Clustering - V

Prof. Dan A. Simovici

UMB

Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of vectors in $\mathbb{R}^m$ and let $\phi : \mathbb{R}^m \longrightarrow \mathbb{R}^p$ be a function, where $\mathbb{R}^p$ is a high-dimensional space referred to as the *feature space*.

Since $\mathbb{R}^p$ may have a much higher dimension than m, the computation of the inner product $\phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$ can be costly. Therefore, it is interesting to examine transformations of the form $\phi : \mathbb{R}^m \longrightarrow \mathbb{R}^p$ for which there exists a symmetric function $K : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow \mathbb{R}$ such that

$$\phi(\mathbf{x}_i)'\phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$$

for $\mathbf{x}_i, \mathbf{x}_j \in X$. Thus, a kernel function facilitates the computation of the inner product $\phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$ in the high-dimensional space.

Recall:

Definition

A function $K : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow \mathbb{R}$ is a ϕ-*kernel* if $\phi(\mathbf{x}_i)'\phi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$ for all $\mathbf{x}_i, \mathbf{x}_j \in X$.
A function $K : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow \mathbb{R}$ is a *kernel* if there exists a function $\phi : \mathbb{R}^m \longrightarrow \mathbb{R}^p$ such that K is a ϕ-kernel.

Example

Let $K : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow \mathbb{R}$ be the function defined as

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}'\mathbf{y} + a)^d$$

for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$. We can write

$$\begin{aligned} & K(\mathbf{x}, \mathbf{y}) \\ &= (\mathbf{x}'\mathbf{y} + a)^d = (x_1 y_1 + \cdots + x_m y_m + \sqrt{a}\sqrt{a})^d \\ &= \sum_{n_1 + \cdots + n_{m+1} = d} \binom{d}{n_1 \cdots n_m n_{m+1}} x_1^{n_1} y_1^{n_1} \cdots x_m^{n_1} y_m^{n_1} \cdots a^{\frac{n_m+1}{2}} a^{\frac{n_m+1}{2}} \\ &= (\phi(\mathbf{x}), \phi(\mathbf{y})), \end{aligned}$$

where

$$\phi(\mathbf{x}) = \left(\ldots, \sqrt{\binom{d}{n_1 \cdots n_m n_{m+1}}} x_1^{n_1} \cdots x_m^{n_m} a^{\frac{n_m+1}{2}}, \ldots \right).$$

Example cont'd

For each monomial that is a component of $\phi(x)$ we have $\sum_{j=1}^{m+1} n_j = d$ and $n_j \geqslant 0$ for $1 \leqslant j \leqslant m+1$.
When $m = 2$ and $d = 2$ we have

$$K(\mathbf{x}, \mathbf{y}) = (x_1 y_1 + x_2 y_2 + a)^2 = x_1^2 y_1^2 + a^2 + 2x_1 y_1 x_2 y_2 + 2a x_1 y_1 + 2a x_2 y_2$$

$=$

where $\phi : \mathbb{R}^2 \longrightarrow \mathbb{R}^6$ is defined as

$$\phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2} x_1 x_2 \\ \sqrt{2a} x_1 \\ \sqrt{2a} x_2 \\ a \end{pmatrix}$$

for $\mathbf{x} \in \mathbb{R}^2$.

Example

Let $K : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow \mathbb{R}$ be the *radial basis function* defined as:

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}}$$

for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$.
The linear space $\ell^2(\mathbb{R})$ is an infinite-dimensional linear space that consists of infinite sequences $\mathbf{s} = (s_0, s_1, \ldots) \in \mathbf{Seq}(\mathbb{R})$ such that $\sum_{n\in\mathbb{N}} s_n^2$ is finite.
We shall prove that K corresponds to a feature function $\phi : \mathbb{R}^m \longrightarrow \ell^2(\mathbb{R})$.

