means analysis
Cluster analysis by reallocation of objects using Trace W
criterion
Variables are standardized before clustering
Initial allocation is random
k      Initial      Final Reallocations
5      190.45        45.582      77
5      45.582        35.884      19
5      35.884        28.866      14
5      28.866        28.386       5
5      28.386        28.386       0
Merging clusters 3 and 5; criterion = 38.985
k      Initial      Final Reallocations
4      38.985        35.321       6
4      35.321        35.252       1
4      35.252        35.252       0
Merging clusters 1 and 4; criterion = 53.093
k      Initial      Final Reallocations
3      53.093        50.767       2
3      50.767        50.748       1
3      50.748        50.748       0
    
```

This first found a 5 group clustering (kmax:5), taking four cycles through the cases before no more points to move on cycle 5. Then in merged clusters 3 and 5, the pair with the smallest value of

$$(n_i n_j / (n_i + n_j)) \| \bar{y}_i - \bar{y}_j \|^2$$

This minimizes the tr **E** criterion after merging.

Here is the 5 cluster solution found



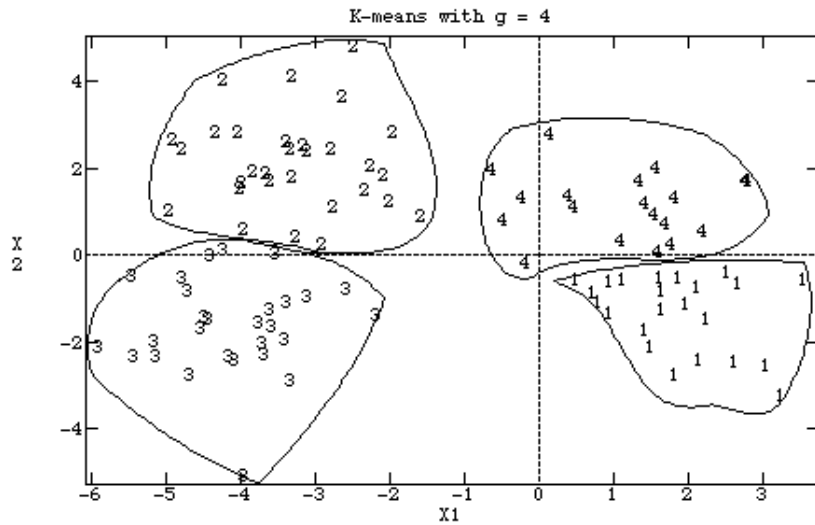
Here is the confusion matrix with the "true" clusters:

```

Cmd> tabs(,groups,results$classes[,1])
(1,1)      3      0      0      17      0 Group 1
(2,1)     20      0      0      2      0 Group 2
(3,1)      0     24      6      0      0 Group 3
(4,1)      0      0      9      0     19 Group 4
    
```

Groups 1 and 2 are almost entirely in k-means clusters 4 and 1, respectively; group 3 is 75% in k-means cluster 2 and group 4 is 68% in k-means cluster 5, with kmeans cluster 3 overlapping both groups 3 and 4.

The 4 cluster solution:

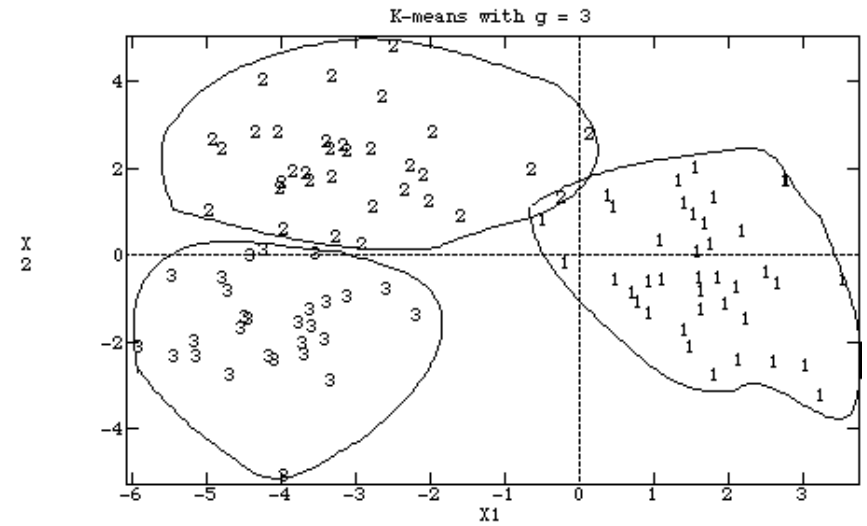


with confusion matrix

```
Cmd> tabs(,groups,results$classes[,2])
(1,1)    3    0    0    17 Group 1
(2,1)   20    0    0    2  Group 2
(3,1)    0   29    1    0 Group 3
(4,1)    0    1   27    0 Group 4
```

This does a remarkably good job each cluster almost coinciding with a sample.

The three cluster solution merges the two clusters on the right.



Confusion matrix

```
Cmd> tabs(,groups,results$classes[,3])
(1,1)   17    3    0 Group 1
(2,1)   22    0    0 Group 2
(3,1)    0   29    1 Group 3
(4,1)    0    1   27 Group 4
```

By carefully comparing the solutions, you can verify that this process is not hierarchical. For example, although the new cluster 1 is primarily a merging of clusters 4 and 1, some of cluster 4 ended up in cluster 2.