Clustering Algorithms

- Optimization based clsutering
- Hierarchical clustering
- Density based and mode-seeking clustering
- Graph based clustering

K-Means

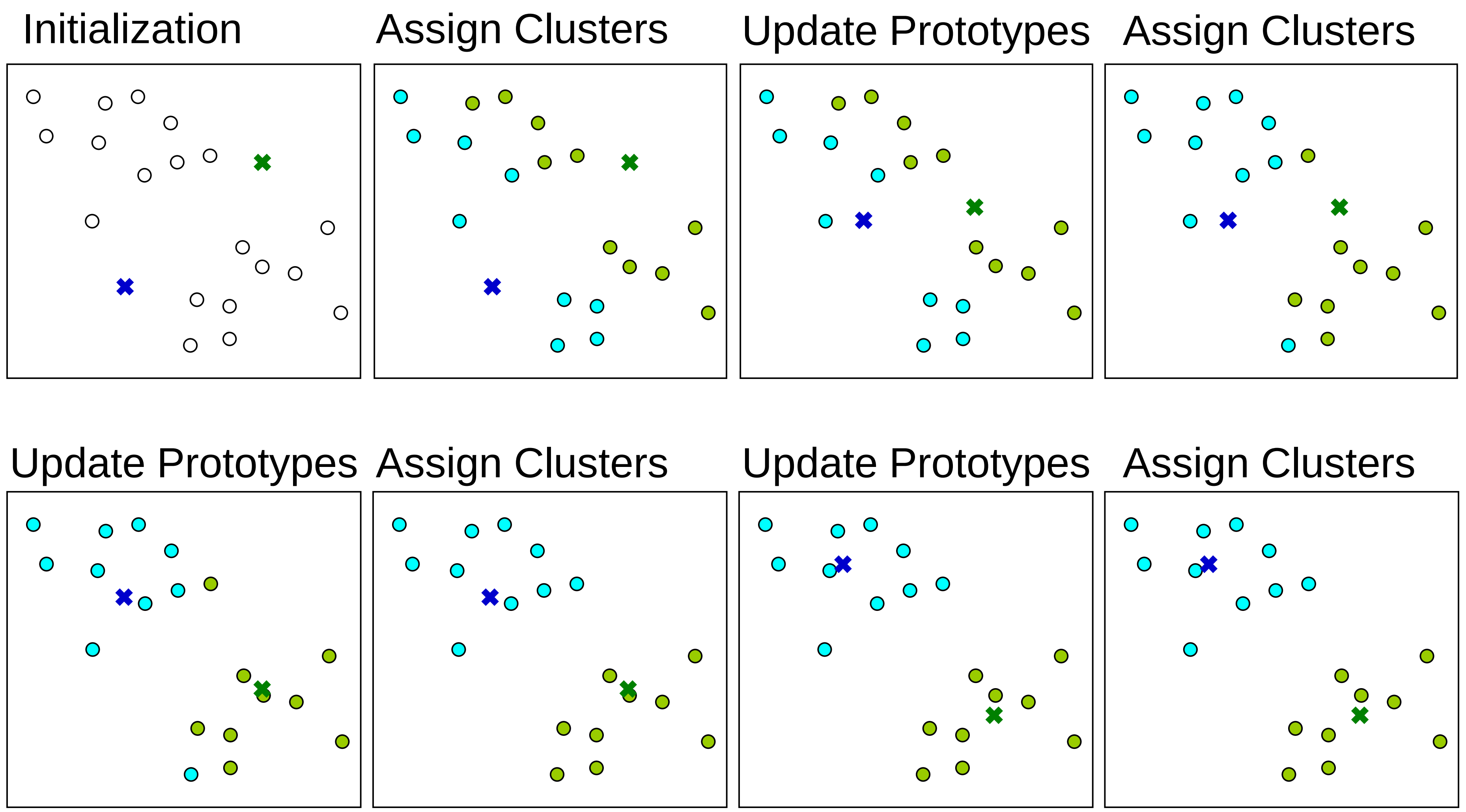
Each sample (***x***) belongs to exactly one cluster.

Algorithm (Lloyd's k-means):

- Initialize the ***k*** prototypes (cluster representatives)
- Repeat until "stopping criteria"
  - Assign each ***x*** to the cluster of the closest prototype
  - Recalculate each prototype as the **mean** of all ***x*** belonging to that cluster

- Common stopping criterion: no change of prototypes (or cluster assignments) between iterations.
- Alternative initialization: Start with random cluster assignments of all the samples.
- An example of **competitive learning** (the clusters "compete" for samples), which is of greedy nature.

k-Means Example (2-D)



No change ➔ Stop!

