Unsupervised Learning: Clustering

Dr Liew How Hui

Jan 2022

On Assignment

Oral Presentation:

- 18 minutes presentation.
- 2 minutes Q&A.
- The presentation will be recorded.
- Submission of presentation slides after presentation.

Assignment submission deadline:

- Week 11 Wednesday 4pm.
- Report and Computer Programs in proper format

On Final Exam

Final Assessment/Exam:

- Check your Timetable & Set the Alarm!!!
- Compulsory questions (Q1,Q2,Q3) + Q4 or Q5 (Choose 1)
- Write down your INDEX number (otherwise your exam marks cannot be keyed into UTAR system)
- Write answers in Answer Booklet.

Outline

1. Theory
2. Partitioning Methods
    - k-Means, PAM, ...
    - GMM (and SOM)
3. Hierarchical Methods: AGNES, DIANA, ...
4. Density-Based Methods
5. Grid-Based Methods
6. Kernel and Spectral Based Methods

Theory

Clustering is a broad class of unsupervised learning methods for finding "clusters" by labelling data with integers.

Clustering methods for static data can be classified into

- Partitioning methods
- Hierarchical methods
- Density-Based methods
- Grid-Based methods
- Kernel and Spectral methods

Hidden Markov models: uses observed data to recover the sequence of states → cluster sequences

Theory (cont)

Cluster analysis's primary purpose:

- group objects based on the "similarity" in the features
- reduce the information from an entire sample into information about smaller, manageable subgroups.

A good clustering algorithm produces high quality clusters with

- high intra-cluster (within-cluster) similarity
- low inter-cluster (between-cluster) similarity

Theory (cont)

A “good” clustering need a good measure of “dissimilarity”, in particular, “distance” measures.

Measurement of “dissimilarity”: A *dissimilarity (function)* $d(\mathrm{x}_i, \mathrm{x}_j)$ is a function which satisfies the following conditions: For any x_i, x_j, x_k,

(i) $d(\mathrm{x}_i, \mathrm{x}_j) \geq 0$ and $d(\mathrm{x}_i, \mathrm{x}_j) = 0$ iff $\mathrm{x}_i = \mathrm{x}_j$;

(ii) $d(\mathrm{x}_i, \mathrm{x}_j) = d(\mathrm{x}_j, \mathrm{x}_i)$; and

A *distance function* (defined in Topic 2) is a dissimilarity function which satisfies

(iii) $d(\mathrm{x}_i, \mathrm{x}_j) \leq d(\mathrm{x}_i, \mathrm{x}_k) + d(\mathrm{x}_k, \mathrm{x}_j)$ (triangle inequality).

Theory (cont)

Minkowski distance (Topic 2):

$$d(\mathrm{x},\mathrm{z}) = \|\mathrm{x}-\mathrm{z}\|_r = \left(\sum_{i=1}^{p} |x_i - z_i|^r\right)^{\frac{1}{r}}, \quad \mathrm{x},\ \mathrm{z} \in \mathbb{R}^p. \tag{1}$$

- $r = 1$: Manhattan distance
- $r = 2$: Euclidean distance
- $r = \infty$: Chebyshev distance

Theory (cont)

Cosine Dissimilarity:

$$\text{cos-dist}(\mathrm{x}, \mathrm{y}) = 1 - \frac{\langle \mathrm{x}, \mathrm{y} \rangle}{\|\mathrm{x}\|_2 \cdot \|\mathrm{y}\|_2} = 1 - \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}}$$

and it is generally used as a metric for measuring distance when the *magnitude of the vectors does not matter*.

R's implementations: `proxy::dist(x, method="cosine")`, `text2vec::sim2(x,method="cosine")`, `lsa::cosine()`, `clv::dot_product()`, `rules:: dissimilarity()`

Theory: Dissimilarity / Distance Matrix

Suppose we have n data samples $\mathrm{x}_i = (x_{i1}, \cdots, x_{ip})$, $i = 1, \cdots, n$. Stacking the sample data x_i leads to a matrix

$$\mathrm{X} = \begin{bmatrix} \mathrm{x}_1 \\ \mathrm{x}_2 \\ \vdots \\ \mathrm{x}_n \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & \ddots & & x_{2p} \\ \vdots & & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

where x_{ij} is the ith sample/observation for the jth variable/feature.

Dissimilarity / Distance Matrix (cont)

The *distance (or dissimilarity) matrix* between observations can be measured by a dissimilarity function d as

$$D = \begin{bmatrix} 0 & d_{12} & \cdots & d_{1n} \\ d_{21} & 0 & & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & 0 \end{bmatrix} \tag{2}$$

where $d_{ij} = d(\mathrm{x}_i, \mathrm{x}_j)$, $h, k = 1, \cdots, n$.

Distance Matrix in R

The basic distance matrix in R has the form:

```
dist(x, method = "euclidean", diag = FALSE,
     upper = FALSE, p = 2)
```

Apart from ‘euclidean’, other ‘method’s are ‘maximum’, ‘manhattan’, ‘canberra, ‘binary’, ‘minkowski’.

The `cluster::daisy` function supports euclidean, manhattan and gower.

```
daisy(x, metric = c("euclidean","manhattan","gower"),
      stand=FALSE, type=list(), weights=rep.int(1,p),
      warnBin = warnType, warnAsym = warnType,
      warnConst = warnType, warnType = TRUE)
```

When ‘stand’ is set TRUE and all ‘x’s are numeric, the measurements in ‘x’ are standardised before calculating the dissimilarities.

Distance Matrix in Python

The Scikit-learn library has a richer variety of distance measures.

```
sklearn.metrics.pairwise_distances(X, Y=None,
  metric=’euclidean’, *, n_jobs=None,
  force_all_finite=True, **kwds)
```

The “metric” options include euclidean/l2, manhattan/l1/cityblock, cosine, braycurtis, canberra, chebyshev, correlation, dice, hamming, jaccard, kulsinski, mahalanobis, minkowski, rogestanimoto, russellrao, seuclidean, sokalmichener, sokalsneath, sqeuclidean, yule.

Outline

1. Theory
2. Partitioning Methods
    - k-Means, PAM, ...
    - GMM (and SOM)
3. Hierarchical Methods: AGNES, DIANA, ...
4. Density-Based Methods
5. Grid-Based Methods
6. Kernel and Spectral Based Methods

Partitioning Methods

Basic Idea:

- Partition n observations into k clusters.
- Try all possible partitions of observations.

Partitioning Methods (cont)

- *k-Means clustering*: partitions data into *k* distinct clusters based on distance to the cluster centroid
- *k*-Medoids clustering: partitions around medroids (PAM), clustering large application (CLARA) and its variant CLARANS, etc.
- Gaussian mixture models: models clusters as a mixture of multivariate normal density components.
- Self-organising maps: uses neural networks that learn the topology and distribution of the data.

k-Means Clustering

- Most popular clustering method!
- It partitions a data set of n observations into K distinct, non-overlapping clusters (groups) $C_1, \cdots, C_K$ with the nearest mean, serving as a prototype for the cluster. It tries to find a "cluster" to optimise

$$\min_{C_1,\cdots,C_K}\left\{\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{i,j\in C_k} d(\mathrm{x}_i,\mathrm{x}_j)^2\right\}.$$

Note that $WSS_k = \sum_{i,j\in C_k} d(\mathrm{x}_i,\mathrm{x}_j)^2$ is called the *within sum of squares*.

k-Means Clustering (cont)

After the desired number of clusters k is specified, the *(naïve) k-means algorithm* below can be applied to perform the k-means clustering:

1. Randomly pick K observations as the initial centroid(s) c_i.
2. Iterate until the centroid(s) stop changing:
    1. Assign each observation to the cluster with the **nearest centroid**:
    $$C_i^{(t)} = \left\{\mathrm{x}_p : d(\mathrm{x}_p, \mathrm{c}_i^{(t)}) \le d(\mathrm{x}_p, \mathrm{c}_j^{(t)}), \quad \forall j, 1 \le j \le k\right\},$$
    2. Recalculate centroids for observations assigned to each cluster:
    $$\mathrm{c}_i^{(t+1)} = \frac{1}{\left|C_i^{(t)}\right|} \sum_{x_j \in C_i^{(t)}} \mathrm{x}_j.$$

k-Means Clustering (cont)

Software Implementations:

- R's implementation has the form:

```
kmeans(x, centers, iter.max = 10, nstart = 1, trace=FALSE,
  algorithm=c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"))
```

  It uses *Hartigan-Wong method* (see `https://en.wikipedia.org/wiki/K-means_clustering`).

- The official Python implementation has the following form:

```
sklearn.cluster.KMeans(n_clusters=8, *, init=’k-means++’,
  n_init=10, max_iter=300, tol=0.0001, verbose=0)
```

  where ‘`k-means++`’ is the method for initialisation which selects initial cluster centres for k-mean clustering in a smart way to speed up convergence. Alternative are ‘random’, which chooses n_clusters observations (rows) at random from data for the initial centroids; and an `ndarray` in the shape of (n_clusters, n_features) which gives the initial centres.

k-Means Clustering (cont)

Example 12.3.1:

You are given the following observations with two variables, x_1 and x_2.