Example cont'd

We have:

$$K(\mathbf{x},\mathbf{y}) = e^{-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2\sigma^2}} = e^{-\frac{\|\mathbf{x}\|^2+\|\mathbf{y}\|^2-2(\mathbf{x},\mathbf{y})}{2\sigma^2}}$$
$$= e^{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}} \cdot e^{-\frac{\|\mathbf{y}\|^2}{2\sigma^2}} \cdot e^{\frac{(\mathbf{x},\mathbf{y})}{\sigma^2}}$$

Taking into account that $e^{\frac{(\mathbf{x},\mathbf{y})}{\sigma^2}} = \sum_{j=0}^{\infty} \frac{1}{j!} \frac{(\mathbf{x},\mathbf{y})^j}{\sigma^{2j}}$ we can write:

$$e^{\frac{(\mathbf{x},\mathbf{y})}{\sigma^2}} \cdot e^{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}} \cdot e^{-\frac{\|\mathbf{y}\|^2}{2\sigma^2}} = \sum_{j=0}^{\infty} \frac{(\mathbf{x},\mathbf{y})^j}{j!\sigma^{2j}} e^{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}} \cdot e^{-\frac{\|\mathbf{y}\|^2}{2\sigma^2}}$$
$$= \sum_{j=0}^{\infty} \left(\frac{e^{-\frac{\|\mathbf{x}\|^2}{2j\sigma^2}}}{\sigma\sqrt{j!}^{\frac{1}{j}}} \frac{e^{-\frac{\|\mathbf{y}\|^2}{2j\sigma^2}}}{\sigma\sqrt{j!}^{\frac{1}{j}}} (\mathbf{x},\mathbf{y}) \right)^j = (\phi(\mathbf{x}),\phi(\mathbf{y})).$$

The function ϕ is given by:

$$\phi(\mathbf{x}) = \left(\ldots, \frac{e^{-\frac{\|\mathbf{x}\|^2}{2j\sigma^2}}}{\sigma^j \sqrt{j!}^{\frac{1}{j}}} \binom{j}{n_1, \ldots, n_k}^{\frac{1}{2}} x_1^{n_1} \cdots x_k^{n_k}, \ldots \right).$$

In this formula j varies in $\mathbb{N}$ and $n_1 + \cdots + n_k = j$.

Theorem

Let function $\phi : \mathbb{R}^m \longrightarrow \mathbb{R}^p$ *be a function,* c^* *be the centroid of the set* $\phi(X) \subseteq \mathbb{R}^p$*, and let* $sse^*(X)$ *be the sum of square errors of* $\phi(X)$ *in* $\mathbb{R}^p$*. We have:*

$$sse^*(X) = \sum_{j=1}^{n} K(\mathrm{x}_j, \mathrm{x}_j) - nK(\mathrm{c}^*, \mathrm{c}^*).$$

Furthermore, if $\sigma = \{C_1, \ldots, C_m\}$ *is a partition of* X *and* c_k^* *is the centroid of* $\phi(C_k)$ *for* $1 \leqslant k \leqslant m$*, define the sum of the squared errors of* σ *in* $\mathbb{R}^p$ *as*

$$sse^*(\sigma) = \sum_{\ell=1}^{m} sse^*(C_\ell).$$

Then, we have:

$$sse^*(\sigma) = \sum_{j=1}^{n} K(\mathrm{x}_j, \mathrm{x}_j) - \sum_{\ell=1}^{m} |C_\ell| K(\mathrm{c}_\ell^*, \mathrm{c}_\ell^*).$$

Proof

The definition of sse^* implies;

$$\text{sse}^*(X) = \sum_{j=1}^{n} \parallel \phi(\mathbf{x}_j) - \mathbf{c}^* \parallel^2,$$

where $\mathbf{c}^* = \frac{1}{n}\sum_{j=1}^{n} \phi(\mathbf{x}_j)$. Therefore,

$$\begin{aligned}
\text{sse}^*(X) &= \sum_{j=1}^{n} (\phi(\mathbf{x}_j) - \mathbf{c}^*)'(\phi(\mathbf{x}_j) - \mathbf{c}^*) \\
&= \sum_{j=1}^{n} \phi(\mathbf{x}_j)'\phi(\mathbf{x}_j) - 2\sum_{j=1}^{n} \phi(\mathbf{x}_j)'\mathbf{c}^* + n(\mathbf{c}^*)'\mathbf{c}^* \\
&= \sum_{j=1}^{n} K(\mathbf{x}_j, \mathbf{x}_j) - nK(\mathbf{c}^*, \mathbf{c}^*)
\end{aligned}$$

