LECTURE 8: CLUSTERING ALGORITHMS

STAT 545: Intro. to Computational Statistics

Vinayak Rao

Purdue University

September 11, 2019

Clustering

Given a large dataset, group data points into 'clusters'.

Data points in the same cluster are similar in some sense.

E.g. cluster students scores (to decide grade)

Clustering

Given a large dataset, group data points into 'clusters'.

Data points in the same cluster are similar in some sense.

E.g. cluster students scores (to decide grade)

Applications:

Compression/feature-extraction/exploration/visualization

- simpler representation of complex data

Image segmentation, community detectn, co-expressed genes

We are given N data vectors $(\mathbf{x}_1, \ldots, \mathbf{x}_N)$ in $\Re^d$.
Let c_i be the cluster assignment of observation x_i:

$$c_i \in \{1 \cdots K\}$$

Equivalently, we can use one-hot (or 1-of-K) encoding:

$$r_{ic} = \begin{cases} 1, & \text{if } c_i = c \\ 0, & \text{otherwise} \end{cases}$$

$$\text{E.g.} \qquad c_i = 3 \quad \Longleftrightarrow \quad \mathbf{r}_i = (0, 0, 1, 0, 0, ..., 0)$$

Observe: $r_{ic} \geq 0$ and $\sum_{c=1}^{K} r_{ic} = 1$ just like a probability vector.

However, r_{ic} is binary: we will relax this in later lectures.

Cluster parameters

Associate cluster i with parameter $\theta_i \in \Re^d$ (cluster prototype).

Write $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_K\}$, $\mathsf{C} = \{c_1, \ldots, c_N\}$ (or $\mathsf{R} = \{\mathbf{r}, \ldots, \mathbf{r}_N\}$).

Associate cluster i with parameter $\theta_i \in \Re^d$ (cluster prototype).

Write $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_K\}$, $\mathsf{C} = \{c_1, \ldots, c_N\}$ (or $\mathsf{R} = \{\mathsf{r}, \ldots, \mathsf{r}_N\}$).

Problem: Given data $(\mathsf{x}_1, \ldots, \mathsf{x}_N)$ find $\boldsymbol{\theta}$ and C.

Define a loss-function $L(\boldsymbol{\theta}, \mathsf{C})$, and minimize it.

Cluster parameters

Associate cluster i with parameter $\theta_i \in \Re^d$ (cluster prototype).

Write $\boldsymbol{\theta} = \{\theta_1, \ldots, \theta_K\}$, $\mathsf{C} = \{c_1, \ldots, c_N\}$ (or $\mathsf{R} = \{\mathsf{r}, \ldots, \mathsf{r}_N\}$).

Problem: Given data $(\mathsf{x}_1, \ldots, \mathsf{x}_N)$ find $\boldsymbol{\theta}$ and C.

Define a loss-function $L(\boldsymbol{\theta}, \mathsf{C})$, and minimize it.

Start by defining a distance (or similarity measure) $d(\mathsf{x}, \boldsymbol{\theta})$:

$$d(\mathsf{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d} (x_j - \theta_j)^2 \qquad \text{Squared Euclidean or } L_2 \text{ dist.}$$

$$d(\mathsf{x}, \boldsymbol{\theta}) = \sum_{j=1}^{d} |x_j - \theta_j| \qquad L_1 \text{ distance}$$

Clustering loss function

We want all members of a cluster to be close to the prototype.

$$\sum_{i \text{ s.t. } c_i = c} d(\mathbf{x}_i, \boldsymbol{\theta}_c) = \sum_{i=1}^{N} r_{ic} d(\mathbf{x}_i, \boldsymbol{\theta}_c) \text{ should be small for each } c.$$

Overall loss function:

$$\begin{aligned} L(\boldsymbol{\theta}, \mathbf{C}) &= \sum_{c=1}^{K} \sum_{i s.t. c_i = c1} d(\mathbf{x}_i, \boldsymbol{\theta}_c) \\ &= \sum_{c=1}^{K} \sum_{i=1}^{N} r_{ic} d(\mathbf{x}_i, \boldsymbol{\theta}_c) \end{aligned}$$

Optimize over both:

- cluster assignments (discrete)
- cluster parameters (continuous)

Minimizing $L(\boldsymbol{\theta}, \mathsf{C})$ is hard ($O(N^{DK+1})$).

K-MEANS

Minimizing $L(\boldsymbol{\theta}, \mathsf{C})$ is hard ($O(N^{DK+1})$).

Instead, use heuristic greedy (local-search) algorithms.
When $d(\cdot,\cdot)$ is Euclidean, the most popular is Lloyd's algorithm.