Obs	x_1	x_2
A	1	1
B	1	2
C	2	1
D	2	2
E	8	8
F	8	9
G	9	8
H	9	9

k-Means Clustering (cont)

Example 12.3.1 (cont):
Perform *k*-means clustering to group the observations into two groups using Euclidean distance. Given that the random initial centroids be D and A.

k-Means Clustering (cont)

Example 12.3.1 (cont):

$t = 0$: $\mathrm{c}_1^{(0)} = (2, 2)$, $\mathrm{c}_2^{(0)} = (1, 1)$

Obs	x	y	distance from c_1	distance from c_2	closest
A	1	1	1.414214	0.000000	2
B	1	2	1.000000	1.000000	1
C	2	1	1.000000	1.000000	1
D	2	2	0.000000	1.414214	1
E	8	8	8.485281	9.899495	1
F	8	9	9.219544	10.630146	1
G	9	8	9.219544	10.630146	1
H	9	9	9.899495	11.313708	1

k-Means Clustering (cont)

Example 12.3.1 (cont):

$t = 1$: $c_1^{(1)} = \frac{(1,2)+\ldots+(9,9)}{7} = (5.571429, 5.571429)$, $c_2^{(1)} = (1, 1)$

Obs	x	y	distance from c_1	distance from c_2	closest
A	1	1	6.464976	0.000000	2
B	1	2	5.801126	1.000000	2
C	2	1	5.801126	1.000000	2
D	2	2	5.050763	1.414214	2
E	8	8	3.434519	9.899495	1
F	8	9	4.201555	10.630146	1
G	9	8	4.201555	10.630146	1
H	9	9	4.848732	11.313708	1

k-Means Clustering (cont)

Example 12.3.1 (cont):

$t = 2$: $\mathrm{c}_1^{(1)} = \frac{(8,8)+\ldots+(9,9)}{4} = (8.5, 8.5)$,

$\mathrm{c}_2^{(1)} = \frac{(1,1)+\ldots+(2,2)}{4} = (1.5, 1.5)$

Obs	x	y	distance from c_1	distance from c_2	closest
A	1	1	10.6066017	0.7071068	2
B	1	2	9.9247166	0.7071068	2
C	2	1	9.9247166	0.7071068	2
D	2	2	9.1923882	0.7071068	2
E	8	8	0.7071068	9.1923882	1
F	8	9	0.7071068	9.9247166	1
G	9	8	0.7071068	9.9247166	1
H	9	9	0.7071068	10.6066017	1

Stop when clusters / centroids unchanged.

k-Means Clustering (cont)

Example 12.3.2: You are given the following observations with two variables, x_1 and x_2.

Obs	x_1	x_2
A	0	0
B	0	2
C	20	0
D	20	2
E	80	8
F	80	10
G	100	8
H	100	10

Use B and E as initial centroids to find 2 clusters using k-means clustering.

k-Means Clustering (cont)

Example 12.3.2 (cont):

Try:

$t = 0$:
$c_1^{(0)} = (0, 2)$, $c_2^{(0)} = (80, 8)$
$t = 1$:
$$c_1^{(1)} = \frac{(0,0)+\ldots+(20,2)}{4} = (10, 1),$$
$$c_2^{(1)} = \frac{(80,8)+\ldots+(100,10)}{4} = (90, 9)$$
$t = 2$:
$$c_1^{(2)} = \frac{(0,0)+\ldots+(20,2)}{4} = (10, 1),$$
$$c_2^{(2)} = \frac{(80,8)+\ldots+(100,10)}{4} = (90, 9)$$
Stop since no changes in clusters.

k-Means Clustering (cont)

Example 12.4.1: By using k-means clustering with B and E as initial centroids to find 2 clusters for the following data:

Obs	x_1	x_2
A	0	0
B	0	2
C	20	0
D	20	2
E	80	8
F	80	10
G	100	8
H	100	10
I	10	7
J	30	2
K	40	9
L	60	1
M	70	8
N	90	3

(a) Rescale both variables using min-max normalisation and perform *k*-means clustering using B and E as initial centres.

(b) Rescale both variables x_1 and x_2 using standardisation and perform *k*-means clustering using B and E as initial centres.

k-Means Clustering (cont)

Example 12.4.1 (a): After the min-max scaling and performing k-mean clustering with the initial centres: $c_1 = (0.2, 0.2), \quad c_2 = (0.8, 0.8)$

	x'_1	x'_2	dist1	dist2	cluster
A	0.0	0.0	0.2828427	1.1313708	1
B	0.0	0.2	0.2000000	1.0000000	1
C	0.2	0.0	0.2000000	1.0000000	1
D	0.2	0.2	0.0000000	0.8485281	1
E	0.8	0.8	0.8485281	0.0000000	2
F	0.8	1.0	1.0000000	0.2000000	2
G	1.0	0.8	1.0000000	0.2000000	2
H	1.0	1.0	1.1313708	0.2828427	2
I	0.1	0.7	0.5099020	0.7071068	1
J	0.3	0.2	0.1000000	0.7810250	1
K	0.4	0.9	0.7280110	0.4123106	2
L	0.6	0.1	0.4123106	0.7280110	1
M	0.7	0.8	0.7810250	0.1000000	2
N	0.9	0.3	0.7071068	0.5099020	2

k-Means Clustering (cont)

Example 12.4.1 (b): First calculate the mean

$$\overline{X}_1 = \frac{0 + ... + 90}{14} = 50; \quad \overline{X}_2 = \frac{0 + ... + 3}{14} = 5$$

and then the (sample) standard deviation

$$\begin{aligned}(s_X)_1 &= \sqrt{\frac{(0-50)^2 + \cdots + (90-50)^2}{14-1}} \\ &= \sqrt{1942.857143} = 37.00312 \\ (s_X)_2 &= \sqrt{\frac{(0-5)^2 + \cdots + (3-5)^2}{14-1}} = 3.86304\end{aligned}$$

k-Means Clustering (cont)

Example 12.4.1 (b) cont: The initial centres are

$$c_1 = (-0.8107425, -0.7765905),$$
$$c_2 = (0.8107425, 0.7765905)$$

Obs	$\tilde{x}_1$	$\tilde{x}_2$	$d(\tilde{x}_1, c_1)$	$d(\tilde{x}_1, c_2)$	cluster
A	-1.3512375	-1.2943175	0.7484491	2.9937964	1
B	-1.3512375	-0.7765905	0.5404950	2.6620534	1
C	-0.8107425	-1.2943175	0.5177270	2.6301850	1
D	-0.8107425	-0.7765905	0.0000000	2.2453473	1
E	0.8107425	0.7765905	2.2453473	0.0000000	2
F	0.8107425	1.2943175	2.6301850	0.5177270	2
G	1.3512375	0.7765905	2.6620534	0.5404950	2
H	1.3512375	1.2943175	2.9937964	0.7484491	2
I	-1.0809900	0.5177270	1.3222297	1.9093617	1
J	-0.5404950	-0.7765905	0.2702475	2.0586923	1
K	-0.2702475	1.0354540	1.8909363	1.1115528	2
L	0.2702475	-1.0354540	1.1115528	1.8909363	1
M	0.5404950	0.7765905	2.0586923	0.2702475	2
N	1.0809900	-0.5177270	1.9093617	1.3222297	2

k-Means Clustering (cont)

Final Exam May 2019, Q1(a):

(i) Normalisation is normally applied to the algorithm involved distance measures such as clustering. State the reason why normalization is required and how it helps. (2 marks)

Solution

The distance measures are affected by the scale of the variables. By putting all variables into the same range (normalize), the variables will be weighted equally.

(ii) State two types of normalization. Explain on how the normalisation works. (4 marks)

Solution

Min-max normalisation transforms all variables into an interval [0,1] by using

$$M(X_{ij}) = \frac{X_{ij} - X_{\text{min,j}}}{X_{\text{max,j}} - X_{\text{min,j}}}.$$

Standardisation transforms all variables to a standard scale with mean of zero and standard deviation of one by using

$$M(X_{ij}) = \frac{X_{ij} - \overline{X}_j}{s_{X,j}}$$

k-Means Clustering (cont)

Final Exam Jan 2019, Q1(b):

A factory doing oranges packaging would like to group their oranges according to oranges' weight and diameter. Table Q1(b) - i shows the weight and diameter of the oranges.

Orange	Weight (g)	Diameter (cm)
1	108	7.26
2	81	4.61
3	132	6.92
4	118	5.02
5	126	4.16
6	94	5.12
7	150	5.53
8	86	5.10

Table Q1(b) - i

k-Means Clustering (cont)

Final Exam Jan 2019, Q1(b) cont:
With standardisation, *k*-means clustering has been performed to group the oranges into three clusters (A, B, C). Table Q1(b) - ii shows part of the results in final iteration of this *k*-means clustering.

Orange	Distance A	Distance B	Distance C
1	2.3802	2.3082	0.5228
2	2.1092	0.3959	2.8035
3	1.8604	2.6118	0.5228
4	0.5645	1.2903	1.9111
5	0.7206	1.7746	2.7141
6	1.5644	0.3334	2.1141
7	0.9674	2.6736	1.9039
8	1.8927	0.1504	2.3164

Table Q1(b) - ii

k-Means Clustering (cont)

Final Exam Jan 2019, Q1(b) cont:

(i) Group the oranges into appropriate cluster and state the centroids for each cluster formed. (6 marks)

(ii) Based on (i), calculate the within cluster sum of square (wss) for each cluster. (6 marks)

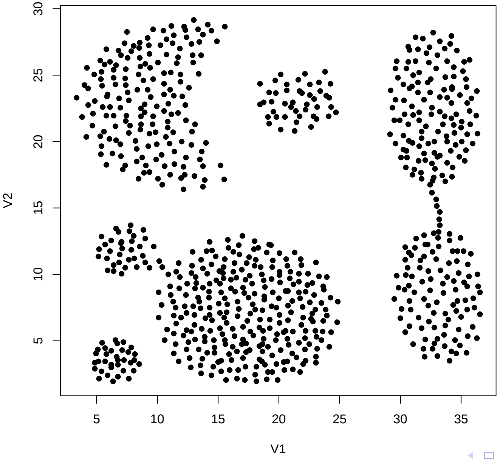
k-Means Clustering (cont)

One of the requirements to perform k-means clustering is a pre-determined number of clusters.
Sometimes, appropriate k is known.
Most of the times, k is unknown. Use “elbow principle”: Observations within clusters will be close together when plotted geometrically. We can make use of this property to determine the optimum number of clusters to be formed. An optimum number of clusters is the one which *within groups sum of squares (wss)* decreases dramatically as compared to the previous number, and only decreased a little as compared to the after number.

k-Means Clustering (cont)

Elbow principle is not science.