Proof cont'd

For the second equality note that

$$\text{sse}^*(C_\ell) = \sum_{\mathbf{x}_j \in C_\ell} K(\mathbf{x}_j, \mathbf{x}_j) - |C_\ell| K(\mathbf{c}_\ell^*, \mathbf{c}_\ell^*).$$

This allows us to write

$$\begin{aligned}
&\sum_{\ell=1}^{m} \text{sse}^*(C_\ell) \\
&\quad = \sum_{\ell=1}^{m} \left(\sum_{\mathbf{x}_j \in C_\ell} K(\mathbf{x}_j, \mathbf{x}_j) - |C_\ell| K(\mathbf{c}_\ell^*, \mathbf{c}_\ell^*) \right) \\
&\quad = \sum_{j=1}^{n} K(\mathbf{x}_j, \mathbf{x}_j) - \sum_{\ell=1}^{m} |C_\ell| K(\mathbf{c}_\ell^*, \mathbf{c}_\ell^*)
\end{aligned}$$

Theorem

The distance in the feature space between $\phi(x_j)$ and the centroid c_i^* of the set C_i can be expressed using the kernel K.

Proof

We have:

$$
\begin{aligned}
\| \phi(\mathbf{x}_j) - \mathbf{c}_i^* \|^2 &= \| \phi(\mathbf{x}_j) \|^2 - 2\phi(\mathbf{x}_j)'\mathbf{c}_i^* + \| \mathbf{c}_i^* \|^2 \\
&= K(\mathbf{x}_j, \mathbf{x}_j) - \frac{2}{|C_i|} \sum_{\mathbf{x}_p \in C_i} \phi(\mathbf{x}_j)'\phi(\mathbf{x}_p) \\
&\quad + \frac{1}{|C_i|^2} \sum_{\mathbf{x}_p \in C_i} \sum_{\mathbf{x}_q \in C_i} K(\mathbf{x}_p, \mathbf{x}_q) \\
&= K(\mathbf{x}_j, \mathbf{x}_j) - \frac{2}{|C_i|} \sum_{\mathbf{x}_p \in C_i} K(\mathbf{x}_j, \mathbf{x}_p) \\
&\quad + \frac{1}{|C_i|^2} \sum_{\mathbf{x}_p \in C_i} \sum_{\mathbf{x}_q \in C_i} K(\mathbf{x}_p, \mathbf{x}_q).
\end{aligned}
$$

The theorem shows that the distance from the image of a point to the centroid of the cluster in the feature space can be computed via the kernel. The point $\mathbf{x}_j$ is assigned to the cluster C_i, where

$$
\begin{aligned}
i &= \text{argmin}_i \parallel \phi(\mathbf{x}_j) - \mathbf{c}_i^* \parallel^2 \\
&= \text{argmin}_i \left(K(\mathbf{x}_j, \mathbf{x}_j) - \frac{2}{|C_i|} \sum_{\mathbf{x}_p \in C_i} K(\mathbf{x}_j, \mathbf{x}_p) + \frac{1}{|C_i|^2} \sum_{\mathbf{x}_p \in C_i} \sum_{\mathbf{x}_q \in C_i} K(\mathbf{x}_p, \mathbf{x}_q) \right) \\
&= \text{argmin}_i \left(\frac{1}{|C_i|^2} \sum_{\mathbf{x}_p \in C_i} \sum_{\mathbf{x}_q \in C_i} + K(\mathbf{x}_p, \mathbf{x}_q) - \frac{2}{|C_i|} \sum_{\mathbf{x}_p \in C_i} K(\mathbf{x}_j, \mathbf{x}_p) \right),
\end{aligned}
$$

because the term $K(\mathbf{x}_j, \mathbf{x}_j)$ is the same for every i. The kernelized version of the k-means algorithm proceeds along the same line as the standard k-means except that the assignment on points to centroids is done in the feature space rather than the original space.