K-means as Optimization

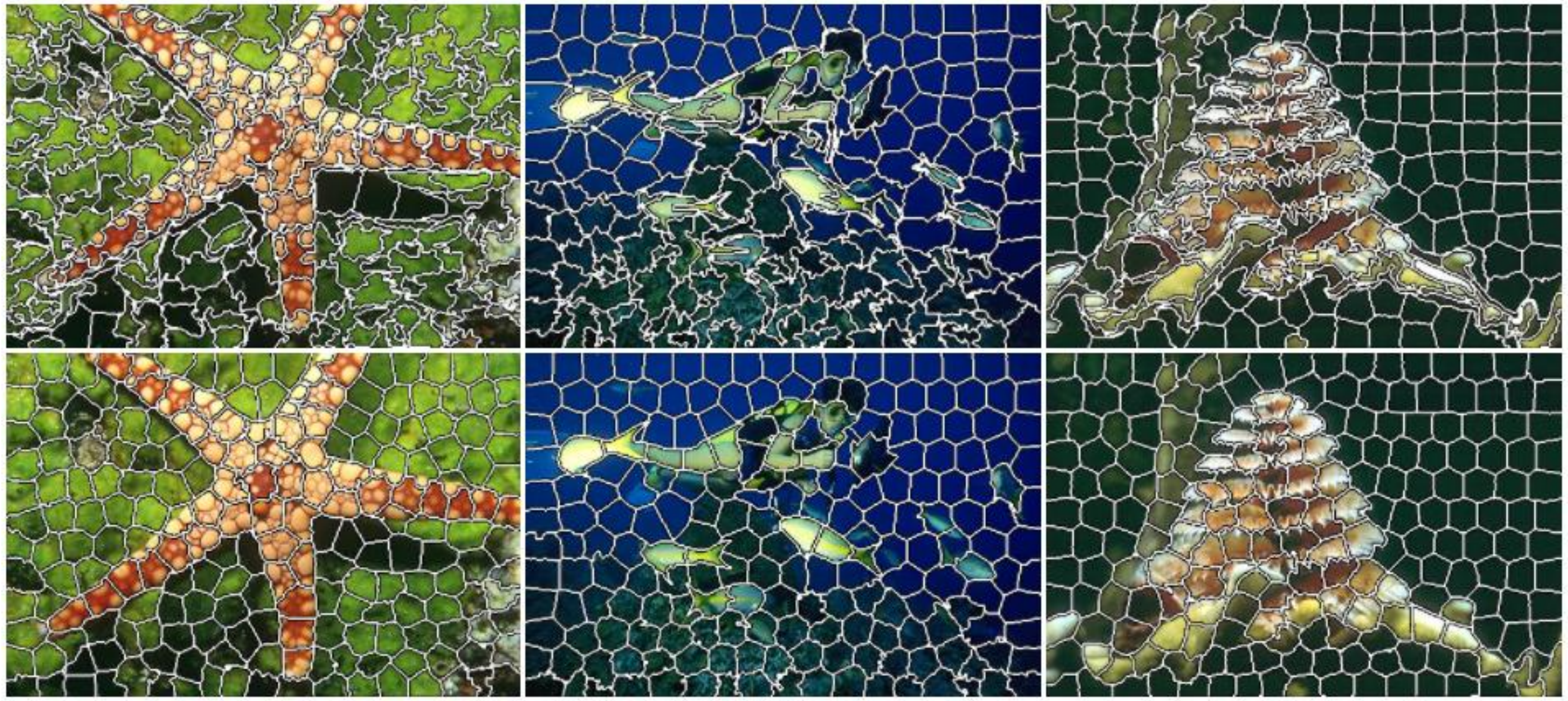
- K-means algorithm is an example of alternating optimization.
- The cost function:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{k} u_{ij}\, d_{ij}^{2}$$

  - $u_{ij} \in \{0, 1\}$: Indicating whether sample $\#i$ is assigned to cluster $\#j$.
  - d_{ij}: Distance between sample $\#i$ and cluster prototype $\#j$.
- u_{ij} and d_{ij} are optimized in alternating steps:
  - u_{ij}: When a sample is assigned to the cluster with the smallest distance.
  - d_{ij}: When the prototype is set to the mean point of samples in the cluster.

K-Means for Image Segmentation

- The widely used SLIC superpixel algorithm is based on k-means, with k being the number of superpixels.
- Each pixel is defined by five values: three for color (usually RGB) and two for location.