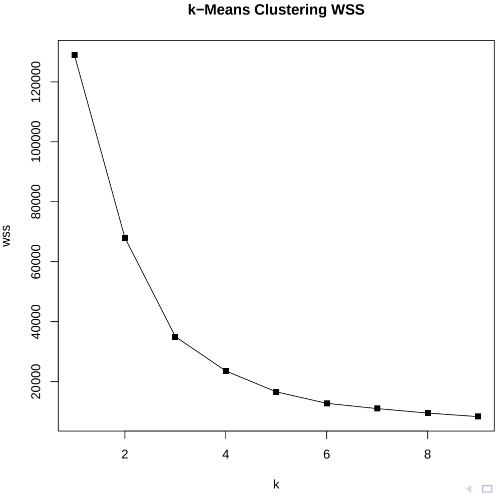
Example: Consider the following data:



k-Means Clustering (cont)

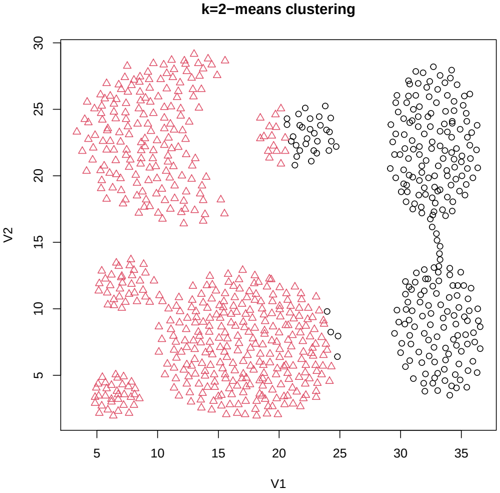
Elbow principle is not science.

Example (cont):



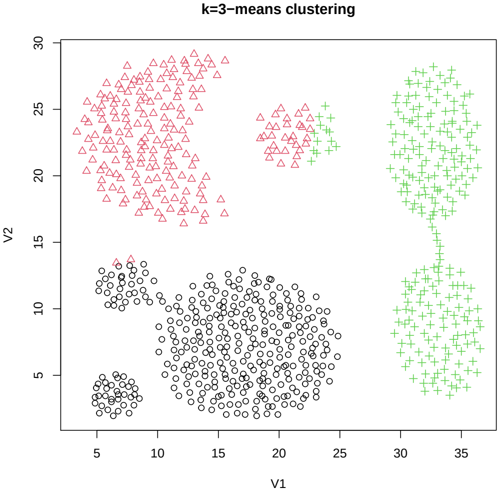
k-Means Clustering (cont)

Example (cont):



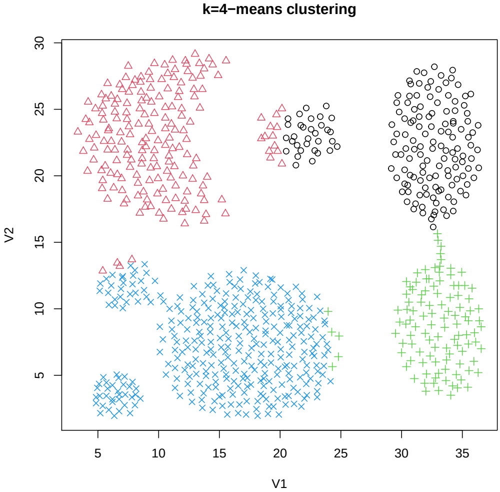
k-Means Clustering (cont)

Example (cont):



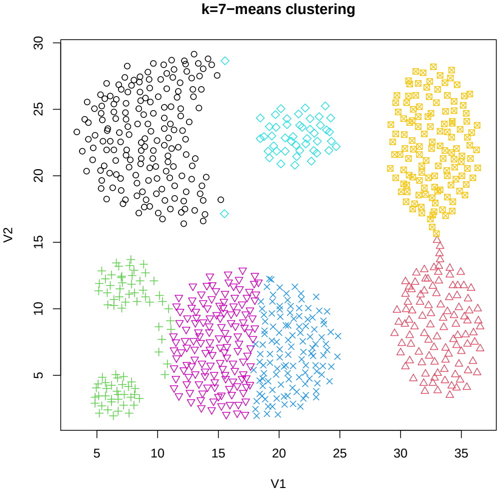
k-Means Clustering (cont)

Example (cont):



k-Means Clustering (cont)

Example (cont):



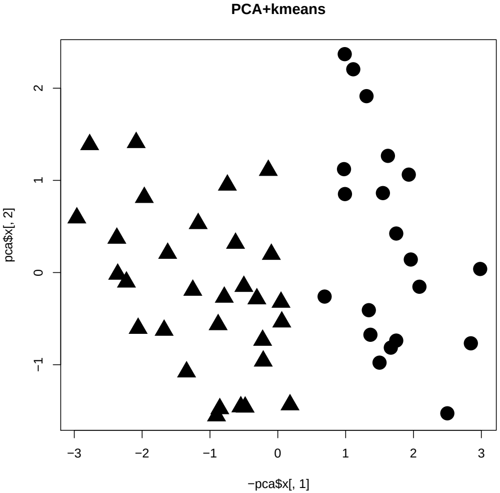
k-Means Clustering (cont)

Example (4D data — USArrest):

```
data(USArrests)
X = scale(USArrests)
pca = prcomp(X)
kmeans.m = kmeans(X, centers=2)  # centers=k
# The eigenvectors differ by a negative sign can cause funny effect
pdf("usarrests_kmeans1.pdf")
plot(-pca$x[,1], pca$x[,2], pch=kmeans.m$cluster+15,
  main="PCA+kmeans", cex=2.5)
library(cluster)
pdf("usarrests_kmeans2.pdf")
clusplot(X, kmeans.m$cluster, main="PCA+PAM???", cex=2.5)
```

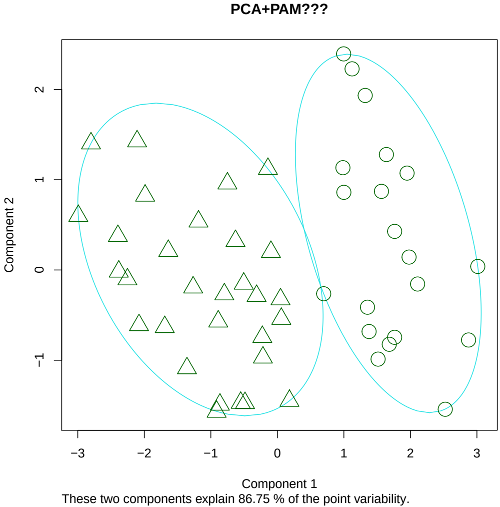
k-Means Clustering (cont)

Example (4D data) cont: PCA + k-means ($k = 2$)



k-Means Clustering (cont)

Example (4D data) cont: Using clusplot from R's cluster



k-Prototype Clustering

The k-Means only deals with numeric data. The k-Modes deals with categorical data. The k-Prototype clustering merges k-Means and k-Modes.

The applications involving mixed data is the ‘spatial clustering’ in hotspot detection:

- pollution analysis: Malaysia’s major concerned, a million of Selangor residents were facing the cease of water supply due to water pollution in Sept 2020 during COVID-19 pendanmic;
- disease analysis: Flu, influensa, Denggi, foot-and-mouth disease;
- crime analysis;
- fire analysis, etc.

k-Prototype Clustering (cont)

R's support: clustMixType::kproto (slow as snail!)

```
kp = kproto(x, k, lambda=NULL, iter.max=100,
  nstart=1, na.rm=TRUE, keep.data=TRUE,
  verbose=TRUE, ...)
```

Partitioning Around Medoids (PAM)

PAM algorithm is the most popular k-medoids algorithm. The k-medoids algorithm is a clustering approach related to k-means clustering for partitioning a data set into k groups or clusters. In k-medoids clustering, each cluster is represented by one of the data point (called “cluster medoid”) in the cluster.

K-medoid is a robust alternative to k-means clustering. This means that, the algorithm is **less sensitive to noise and outliers, compared to k-means**, because it uses medoids as cluster centres instead of means (used in k-means).

PAM (cont)

PAM algorithm:

1. Select k objects to become the medoids, or in case these objects were provided use them as the medoids;
2. Calculate the dissimilarity matrix if it was not provided;
3. Assign every object to its closest medoid;
4. For each cluster search if any of the object of the cluster decreases the average dissimilarity coefficient; if it does, select the entity that decreases this coefficient the most as the medoid for this cluster;
5. If at least one medoid has changed go to 3., else end the algorithm.