The `kernlab` package contains the function `kkmeans` that implements kernel k-means using a variety of possible kernels: the radial base kernel, `rbfdot`, the polynomial kernel `polydot`, and many others. After loading this package using

```
> library(kernlab)
```

we can build a clustering of the `iris` database excluding the attribute `Species` using

```
> c1 <- kkmeans(as.matrix(iris[,-5]),centers=3)
```

This call to function `kkmeans` is using the `rbfdot` kernel automatic estimation of σ and results in the clustering `c1` described below:

```
> c1
Spectral Clustering object of class "specc"

 Cluster memberships:

2 3 2 3 2 2 3 2 3 2 2 2 3 3 2 2 2 2 2 2 2 2 3 2  ...

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  1.15460641776431

Centers:
         [,1]     [,2]     [,3]      [,4]
[1,] 6.262000 2.872000 4.906000 1.6760000
[2,] 5.151351 3.556757 1.502703 0.2594595
[3,] 4.592308 3.061538 1.346154 0.2076923

Cluster size:
[1] 100  37  13
```

A better result can be obtained with the `polydot` kernel:

```
> c2 <- kkmeans(as.matrix(iris[,-5]),kernel="polydot",
        kpar=list(degree=2),centers=3)
```

The resulting clustering is

```
> c2
Spectral Clustering object of class "specc"

 Cluster memberships:

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 3 3 ...

Polynomial kernel function.
 Hyperparameters : degree =  2  scale =  1  offset =  1

Centers:
         [,1]     [,2]     [,3]     [,4]
[1,] 6.853846 3.076923 5.715385 2.053846
[2,] 5.991667 2.795833 4.562500 1.512500
[3,] 5.104762 3.244444 1.933333 0.431746
```

```
Cluster size:
[1] 39 48 63

Within-cluster sum of squares:
[1] 1182.046 1121.498 1566.437
```

Note that the sizes of the clusters are closer to the sizes of the reference clusters; also the values of the within-clusters sums of squares are more balanced.
Similar results can be obtained with the `polydot` kernel of degree 3.

Another algorithm, named PAM (an acronym of "Partition Around Medoids") developed by Kaufman and Rousseeuw, also requires as an input parameter the number k of clusters to be extracted.
The k clusters are determined based on a representative object from each cluster, called the *medoid* of the cluster. The medoid of a cluster is one of the objects that have a most central position in the cluster.

Objects that are tentatively defined as medoids are placed into a set S of *selected objects*. If O is the set of objects then the set $U = O - S$ is the set of *unselected objects*.
The goal of the algorithm is to minimize the average dissimilarity of objects to their closest selected object. Equivalently, we can minimize the sum of the dissimilarities between object and their closest selected object.

The algorithm has two phases:

i. In the first phase, BUILD, a collection of k objects are selected for an initial set S.
ii. In the second phase, SWAP, one tries to improve the quality of the clustering by exchanging selected objects with unselected objects.

For each object p we maintain two numbers:

- D_p, the dissimilarity between p and the closest object in S, and
- E_p, the dissimilarity between p and the second closest object in S.

These numbers *must be updated every time when the sets S and U change*. Note that $D_j \leqslant E_j$ and that we have $p \in S$ if and only if $D_p = 0$.

PAM begins with a set of objects O, where $|O| = n$, a dissimilarity $n \times n$ matrix D, and a prescribed number of clusters k. The d_{ij} entry of the matrix D is the dissimilarity $d(o_i, o_j)$ between the objects o_i and o_j.

The algorithm has two phases:

- the *building phase*, and
- the *swapping phase*.

The BUILD phase

1. Initialize S by adding to it an object for which the sum of the distances to all other objects is minimal.
2. Consider an object $i \in U$ as a candidate for inclusion into the set of selected objects.
3. For an object $j \in U$ compute D_j, the dissimilarity between j and the closest object in S.
4. If $D_j > d(i,j)$ object j will contribute to the decision to select object i (because the quality of the clustering may benefit); let $C_{ji} = \max\{D_j - d(j,i), 0\}$.
5. Compute the total gain obtained by adding i to S as $g_i = \sum_{j \in U} C_{ji}$.
6. Choose that object i that maximizes g_i; let $S := S \cup \{i\}$ and $U = U - \{i\}$.

These steps are performed until k objects have been selected.

The second phase, SWAP, attempts to improve the the set of selected objects and, therefore, to improve the quality of the clustering.
This is done by considering all pairs $(i, h) \in S \times U$ and consists of computing the effect T_{ih} on the sum of dissimilarities between objects and the closest selected object caused by swapping i and h, that is, by transferring i from S to U and transferring h to from U to S.
The computation of T_{ih} involves the computation of the contribution K_{jih} of each object $j \in U$ to the swap of i and h.