Drawbacks and Challenges of K-means

- Selection of k: Use cluster validity measures or allow cluster merging/splitting
- Sensitivity to initialization: More systematic initial prototype selection
- Local minimums of cost function:
- Sensitivity to outliers: Use robust estimators or automatic outlier detection
- Natural preference of similarly sized spherical clusters: Use different prototype representations / cluster-sample distance measures
- Extremely large datasets: Online or approximate methods

Fuzzy Partition of Data

- Idea: Let "ambiguous" samples to have partial memberships in multiple clusters. (This means that the clusters become "fuzzy".)
- Probabilistic partition: The total membership of a sample in all the clusters sum to one.
- Cost function and constraints of **fuzzy k-means**:

$$J = \sum_{i=1}^{N} \sum_{j=1}^{k} u_{ij}^{q} \, d_{ij}^{2} \qquad \sum_{i} u_{ij} > 0, \ \forall j \qquad 0 \le u_{ij} \le 1 \qquad \sum_{j=1}^{m} u_{ij} = 1, \ \forall i$$

$q > 1$ is the "fuzzification factor"

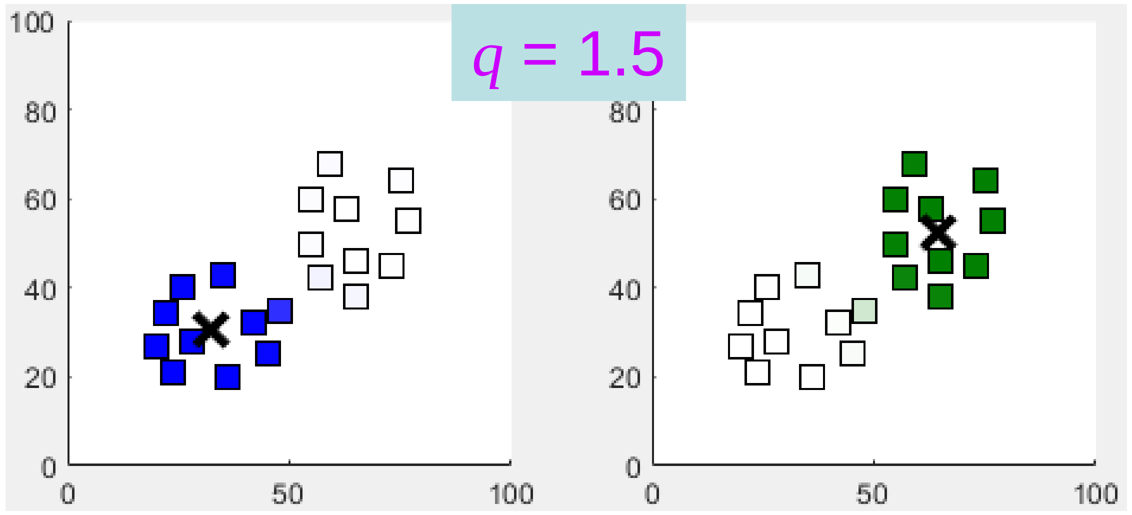
- Update equations:

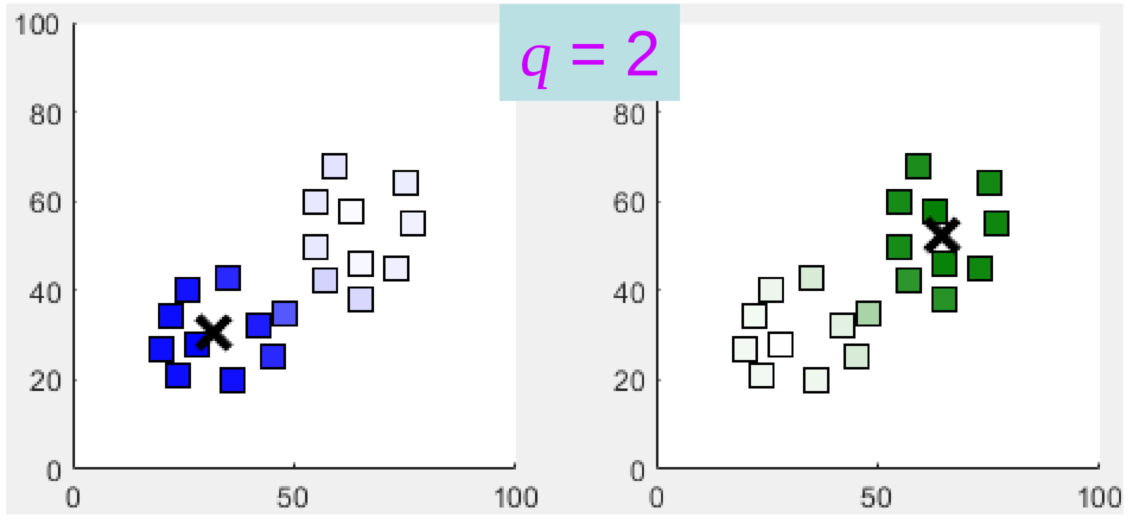
Memberships: $$u_{ij} = \left[\sum_{p=1}^{k} \left(\frac{d_{ij}}{d_{ip}} \right)^{\frac{2}{q-1}} \right]^{-1}$$