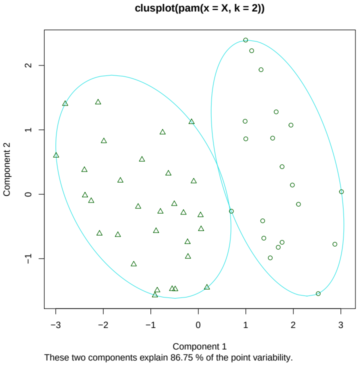
PAM (cont)

Example (4D data — USArrest):

```
# https://www.datanovia.com/en/lessons/k-medoids-in-r-algorithm-and-p
data(USArrests)
X = scale(USArrests)
library(cluster)
pam.m = pam(X, k=2)
pam.m$medoids
clusplot(pam.m)
#pdf("usarrests_pam.pdf")
#plot(pam.m, ask=FALSE, which.plots=1)   # call: plot.partition
#x11()
pdf("usarrests_pam_silh.pdf")
plot(pam.m, ask=FALSE, which.plots=2)   # call: plot.partition
```

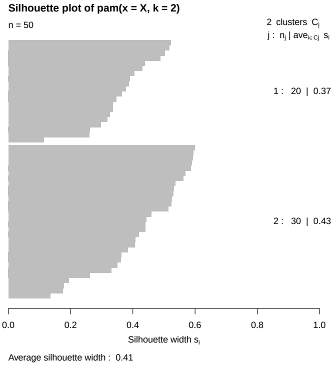
PAM (cont)

Example (4D data — USArrest) cont:



clusplot(pam(x = X, k = 2))

These two components explain 86.75 % of the point variability.



Silhouette plot of pam(x = X, k = 2)

Average silhouette width : 0.41

PAM (cont)

Pros and Cons of PAM

- PAM is more robust than k-means in the presence of noise and outliers — Medoids are less influenced by outliers
- PAM is efficiently for small data sets but does not scale well for large data sets — $O(k(n-k)^2)$ for each iteration
- Sampling based method: CLARA

CLARA

CLARA = Clustering LARge Applications (Kaufmann and Rousseeuw in 1990)

- Draw multiple samples of the data set, apply PAM on each sample, give the best clustering
- Perform better than PAM in larger data sets
- Efficiency depends on the sample size — A good clustering on samples may not be a good clustering of the whole data set

CLARANS

CLARANS = Clustering Large Applications based upon RAndomized Search

- The problem space graph of clustering
  - A vertex is k from n numbers, $\binom{n}{k}$ vertices in total
  - PAM searches the whole graph
  - CLARA searches some random sub-graphs
- CLARANS climbs hills
  - Randomly sample a set and select k medoids
  - Consider neighbours of medoids as candidate for new medoids
  - Use the sample set to verify
  - Repeat multiple times to avoid bad samples

FANNY

FANNY / Fuzzy C-Means = Fuzzy ANalysis Clustering

- Each point x_i takes a probability w_{ij} to belong to a cluster C_j
- For each point x_i, $\sum_{j=1}^{k} w_{ij} = 1$
- For each cluster C_j, $0 < \sum_{i=1}^{n} w_{ij} < n$.

Cons: The methods 'k-means', 'pam', 'clara' and 'fanny' require that the number of clusters be given by the user.

FANNY (cont)

This is different from k-means and k-medoid clustering, where each object is affected exactly to one cluster. K-means and k-medoids clustering are known as hard or non-fuzzy clustering.
In fuzzy clustering, points close to the centre of a cluster, may be in the cluster to a higher degree than points in the edge of a cluster. The degree, to which an element belongs to a given cluster, is a numerical value varying from 0 to 1.

FANNY (cont)

Fuzzy C-Means algorithm:

- Select an initial fuzzy pseudo-partition, i.e., assign values to all the w_{ij}
- Repeat
  - Compute the centroid of each cluster using the fuzzy pseudo-partition
  - Recompute the fuzzy pseudo-partition, i.e., the w_{ij}
- Until the centroids do not change (or the change is below some threshold)

FANNY (cont)

Critical Details of the Algorithm:

- Optimisation on sum of the squared error (SSE):

$$SSE(C_1, \ldots, C_k) = \sum_{j=1}^{k} \sum_{i=1}^{n} w_{ij}^{p} d(\mathrm{x}_i, \mathrm{c}_j)^2$$

- Computing centroids: $\mathrm{c}_j = \sum_{i=1}^{n} w_{ij}^{p} \mathrm{x}_i / \sum_{i=1}^{n} w_{ij}^{p}$
- Updating the fuzzy pseudo-partition

$$w_{ij} = \left(\frac{1}{d(\mathrm{x}_i, \mathrm{c}_j)^2}\right)^{1/(p-1)} \Bigg/ \sum_{q=1}^{k} \left(\frac{1}{d(\mathrm{x}_i, \mathrm{c}_q)^2}\right)^{1/(p-1)}$$

FANNY (cont)

Choice of p in the Algorithm:

- When $p \to 1$, FCM behaves like traditional k-means
- When p is larger, the cluster centroids approach the global centroid of all data points
- The partition becomes fuzzier as p increases

Gaussian Mixture Models (GMM)

A **mixture model** is a mixture of k component distributions that collectively make a mixture distribution:

$$f(x) = \sum_{i=1}^{k} \alpha_i f_i(x).$$

The α_i represents a mixing weight for the ith component where $\sum_{i=1}^{k} \alpha_i = 1$. The $f_i(x)$ components in principle are arbitrary in the sense that you can choose any sort of distribution.

For continuous case, it has the form

$$g(x) = \int_{\Theta} f(x|\alpha)\rho(\alpha)d\alpha.$$

GMM (cont)

In practice, parametric distribution (e.g. gaussians), are often used since a lot work has been done to understand their behaviour.

If we substitute each $f_i(x)$ for a gaussian we get the **gaussian mixture models (GMM)**:

$$
\begin{aligned}
&f(\mathrm{X} = \mathrm{x}, Y = k) \\
&= \sum_{i=1}^{k} \alpha_i \frac{1}{\sqrt{(2\pi)^p |\Sigma_i|}} \exp(-\frac{1}{2}(\mathrm{x} - \mu_i)\Sigma_i^{-1}(\mathrm{x} - \mu_i)^T).
\end{aligned}
$$

Likewise, if we substitute each $f_k(x)$ for a binomial distribution, we get a *binomial mixture model (BMM)*.

GMM (cont)

Implementations:

- R: mclust, mixtools, plotGMM(?) library
- Python: sklearn.mixture.GaussianMixture

Example:

```
# https://cran.r-project.org/web/packages/sBIC/vignettes/GaussianMixt
library(MASS)
data(galaxies)
# hist(galaxies)
X = galaxies/1000

library(mclust, quietly=TRUE)
gmm.model = Mclust(X, G=4, model="V")
summary(gmm.model)
plot(gmm.model, what="density", main="", xlab="Velocity (Mm/s)")
rug(X)
```

GMM (cont)

Expectation maximisation (EM) clustering algorithm

$$
\begin{aligned}
&f(\mathrm{x}; \alpha_1, \ldots, \alpha_{k-1}, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k) \\
&= \sum_{i=1}^{k} \alpha_i \phi_i(\mathrm{x}; \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k)
\end{aligned}
$$

is used in mixture model for computation.
Implementation:
https://cran.r-project.org/web/packages/EMCluster/

GMM (cont)

Expectation Maximisation (EM) Algorithm:

- Select an initial set of model parameters
- Repeat
  - Expectation Step: for each object, calculate the probability that it belongs to each distribution i, i.e., $\mathbb{P}(x_i|i)$
  - Maximisation Step: given the probabilities from the expectation step, find the new estimates of the parameters that maximise the expected likelihood
- Until the parameters are stable

GMM (cont)

The EM algorithm can be used to learn the parameters of a Gaussian mixture model. For this model, we assume a generative process for the data as follows:

$$\mathrm{x}_i|c_i \sim Normal(\mu_{c_i}, \Sigma_{c_i}), \quad c_i \sim Discrete(p)$$

where c_i are one of the $1, ..., K$ class, p is some probability vector.

The EM algorithm seeks to maximise the likelihood

$$f(\mathrm{x}_1, \cdots, \mathrm{x}_n|p, \mu, \Sigma) = \prod_{i=1}^{n} f(\mathrm{x}_i|p, \mu, \Sigma)$$

over all parameters p, $\mu_1, ..., \mu_K, \Sigma_1, ..., \Sigma_K$ using the cluster assignments $c_1, ..., c_n$ as the hidden data.

GMM (cont)

Maximising the log-likelihood:

1. E-step: For $i = 1, ..., n$, set

$$\phi_i(k) = \frac{p_k N(x_i|\mu_k, \Sigma_k)}{\sum_j p_j N(x_i|\mu_j, \Sigma_j)}, \quad k = 1, \ldots, K.$$

2. M-step: For $k = 1, \ldots, K$, define $n_k = \sum_{i=1}^n \phi_i(k)$ and update the values

$$p_k = \frac{n_k}{n}, \ \mu_k = \sum_{i=1}^{n} \phi_i(k) \mathrm{x}_i,$$

$$\Sigma_k = \frac{1}{n_k} \sum_{i=1}^{n} \phi_i(k)(\mathrm{x}_i - \mu_k)(\mathrm{x}_i - \mu_k)^T.$$

GMM (cont)

Advantages

- Mixture models are more general than *k*-means and FANNY clustering
- Clusters can be characterised by a small number of parameters
- The results may satisfy the statistical assumptions of the generative models

Disadvantages:

- Computationally expensive
- Need large data sets
- Hard to estimate the number of clusters

Self-Organising Map (SOM)

It is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a low-dimensional (typically 2D), discretized representation of the input space of the training samples, called a map, and is therefore a method to do dimensionality reduction. SOM differ from other artificial neural networks as they apply competitive learning as opposed to error-correction learning (such as backpropagation with gradient descent), and in the sense that they use a neighbourhood function to preserve the topological properties of the input space.