K_{jih} is computed taking into account the following cases:

(a) if $d(j,i) > D_j$ and $d(j,h) > D_j$ (which means that there is $\ell \in S$ such that $d(j,h) > d(j,\ell)$) then $K_{jih} = 0$;

(b) if $d(j,i) = D_j$, then two cases occur:

(b)-(i) if $d(j,h) < E_j$, where E_j is the dissimilarity between j and the second closest selected object, then $K_{jih} = d(j,h) - d(j,i)$; note that K_{jih} can be either positive or negative.

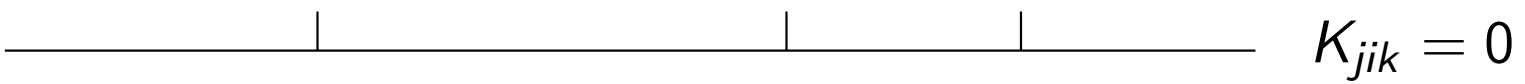
(b)-(ii) if $d(j,h) \geq E_j$, then $K_{jih} = E_j - D_j$; in this case $K_{jih} > 0$.

(c) if $d(j,i) > D_j$ and $d(j,h) < D_j$, then

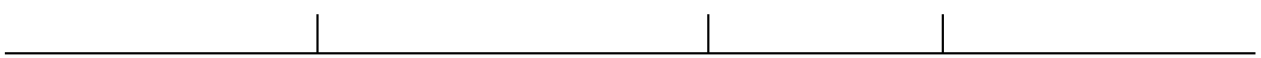
$$K_{jih} = d(j,h) - D_j.$$

Computation of K_{jih}

Cases (a) and (b)-(i)



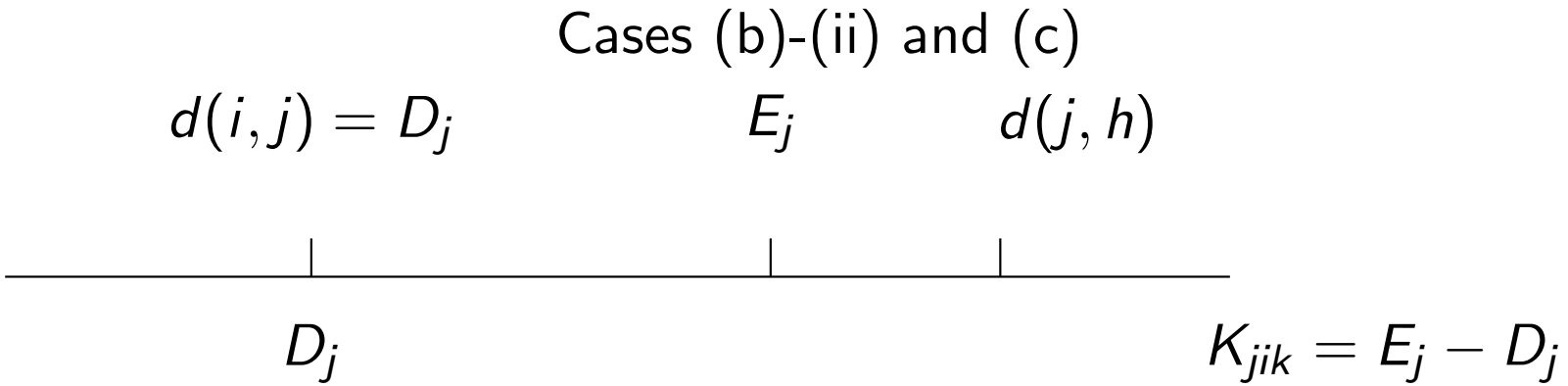
(a)



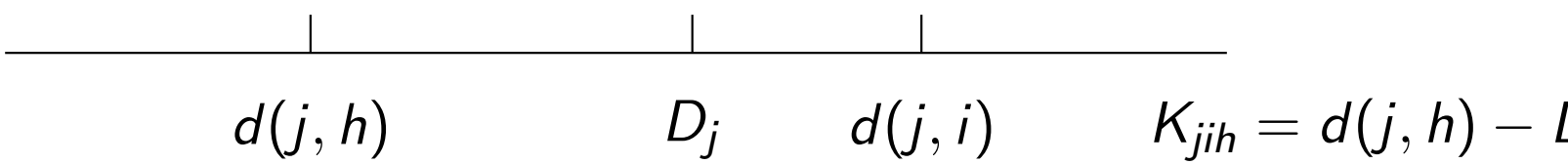
(b)-(i)