Minimizing $L(\boldsymbol{\theta}, \mathsf{C})$ is hard ($O(N^{DK+1})$).

Instead, use heuristic greedy (local-search) algorithms.
When $d(\cdot, \cdot)$ is Euclidean, the most popular is Lloyd's algorithm.

If we had the cluster parameters $\boldsymbol{\theta}^*$, can we solve for C?

$$\mathsf{C}_{opt} = \operatorname{argmin} L(\boldsymbol{\theta}^*, \mathsf{C})$$

If we had the cluster assignments C^*, can we solve for $\boldsymbol{\theta}$?

$$\boldsymbol{\theta}_{opt} = \operatorname{argmin} L(\boldsymbol{\theta}, \mathsf{C}^*)$$

K-means

Start with an initialialization of the parameters, call it $\boldsymbol{\theta}_0$.
Assign observations to nearest clusters, giving R_0.

Repeat for i in 1 to N:

- Recalculate cluster means, $\boldsymbol{\theta}_i$
- Recalculate cluster assignments, R_i

Coordinate-descent.

Start with an initialialization of the parameters, call it $\boldsymbol{\theta}_0$.
Assign observations to nearest clusters, giving R_0.

Repeat for i in 1 to N:

- Recalculate cluster means, $\boldsymbol{\theta}_i$
- Recalculate cluster assignments, R_i

Coordinate-descent.

Resulting algorithm has complexity $O(INKD)$

[demo]

Questions

Does this algorithm converge to a global minimum?

Does it converge at all?

What is the convergence criteria?

Limitations

Local optima: Sensitive to initialization.
Solution: Run many times and pick the best clustering.

Empty clusters.
Solution: discard them, or use heuristics to assign observations to them

Choosing K.
Solution: search over a set of K's, penalizing larger values.

Requires circular clusters.
Solution: use some other method

Variations to k-means

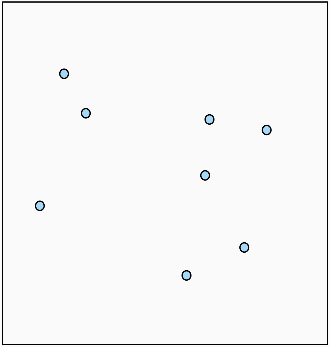
Modify distance functions.

L_1 distance: k-medians

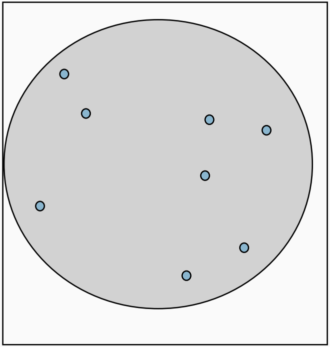
Modify the algorithm.

L_1 distance: k-medoids (exemplar-based)

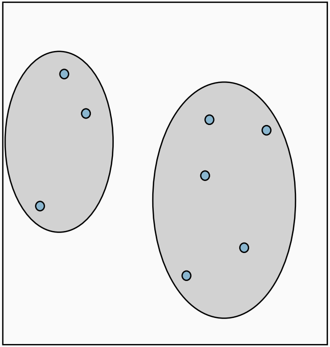
k-means is a partitioning algorithm that assigns each observation to a unique cluster.
Often there is no clear best clustering.



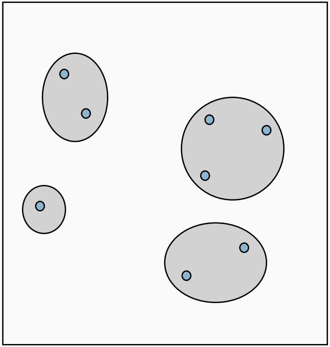
k-means is a partitioning algorithm that assigns each observation to a unique cluster.
Often there is no clear best clustering.



k-means is a partitioning algorithm that assigns each observation to a unique cluster.
Often there is no clear best clustering.

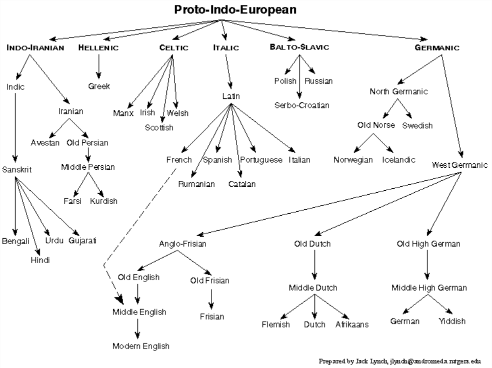


k-means is a partitioning algorithm that assigns each observation to a unique cluster.
Often there is no clear best clustering.