Outline

1. Theory
2. Partitioning Methods
   - k-Means, PAM, ...
   - GMM (and SOM)
3. Hierarchical Methods: AGNES, DIANA, ...
4. Density-Based Methods
5. Grid-Based Methods
6. Kernel and Spectral Based Methods

Hierarchical Methods

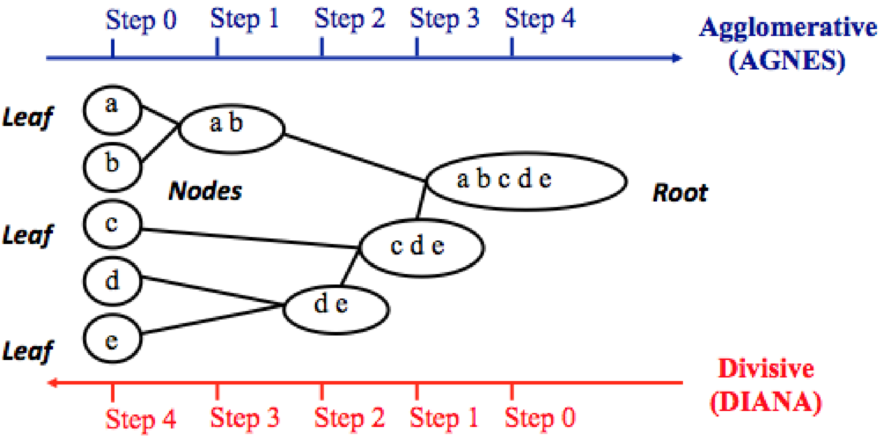
Hierarchical methods construct a hierarchy of clusterings, with the number of clusters ranging from one to the number of observations.

- Arrange or classify things according to inclusiveness
- A natural way of abstraction, summarisation, compression, and simplification for understanding
- Typical setting: organise a given set of objects to a hierarchy
  - No or very little supervision
  - Some heuristic quality guidances on the quality of the hierarchy

Hierarchical Methods (cont)

- *Agglomerative Nesting* (AGNES) or *Hierarchical (Agglomerative) Clustering* (HAC): builds a multilevel hierarchy of clusters from **points** to **clusters**.
- *Divisive Hierarchical Clustering* or *Divisive Analysis (DIANA)*: It breaks a single cluster to many clusters
- Clustering Using REpresentatives: See `https://en.wikipedia.org/wiki/CURE_algorithm`
- Chameleon Clustering: See `http://mlwiki.org/index.php/Chameleon_Clustering`

Hierarchical Methods (cont)



Source: `https://uc-r.github.io/hc_clustering`

AGNES Clustering

Agglomerative Nesting (AGNES) is implemented in R as `hclust` or `cluster::agnes`

The `agnes` is fully described in chapter 5 of Kaufman and Rousseeuw (1990, Finding Groups in Data: An Introduction to Cluster Analysis).

Compared to other agglomerative clustering methods such as `hclust`, agnes has the following features: (a) it yields the agglomerative coefficient (see agnes.object) which measures the amount of clustering structure found; and (b) apart from the usual tree it also provides the banner, a novel graphical display (see plot.agnes).

AGNES Clustering (cont)

Dissimilarity is defined for a **pair of observations**. It needs to be extended to a pair of groups of observations, called **linkage**, which defines the dissimilarity between **two groups of observations** A and B.

Commonly used linkage criteria are:

(a) Minimum or single-linkage clustering: $\min\{\, d(a,b) : a \in A,\ b \in B \,\}$.

(b) Maximum or complete-linkage clustering: $\max\{\, d(a,b) : a \in A,\ b \in B \,\}$.

(c) (Unweighted) average linkage clustering (or UPGMA): $\frac{1}{|A|\cdot|B|}\sum_{a\in A}\sum_{b\in B} d(a,b)$

AGNES Clustering (cont)

(c) Weighted average linkage clustering (or WPGMA): $d(i \cup j, k) = \frac{d(i,k)+d(j,k)}{2}$.

(d) Centroid linkage clustering, or UPGMC: $\|c_s - c_t\|$ where c_s and c_t are the centroids of clusters s and t, respectively.

(e) Minimum energy clustering: $\frac{2}{nm}\sum_{i,j=1}^{n,m}\|a_i - b_j\|_2 - \frac{1}{n^2}\sum_{i,j=1}^{n}\|a_i - a_j\|_2 - \frac{1}{m^2}\sum_{i,j=1}^{m}\|b_i - b_j\|_2$

(f) Ward minimises the sum of squared differences within all clusters. It is a **variance-minimising** approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.

AGNES Clustering (cont)

Implementations:

- R: `hclust`, `cluster::agnes`
  - method = complete (hclust default), single, average (agnes default), ward.D, ward.D2(agnes ward), mcquitty (=WPGMA, agnes weighted), median (= WPGMC), centroid (= UPGMC), etc.
- Python: sklearn.cluster.AgglomerativeClustering
  - linkage = ward (default), complete, average, single

AGNES Clustering: Complete Linkage

In the *complete linkage method*, the dissimilarity between the groups of observations is defined as

$$d_{(hk)l} = \max\{d_{hl},\ d_{kl}\}, \quad l \neq h, k, \quad 1 \leq l \leq n. \tag{3}$$

The method computes all pairwise dissimilarities between the elements in a cluster and the elements in another cluster, and considers the *largest value* of these dissimilarities, i.e. (3), as the distance between the two clusters.

Complete Linkage (cont)

Example (Wikipedia: Complete-Linkage Clustering): Based on a JC69 genetic distance matrix computed from the 5S ribosomal RNA sequence alignment of five bacteria: Bacillus subtilis (A), Bacillus stearothermophilus (B), Lactobacillus viridescens (C), Acholeplasma modicum (D), and Micrococcus luteus (E).

	A	B	C	D	E
A	0	17	21	31	23
B	17	0	30	34	21
C	21	30	0	28	39
D	31	34	28	0	43
E	23	21	39	43	0

Complete Linkage (cont)

Perform complete-linkage hierarchical clustering.
Solution: Step 1: The (nonzero) smallest distance of the table is $d(A, B) = 17$. Therefore, we cluster A and B leading to the table below using the complete-linkage (3):

	A,B	C	D	E
A,B	0	**30**	**34**	**23**
C	**30**	0	28	39
D	**34**	28	0	43
E	**23**	39	43	0

A and B's branch length is $d(A, B) = 17$

Complete Linkage (cont)

Step 2: The smallest distance of the above table is $d((A, B), E) = 23$ and we cluster (A,B) and E leading to the table below using the complete-linkage:

	A,B,E	C	D
A,B,E	0	**39**	**43**
C	**39**	0	28
D	**43**	28	0

(A,B) and E's branch length is $d((A, B), E) - d(A, B) = 23 - 17 = 6.$

Complete Linkage (cont)

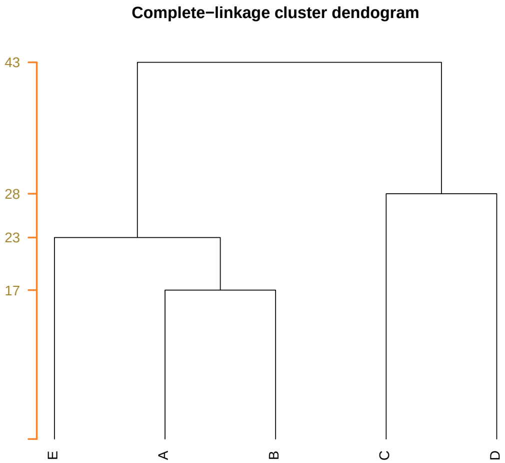
Step 3: The smallest distance of the above table is $d(C, D) = 28$ and we cluster C and D leading to the table below using the complete-linkage:

	A,B,E	C,D
A,B,E	0	**43**
C,D	**43**	0

Root r to (A,B,E)'s branch length is $d(r,(A,B,E)) = d((A,B,E),(C,D)) - d((A,B),E) = 43 - 23 = 20.$ Root r to (C,D)'s branch length is $d(r,(C,D)) = d((A,B,E),(C,D)) - d(C,D) = 43 - 28 = 15.$

Complete Linkage (cont)

These data allows us to construct the dendrogram below (compare to the Wikipedia):



Complete Linkage (cont)

Example (Final Exam May 2019, Q1(b)):
Table Q1(b) – i shows the data collected from five observations with three variables, x_1, x_2 and x_3.

Observation	x_1	x_2	x_3
A	6	3	4
B	3	2	1
C	7	5	5
D	3	1	2
E	2	3	2

Table Q1(b) – i

Complete Linkage (cont)

Example (Final Exam May 2019, Q1(b)) cont:
Table Q1(b) – ii shows parts of the Euclidean distance matrix computed from data collected in Table Q1(b) – i after applying standardisation.

Observation	A	B	C	D	E
A		2.3880		2.2835	
B				0.9082	
C	1.5496	3.6635		3.7430	
D					1.4251
E	2.2104		3.2358		

Table Q1(b) – ii

Complete Linkage (cont)

Example (Final Exam May 2019, Q1(b)) cont:

(i) Based on the information given, complete the distance matrix shown in Table Q1(b) – ii. (4 marks)

(ii) Group the observations using hierarchical clustering with **complete linkage**. Sketch the dendrogram formed by the hierarchical clustering. Group the observations into two groups and state the observations inside each group. (6 marks)

Remark: Answers not given, try to work out the answers.