Computation of K_{jih}

Cases (b)-(ii) and (c)



(b)-(ii)



(c)

- Compute the total result of the swap as

$$T_{ih} = \sum \{K_{jih} \mid j \in U\}.$$

- Select a pair $(i, h) \in S \times U$ that minimizes T_{ih}.
- If $T_{ih} < 0$ the swap is carried out, *D_p and E_p are updated* for every object p, and we return at Step 1. If min $T_{ih} > 0$, the value of the objective cannot be decreased and the algorithm halts. Of course, this happends when all values of T_{ih} are positive and this is precisely the halting condition of the algorithm.

Specialized Functions

Functions that allow visualization of a data frame:

`dim()`	shows the dimensions of the data frame by row and column
`str()`	shows the structure of the data frame
`summary()`	provides summary statistics on the columns
`colnames()`	shows the name of each column in the data frame
`head()`	shows the first 6 rows of the data frame
`tail()`	shows the last 6 rows of the data frame
`View()`	shows a spreadsheet-like display of the entire data frame

Example

We demonstrate the use of these functions on the data frame `USArrests`:

```
> d <- USArrests
> dim(d)
[1] 50  4
> str(d)
'data.frame':   50 obs. of  4 variables:
 $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
 $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
 $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
 $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

Example

```
> summary(d)
     Murder          Assault         UrbanPop          Rape
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00
> colnames(d)
[1] "Murder"   "Assault"  "UrbanPop" "Rape"
```

Example

```
> head(d)
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
```

Example

```
> tail(d)
               Murder Assault UrbanPop Rape
Vermont           2.2      48       32 11.2
Virginia          8.5     156       63 20.7
Washington        4.0     145       73 26.2
West Virginia     5.7      81       39  9.3
Wisconsin         2.6      53       66 10.8
Wyoming           6.8     161       60 15.6
```

We need to install the following packages:

```
cluster
factoextra
ggplot2
ggsignif
```

A clustering of states that consists of two clusters can be obtained writing

```
pam.res <- pam(d,2)
```

where *d* is the data frame that contains data from txUSArrests.

A call to `print(pam.res)` results in