Hierarchical clustering

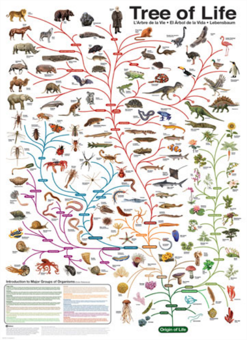
k-means is a partitioning algorithm that assigns each observation to a unique cluster.
Often is is natural to view data as having a hierarchical structure



Hierarchical clustering

k-means is a partitioning algorithm that assigns each observation to a unique cluster.

Often is is natural to view data as having a hierarchical structure



Two approaches:

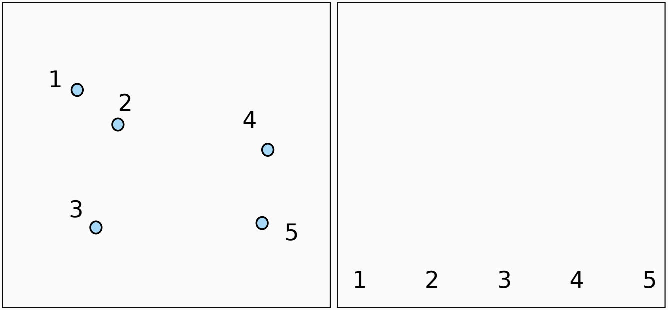
Top-down (divisive) clustering:

- Initialize all observations into a single cluster, and divide clusters sequentially.

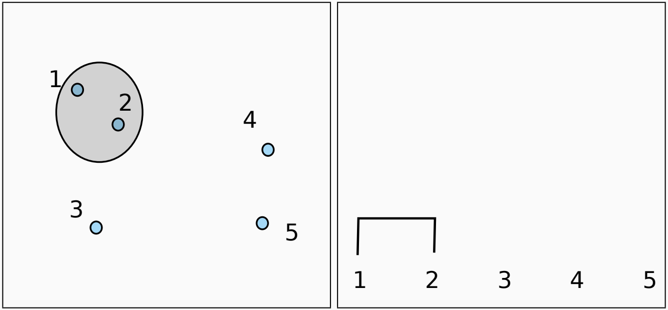
Bottom-up (agglomerative) clustering:

- Initialize each observation in its own cluster, and merge clusters sequentially.
- More flexible, and more common.