AGNES Clustering: Single Linkage

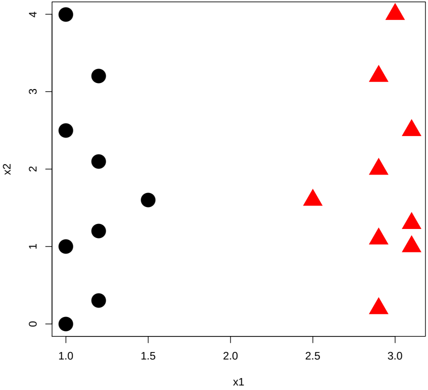
Single link method defines the dissimilarity between the (h, k) groups of observations as follows:

$$d_{(hk)l} = \min\{d_{hl}, d_{kl}\}, \quad l \neq h, k, \quad 1 \leq l \leq n \tag{4}$$

The method computes all pairwise dissimilarities between two clusters and considers the *smallest* of these similarities as a linkage criterion.

Single Linkage (cont)

An application of hierarchical clustering is the discovery of chain-like clusters such as the example shown below:



Single Linkage (cont)

Three-Subject Example:
The scores (1=worst to 10=best) of 8 children in three subjects were recorded.

Child	A	B	C	D	E	F	G	H
Maths	7	4	6	10	6	5	9	10
Music	5	7	8	10	2	9	6	9
History	8	8	6	9	2	4	8	9

Construct a distance matrix among the children. Perform hierarchical clustering process by using single linkage.

Single Linkage (cont)

Example (cont): **Computer Solution in Python**:

```
import pandas as pd, matplotlib.pyplot as plt
from scipy.spatial.distance import pdist
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial.distance import squareform
from latexutils import bmatrix

df = pd.DataFrame({’Child’ : list("ABCDEFGH"),
    ’Maths’ : [7,4,6,10,6,5,9,10], ’Music’ : [5,7,8,10,2,9,6,9 ],
    ’History’ : [8,8,6,9,2,4,8,9],
})
df = df.set_index(’Child’)
# other metric = ’cityblock’ aka Manhattan, ’hamming’, ’cosine’, ...
dists=pdist(df, metric=’euclidean’)
print(dists,"\n",bmatrix(squareform(dists)))
linkage_type ="single" # "single" "complete" "average"
linkage_matrix = linkage(dists, linkage_type)
print(bmatrix(linkage_matrix))
dendrogram(linkage_matrix, labels=df.index)
plt.title("Lecture {} Linkage Example".format(
    linkage_type.capitalize()))
plt.show()
```

Single Linkage (cont)

Solution:
The **distance matrix** among children A to H is

$$\begin{array}{c} A\\ B\\ C\\ D\\ E\\ F\\ G\\ H \end{array} \left[\begin{array}{cccccccc}
0. & & & & & & & \\
3.6056 & 0. & & & & & & \\
3.7417 & 3. & 0. & & & & & \\
5.9161 & 6.7823 & 5.3852 & 0. & & & & \\
6.7823 & 8.0623 & 7.2111 & 11.3578 & 0. & & & \\
6. & 4.5826 & 2.4495 & 7.1414 & 7.3485 & 0. & & \\
2.2361 & 5.099 & 4.1231 & 4.2426 & 7.8102 & 6.4031 & 0. & \\
5.099 & 6.4031 & 5.099 & 1. & 10.6771 & 7.0711 & 3.3166 & 0.
\end{array}\right]$$

Single Linkage (cont)

Note that $d(H, D) = 1$ is the minimum distance. So H and D are most similar and we “cluster” them and calculate:

- $d(\{H, D\}, A) = \min\{d(H, A), d(D, A)\} = 5.099$
- $d(\{H, D\}, B) = \min\{d(H, B), d(D, B)\} = 6.4031$
- $d(\{H, D\}, C) = \min\{d(H, C), d(D, C)\} = 5.099$
- $d(\{H, D\}, E) = \min\{d(H, E), d(D, E)\} = 10.6771$
- $d(\{H, D\}, F) = \min\{d(H, F), d(D, F)\} = 7.0711$
- $d(\{H, D\}, G) = \min\{d(H, G), d(D, G)\} = 3.3166$

Single Linkage (cont)

The step 1 distance matrix becomes

$$\begin{array}{c} A \\ B \\ C \\ D,H \\ E \\ F \\ G \end{array}\begin{bmatrix} 0. & & & & & & \\ 3.6056 & 0. & & & & & \\ 3.7417 & 3. & 0. & & & & \\ 5.099 & 6.4031 & 5.099 & 0. & & & \\ 6.7823 & 8.0623 & 7.2111 & 10.6771 & 0. & & \\ 6. & 4.5826 & 2.4495 & 7.0711 & 7.3485 & 0. & \\ 2.2361 & 5.099 & 4.1231 & 3.3166 & 7.8102 & 6.4031 & 0. \end{bmatrix}.$$

Single Linkage (cont)

The smallest distance in step 1 distance matrix is $d(A, G) = 2.2361$, therefore A and G will be grouped together and single linkage calculations will produce a step 2 distance matrix:

$$\begin{array}{c} A, G \\ B \\ C \\ D, H \\ E \\ F \end{array} \begin{bmatrix} 0. & & & & & \\ 3.6056 & 0. & & & & \\ 3.7417 & 3. & 0. & & & \\ 3.3166 & 6.4031 & 5.099 & 0. & & \\ 6.7823 & 8.0623 & 7.2111 & 10.6771 & 0. & \\ 6. & 4.5826 & 2.4495 & 7.0711 & 7.3485 & 0. \end{bmatrix}.$$

Single Linkage (cont)

The smallest distance in step 2 distance matrix is $d(C, F) = 2.4495$, therefore C and F will be grouped together and single linkage calculations will produce a step 3 distance matrix:

$$\begin{array}{c} A, G \\ B \\ C, F \\ D, H \\ E \end{array} \begin{bmatrix} 0. & & & & \\ 3.6056 & 0. & & & \\ 3.7417 & 3. & 0. & & \\ 3.3166 & 6.4031 & 5.099 & 0. & \\ 6.7823 & 8.0623 & 7.2111 & 10.6771 & 0. \end{bmatrix}.$$

Single Linkage (cont)

Note that $d(\{A, D, G, F\}, B) = \min\{3.6056, 4.5826\}$, $d(\{A, D, G, F\}, C) = \min\{3.7417, 2.4495\}$.
The smallest distance in step 3 distance matrix is $d(\{C, F\}, B) = 3$, therefore {C,F} and B will be grouped together and single linkage calculations will produce a step 4 distance matrix:

$$\begin{array}{c} A, G \\ B, C, F \\ D, H \\ E \end{array} \begin{bmatrix} 0. & & & \\ 3.6056 & 0. & & \\ 3.3166 & 5.099 & 0. & \\ 6.7823 & 7.2111 & 10.6771 & 0. \end{bmatrix}.$$

Single Linkage (cont)

The smallest distance in step 4 distance matrix is $d(\{A,G\},\{D,H\}) = 3.3166$, therefore {A,G} and {D,H} will be grouped together and single linkage calculations will produce a step 5 distance matrix:

$$\begin{array}{c} A,G,D,H \\ B,C,F \\ E \end{array} \begin{bmatrix} 0. & & \\ 3.6056 & 0. & \\ 6.7823 & 7.2111 & 0. \end{bmatrix}.$$

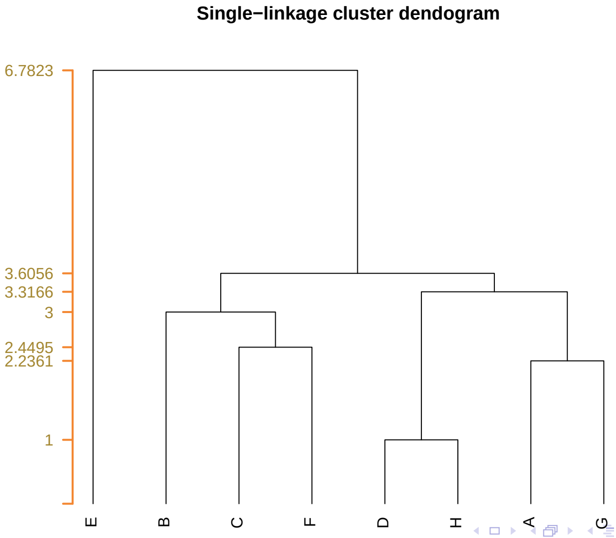
Single Linkage (cont)

The smallest distance in step 5 distance matrix is $d(\{A, G, D, H\}, \{B, C, F\}) = 3.6056$, therefore {A,G,D,H} and {B,C,F} will be grouped together and single linkage calculations will produce a step 6 distance matrix:

$$\begin{array}{c} A, G, D, H, B, C, F \\ E \end{array} \begin{bmatrix} 0. & \\ 6.7823 & 0. \end{bmatrix}.$$

Single Linkage (cont)

The dendrogram is



AGNES Clustering: Average Linkage

The *average-linkage method* or defines distance between groups as the average of the distances between all pairs of individuals in the two groups:

$$d_{(hk)l} = \frac{1}{N_{(hk)}N_l} \sum_{i \in S_{(hk)}} \sum_{j \in S_l} d_{ij}$$

where $N_{(hk)}$ and N_l are the number of items in clusters $S_{(hk)}$ and S_l, respectively.

Average Linkage (cont)

Three-Subject Example:
The scores (1=worst to 10=best) of 8 children in three subjects were recorded.