```
Medoids:
         ID Murder Assault UrbanPop Rape
Michigan 22   12.1     255       74 35.1
Kansas   16    6.0     115       66 18.0
Clustering vector:
       Alabama         Alaska        Arizona       Arkansas      Cali
             1              1              1              1
      Colorado    Connecticut       Delaware        Florida         C
             1              2              1              1
        Hawaii          Idaho       Illinois        Indiana
             2              2              1              2
        Kansas       Kentucky      Louisiana          Maine        Ma
             2              2              1              2
 Massachusetts       Michigan      Minnesota    Mississippi        Mi
             2              1              2              1
       Montana       Nebraska         Nevada  New Hampshire     New
             2              2              1              2
    New Mexico       New York North Carolina   North Dakota
             1              1              1              2
      Oklahoma         Oregon   Pennsylvania   Rhode Island South Ca
```

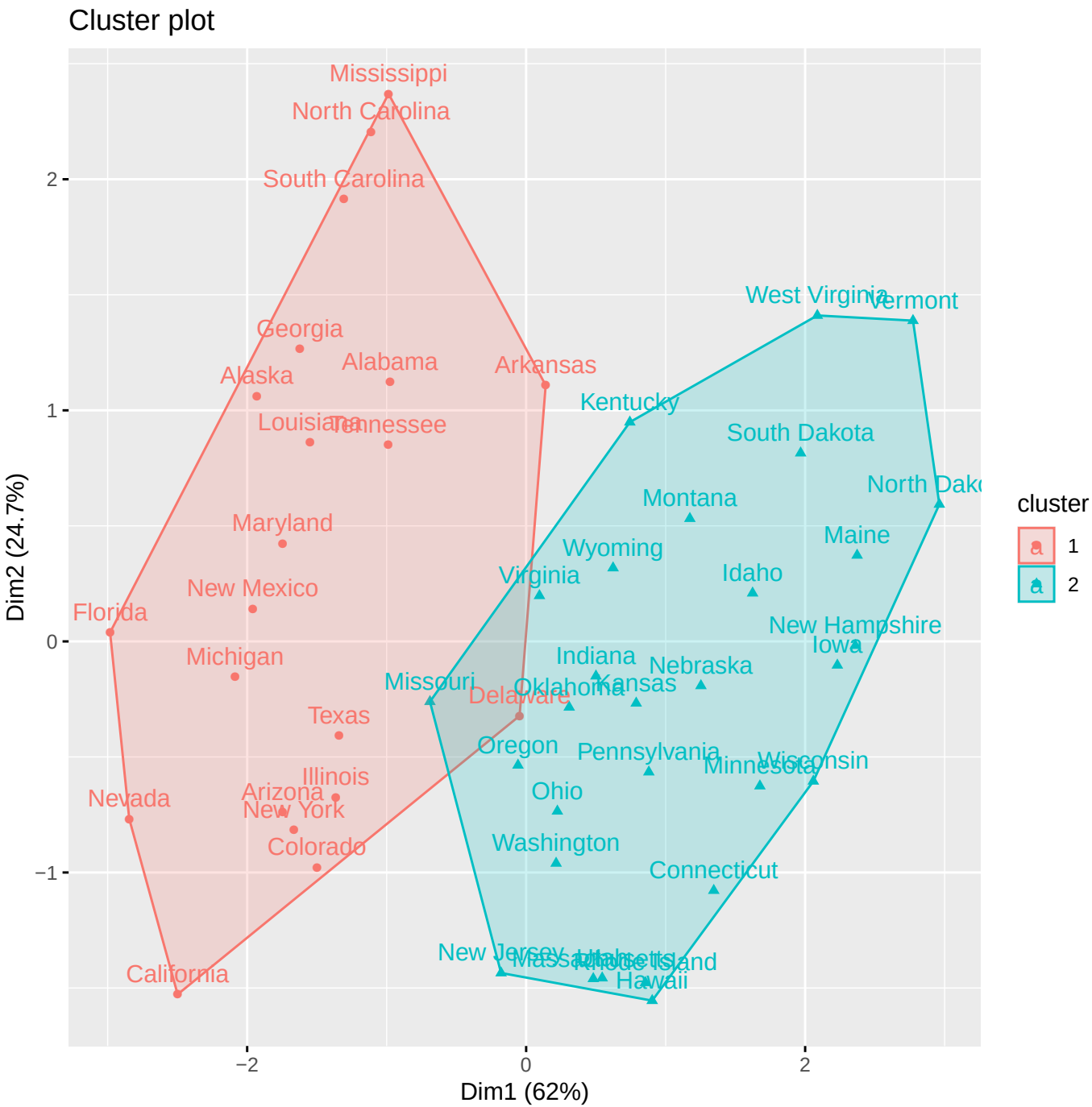
To show the clusters to which the states belong we write:

```
> dd <- cbind(USArrests, cluster = pam.res$cluster)
> head(dd,n=3)
        Murder Assault UrbanPop Rape cluster
Alabama   13.2     236       58 21.2       1
Alaska    10.0     263       48 44.5       1
Arizona    8.1     294       80 31.0       1
> pam.res$medoids
         Murder Assault UrbanPop Rape
Michigan   12.1     255       74 35.1
Kansas      6.0     115       66 18.0
```

The clustering can be visualized using the function `fviz_cluster` of the package `factoextra` invoked as

```
fviz_cluster(pam.res, geom = "point", ellipse.type = "norm")
```

The result is presented in the next slide, where states are reprsented as points using their first two principal components.


Cluster plot
Mississippi
North Carolina
South Carolina
Georgia
Alabama
Arkansas
Alaska
Louisiana
Tennessee
Maryland
New Mexico
Florida
Michigan
Texas
Illinois
Nevada
Arizona
New York
Colorado
California
West Virginia
Vermont
Kentucky
South Dakota
Montana
Maine
Wyoming
Idaho
Virginia
New Hampshire
Iowa
Indiana
Nebraska
Missouri
Oklahoma
Kansas
Delaware
Oregon
Pennsylvania
Minnesota
Wisconsin
Ohio
Washington
Connecticut
Hawaii
cluster
1
2
Dim2 (24.7%)
Dim1 (62%)
2
1
0
−1
−2
0
2