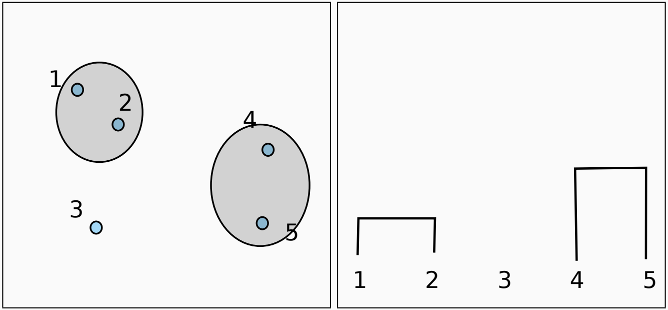
Agglomerative clustering

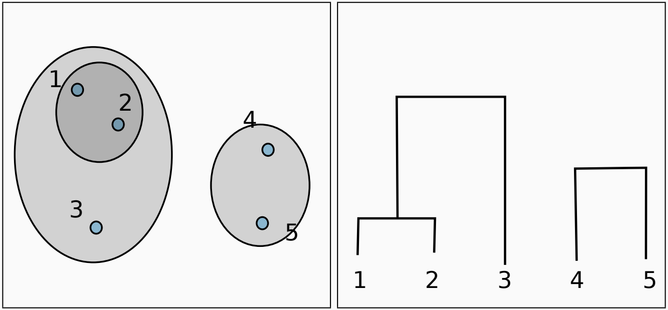


Agglomerative clustering

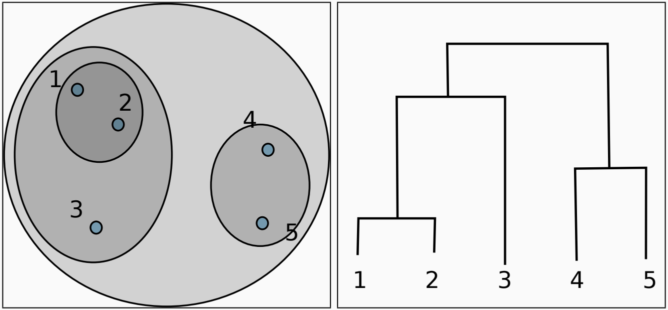


Agglomerative clustering

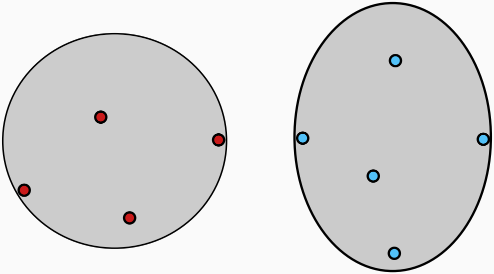



1
2
3
4
5
1
2
3
4
5

Agglomerative clustering



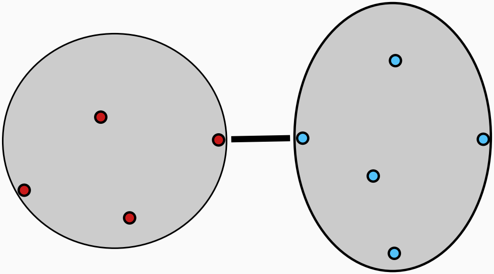
Agglomerative clustering



Pick a distance function (e.g. Euclidean).

Pick a linkage criterion defining distance between two clusters:

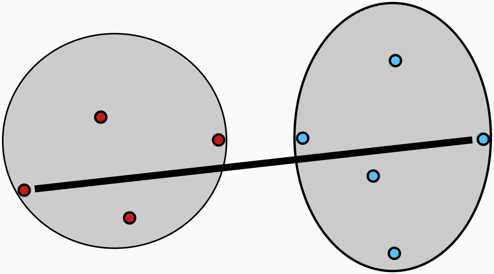
Agglomerative clustering



Pick a distance function (e.g. Euclidean).

Pick a linkage criterion defining distance between two clusters:

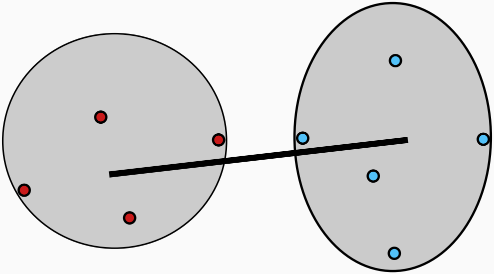
- Single linkage: $d(A, B) = \min_{x \in A, y \in B} d(x, y)$.



Pick a distance function (e.g. Euclidean).

Pick a linkage criterion defining distance between two clusters:

- Single linkage: $d(A, B) = \min_{x \in A, y \in B} d(x, y)$.
- Complete linkage: $d(A, B) = \max_{x \in A, y \in B} d(x, y)$.

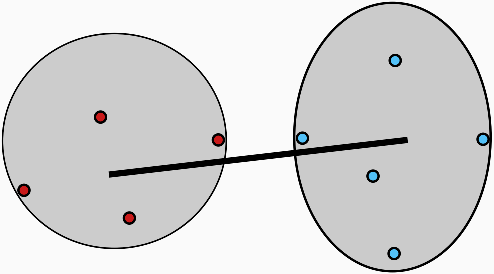


Pick a distance function (e.g. Euclidean).

Pick a linkage criterion defining distance between two clusters:

- Single linkage: $d(A, B) = \min_{x \in A, y \in B} d(x, y)$.
- Complete linkage: $d(A, B) = \max_{x \in A, y \in B} d(x, y)$.
- Centroid linkage: $d(A, B) = d(C_A, C_B)$ (C_A: centroid of A).

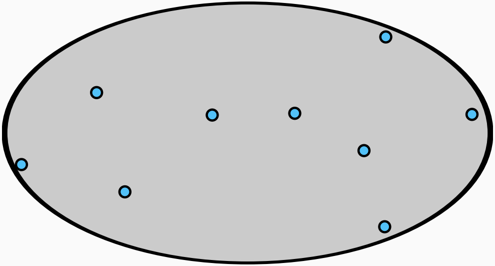
Agglomerative clustering



Pick a distance function (e.g. Euclidean).

Pick a linkage criterion defining distance between two clusters:

- Single linkage: $d(A, B) = \min_{x \in A, y \in B} d(x, y)$.
- Complete linkage: $d(A, B) = \max_{x \in A, y \in B} d(x, y)$.
- Centroid linkage: $d(A, B) = d(C_A, C_B)$ (C_A: centroid of A).
- Average linkage: $d(A, B) = \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y)$.



Pick a distance function (e.g. Euclidean).

Pick a linkage criterion defining distance between two clusters:

- Single linkage: $d(A, B) = \min_{x \in A, y \in B} d(x, y)$.
- Complete linkage: $d(A, B) = \max_{x \in A, y \in B} d(x, y)$.
- Centroid linkage: $d(A, B) = d(C_A, C_B)$ (C_A: centroid of A).
- Average linkage: $d(A, B) = \frac{1}{|A||B|} \sum_{x \in A, y \in B} d(x, y)$.