Child	A	B	C	D	E	F	G	H
Maths	7	4	6	10	6	5	9	10
Music	5	7	8	10	2	9	6	9
History	8	8	6	9	2	4	8	9

Construct a dendrogram using average linkage (`https://en.wikipedia.org/wiki/UPGMA`).

Average Linkage (cont)

Solution: We have construct the distance matrix in Single-Linkage Example.
The minimum distance is $d(D,H)=1$.
Step 1: Merge D,H; calculate new distances and find the min. dist. to be $d(A,G)=2.2361$:

$$\begin{aligned}
d(DH,A) &= \frac{1}{2\times 1}(5.9161+5.099)\approx 5.5076,\\
d(DH,B) &\approx 6.5927, \quad d(DH,C)=5.2421\\
d(DH,E) &= \frac{1}{2\times 1}(11.3578+10.6771)\approx 11.0175,\\
d(DH,F) &\approx 7.1063, \quad d(DH,G)\approx 3.7796.
\end{aligned}$$

Average Linkage (cont)

$$\begin{array}{c} A \\ B \\ C \\ D,H \\ E \\ F \\ G \end{array}\begin{bmatrix} 0. & & & & & & \\ 3.6056 & 0. & & & & & \\ 3.7417 & 3. & 0. & & & & \\ \boxed{5.5076} & \boxed{6.5927} & \boxed{5.2421} & 0. & & & \\ 6.7823 & 8.0623 & 7.2111 & \boxed{11.0175} & 0. & & \\ 6. & 4.5826 & 2.4495 & \boxed{7.1063} & 7.3485 & 0. & \\ 2.2361 & 5.099 & 4.1231 & \boxed{3.7796} & 7.8102 & 6.4031 & 0. \end{bmatrix}.$$

Average Linkage (cont)

Step 2: Merge A,G; calculate new distances and find the min. dist. is $d(C,F) = 2.4495$

$$d(AG,B) = \frac{1}{2\times 1}(3.6056 + 5.099) \approx 4.3523,$$
$$d(AG,C) \approx 6.5927,\ \ d(AG,E) \approx 7.1063,$$

$$\begin{aligned} d(AG,F) &\approx 3.7796, \\ d(AG,DH) &= \frac{1}{2\times 2}(d_{AD} + d_{AH} + d_{GD} + d_{GH}) \\ &= \frac{d(DH,A) + d(DH,G)}{2} \\ &= \frac{5.5076 + 3.7796}{2} \approx 4.6436 \end{aligned}$$

Average Linkage (cont)

$$\begin{array}{c} A,G \\ B \\ C \\ D,H \\ E \\ F \end{array} \begin{bmatrix} 0. & & & & & \\ \boxed{4.3523} & 0. & & & & \\ \boxed{3.9324} & 3. & 0. & & & \\ \boxed{4.6436} & 6.5927 & 5.2421 & 0. & & \\ \boxed{7.2963} & 8.0623 & 7.2111 & 11.0175 & 0. & \\ \boxed{6.2016} & 4.5826 & 2.4495 & 7.1063 & 7.3485 & 0. \end{bmatrix}.$$

Average Linkage (cont)

Step 3: Merge C,F; calculate new distances and find the min. dist. is $d(B, CF) = 3.7913$:

$$\begin{aligned} d(CF, AG) &= \frac{1}{2\times 2}(d_{CA} + d_{CG} + d_{FA} + d_{FG}) \\ &= \frac{d(AG, C) + d(AG, F)}{2} \approx 5.0670 \end{aligned}$$

$$\begin{aligned} d(CF, DH) &= \frac{1}{2\times 2}(d_{CD} + d_{CH} + d_{FD} + d_{FH}) \\ &= \frac{d(DH, C) + d(DH, F)}{2} \approx 6.1742 \end{aligned}$$

$$d(CF, B) \approx 3.7913, \quad d(CF, E) \approx 7.2798$$

Average Linkage (cont)

$$\begin{array}{c} A, G \\ B \\ C, F \\ D, H \\ E \end{array} \begin{bmatrix} 0. & & & & \\ 4.3523 & 0. & & & \\ \boxed{5.0670} & \boxed{3.7913} & 0. & & \\ 4.6436 & 6.5927 & \boxed{6.1742} & 0. & \\ 7.2963 & 8.0623 & \boxed{7.2798} & 11.0175 & 0. \end{bmatrix}.$$

Average Linkage (cont)

Step 4: Merge B,CF; calculate new distances and find the min. dist. to be $d(AG, DH) = 4.6436$:

$$\begin{aligned}
d(BCF, AG) &= \frac{1}{3 \times 2}(4d(CF, AG) + 2d(B, AG)) \\
&= \frac{2 \times 5.0670 + 1 \times 4.3523}{3} \approx 4.8288 \\
d(BCF, DH) &= \frac{1}{3 \times 2}(4d(CF, DH) + 2d(B, DH)) \\
&= \frac{2 \times 6.1742 + 6.5927}{3} \approx 6.3137 \\
d(BCF, E) &= d_{BE} + d_{CE} + d_{FE} \\
&= \frac{1}{3 \times 1}(d(B, E) + 2d(CF, E)) \\
&= \frac{8.0623 + 2 \times 7.2798}{3} \approx 7.5406
\end{aligned}$$

Average Linkage (cont)

$$\begin{array}{c}A,G\\B,CF\\D,H\\E\end{array}\begin{bmatrix}0. & & & \\ \boxed{4.8288} & 0. & & \\ 4.6436 & \boxed{6.3137} & 0. & \\ 7.2963 & \boxed{7.5406} & 11.0175 & 0.\end{bmatrix}.$$

Average Linkage (cont)

Step 5: Merge AG,DH; calculate new distances and find the min. dist. to be $d(AGDH, BCF) = 5.5713$

$$\begin{aligned} d(AGDH, BCF) &= \frac{1}{4 \times 3}(6d(AG, BCF) + 6d(BCF, DH)) \\ &= \frac{4.8288 + 6.3137}{2} \approx 5.5713 \end{aligned}$$

$$\begin{aligned} d(AGDH, E) &= \frac{1}{4 \times 1}(2d(AG, E) + 2d(DH, E)) \\ &= \frac{7.2963 + 11.0175}{2} \approx 9.1569 \end{aligned}$$

Average Linkage (cont)

$$\begin{array}{c} AG, DH \\ B, CF \\ E \end{array} \begin{bmatrix} 0. & & \\ \boxed{5.5713} & 0. & \\ \boxed{9.1569} & 7.5406 & 0. \end{bmatrix}.$$

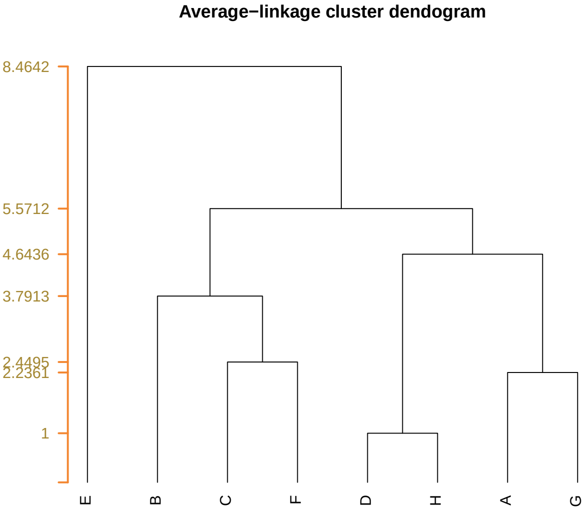
Step 6: Merge AGDH, BCF; calculate new distances and find the min. dist. to be $d(,) =$

$$\begin{aligned} d(AGDHBCF, E) &= \frac{1}{7\times 1}(4d(AGDH, E) + 3d(BCF, E)) \\ &= \frac{4\times 9.1569 + 3\times 7.5406}{7} \approx 8.4642 \end{aligned}$$

$$\begin{array}{c} AGDH, BCF \\ E \end{array} \begin{bmatrix} 0. & \\ \boxed{8.4642} & 0. \end{bmatrix}.$$

Average Linkage (cont)

The dendrogram is



AGNES Clustering (cont)

The benefits of ‘dendrogram’:

- Show how to merge clusters hierarchically (the points are in p-dimensional space which cannot be visualised)
- Decompose data objects into a multi-level nested partitioning (a tree of clusters)
- A clustering of the data objects: cutting the dendrogram at the desired level — Each connected component forms a cluster: R’s cutree(hc, k=#groups)

DIANA Clustering

DIANA = DIvisive ANAlysis

- Initially, all objects are in one cluster
- Step-by-step splitting clusters until each cluster contains only one object

Challenges:

- Hard to choose merge/split points
  - Never undo merging/splitting
  - Merging/splitting decisions are critical
- High complexity $O(n^2)$
- Integrating hierarchical clustering with other techniques — BIRCH, CURE, CHAMELEON, ROCK

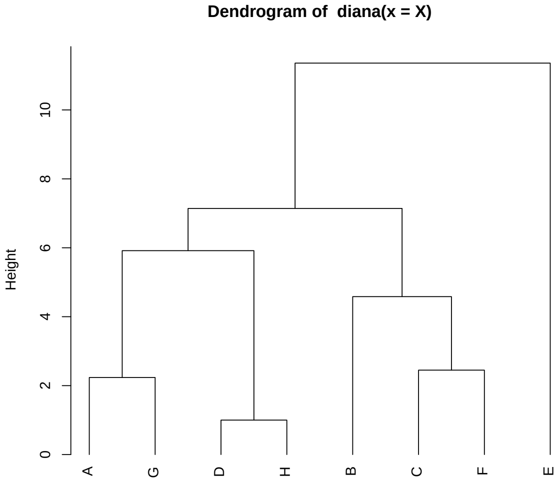
DIANA Clustering (cont)

Three-Subject Example:
R-script:

```
X = data.frame(
    Maths=c(7 ,4 ,6 ,10 ,6 ,5 ,9 ,10),
    Music=c(5 ,7 ,8 ,10 ,2 ,9 ,6 ,9 ),
    Hist=c(8 ,8 ,6 ,9 ,2 ,4 ,8 ,9),
    row.names=LETTERS[1:8]
)
library(cluster)
hc = diana(X)
pltree(hc, hang=-1)
```

DIANA Clustering (cont)

Three-Subject Example (cont): The dendrogram is



Outline

1. Theory
2. Partitioning Methods
   - k-Means, PAM, ...
   - GMM (and SOM)
3. Hierarchical Methods: AGNES, DIANA, ...
4. Density-Based Methods
5. Grid-Based Methods
6. Kernel and Spectral Based Methods

Drawbacks of Earlier Methods

The **drawbacks** of Distance-based Methods (partitioning methods and hierarchical methods):

- Hard to find clusters with irregular shapes
- Hard to specify the number of clusters
- Heuristic: a cluster must be dense

How to Find Irregular Clusters?

- Divide the whole space into many small areas
    - The density of an area can be estimated
    - Areas may or may not be exclusive
    - A dense area is likely in a cluster
- Start from a dense area, traverse connected dense areas and discover clusters in irregular shape

Density-Based Methods

The rationale of density-based clustering is that a cluster is composed of well-connected dense region, while objects in sparse areas are removed as noises.

The advantage of density-based clustering is that it can filter out noise and find clusters of arbitrary shapes (as long as they are composed of connected dense regions).

Density-Based Methods (cont)

- Density-Based Spatial Clustering of Applications with Noise (DBSCAN): Works by expanding clusters to their dense neighbourhood. See https://en.wikipedia.org/wiki/DBSCAN
- Ordering Points To Identify the Clustering Structure (OPTICS): See https://dl.acm.org/doi/pdf/10.1145/304181.304187
- Density-based Clustering (DENCLUE): See https://link.springer.com/chapter/10.1007/978-3-540-74825-0_7

Outline

1. Theory
2. Partitioning Methods
   - k-Means, PAM, ...
   - GMM (and SOM)
3. Hierarchical Methods: AGNES, DIANA, ...
4. Density-Based Methods
5. Grid-Based Methods
6. Kernel and Spectral Based Methods

Grid-Based Methods

Ideas:

- Using multi-resolution grid data structures
- Using dense grid cells to form clusters

Interesting methods:

- Clustering in Quest (CLIQUE): Partition the data space and find the number of points that lie inside each cell of the partition.
- STING (STatistical INformation Grid approach): The spatial area is divided into rectangular cells.
- WaveCluster: A multi-resolution clustering approach using wavelet method

Outline

1. Theory
2. Partitioning Methods
   - k-Means, PAM, ...
   - GMM (and SOM)
3. Hierarchical Methods: AGNES, DIANA, ...
4. Density-Based Methods
5. Grid-Based Methods
6. Kernel and Spectral Based Methods

Kernel Based Methods

Generalised “distance” in partitioning methods to “kernel” functions.

Kernel Clustering Methods

- Kernel K-Means
- Kernel Self-Organising Maps (SOM)
- Kernel Neural Gas
- Kernel Fuzzy C-Means
- Kernel Probabilistic C-Means
- One class SVMs and Support Vector Clustering

Spectral Based Methods

Spectral clustering takes the relative position of data points into account. The algorithm can be broken down into 4 basic steps.

1. Construct a similarity graph;
2. Determine the adjacency matrix W, degree matrix D and the Laplacian matrix L;
3. Compute the eigenvectors of the matrix L;
4. Using the second smallest eigenvector as input, train a k-means model and use it to classify the data.

Spectral Based Methods (cont)

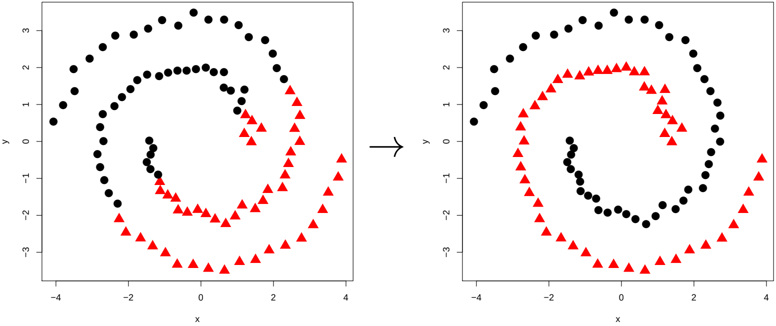
Software Implementations:

- R: `kernlab::specc`, `kknn::specClust`
- Python:

```
sklearn.cluster.SpectralClustering(n_clusters=8, *,
  eigen_solver=None, n_components=None,
  random_state=None, n_init=10, gamma=1.0,
  affinity=’rbf’, n_neighbors=10, eigen_tol=0.0,
  assign_labels=’kmeans’, degree=3, coef0=1,
  kernel_params=None, n_jobs=None, verbose=False)
```

Spectral Based Methods (cont)

k-Means vs spectral clustering



Spectral Based Methods (cont)

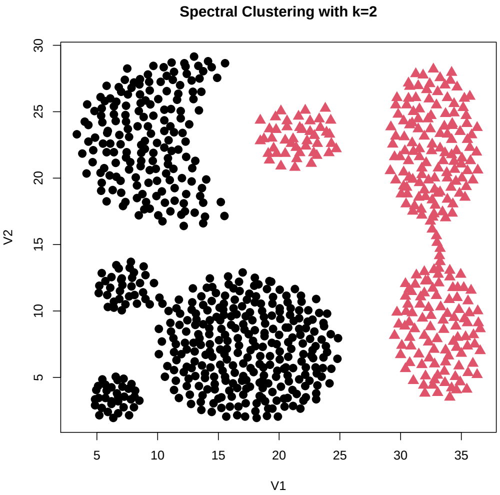
The diagrams are generated using the R script below.

```
library(mlbench)
set.seed(111)
obj = mlbench.spirals(100,1,0.025)
my.data = 4 * obj$x
colnames(my.data) = c('x','y')
km = kmeans(my.data, 2)
pdf("clust_kmean.pdf")
plot(my.data, col=km$cluster, pch=15+km$cluster

library(kernlab)
sc = specc(my.data, centers=2)
pdf("clust_spectral.pdf")
plot(my.data, col=sc, pch=15+sc, cex=2)
```

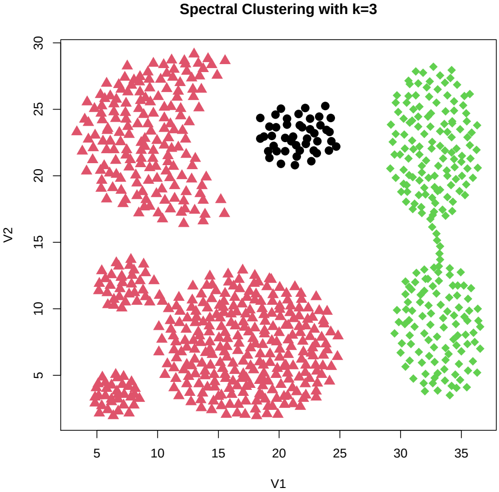
k-Means Clustering (cont)

Multiple Clusters Example (vs k-means):



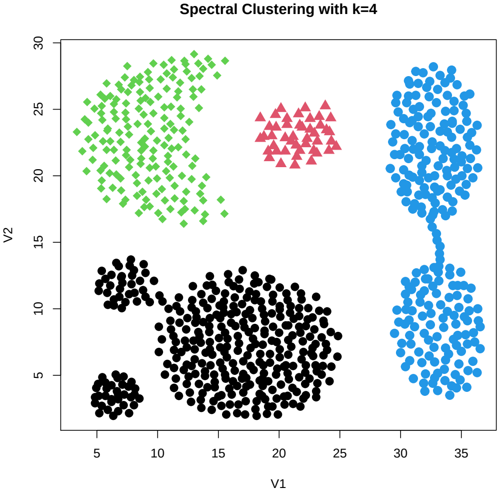
k-Means Clustering (cont)

Multiple Clusters Example (cont):



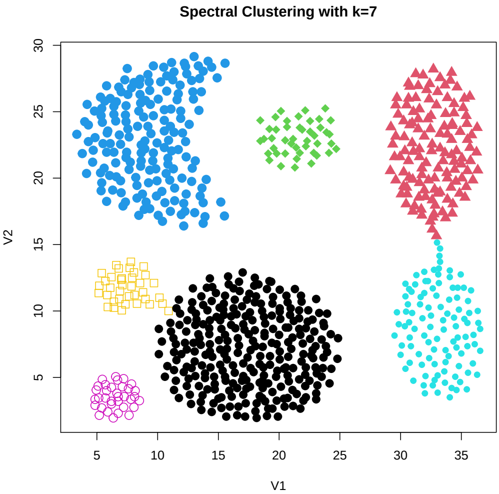
k-Means Clustering (cont)

Multiple Clusters Example (cont):



k-Means Clustering (cont)

Multiple Clusters Example (cont):


Spectral Clustering with k=7
V2
V1