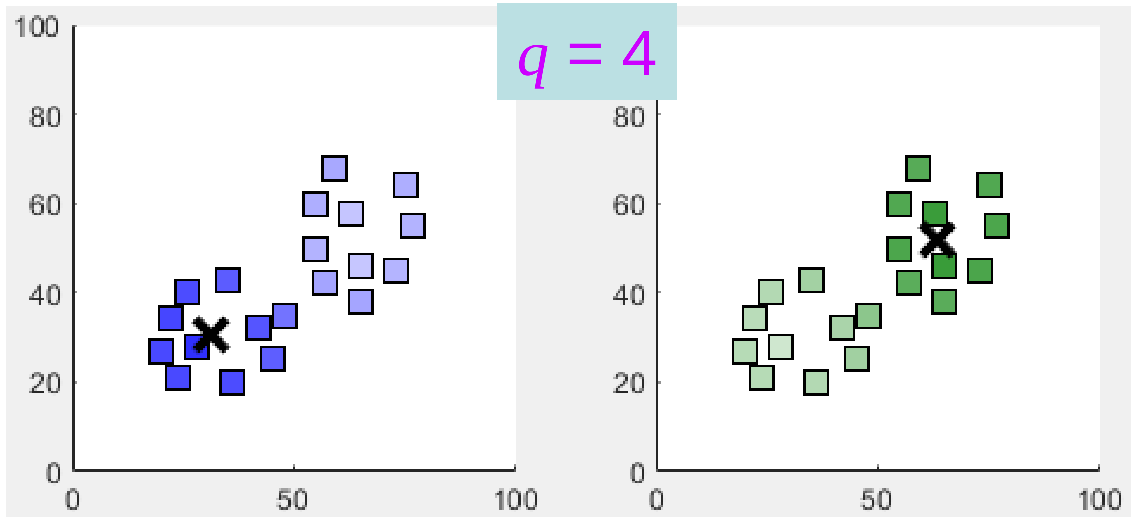
Prototypes: $$\boldsymbol{v}_j = \frac{\sum_{i=1}^{N} (u_{ij})^q \, \boldsymbol{x}_i}{\sum_{i=1}^{N} (u_{ij})^q}$$

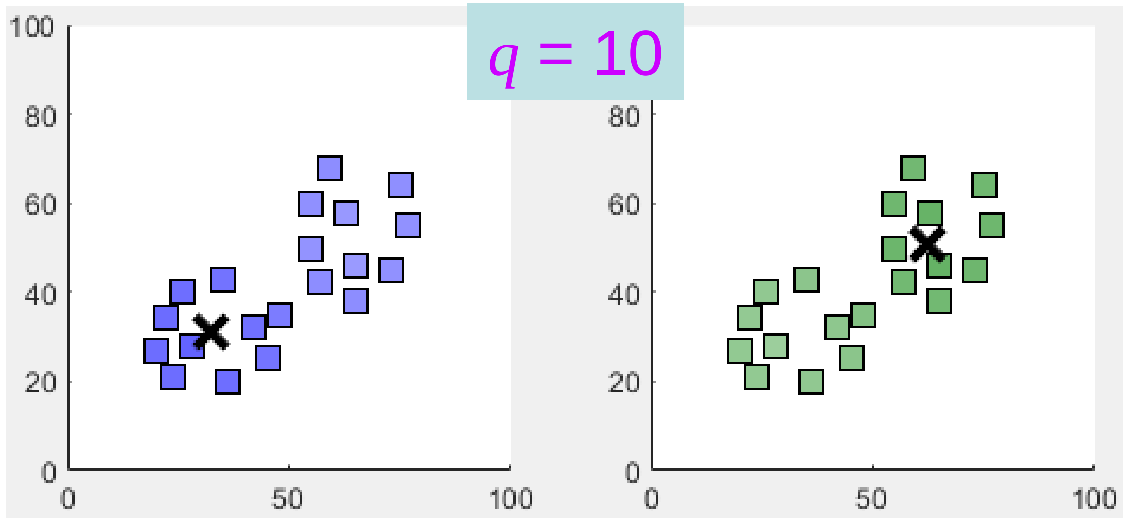
FKM Example

Memberships of the two clusters at convergence





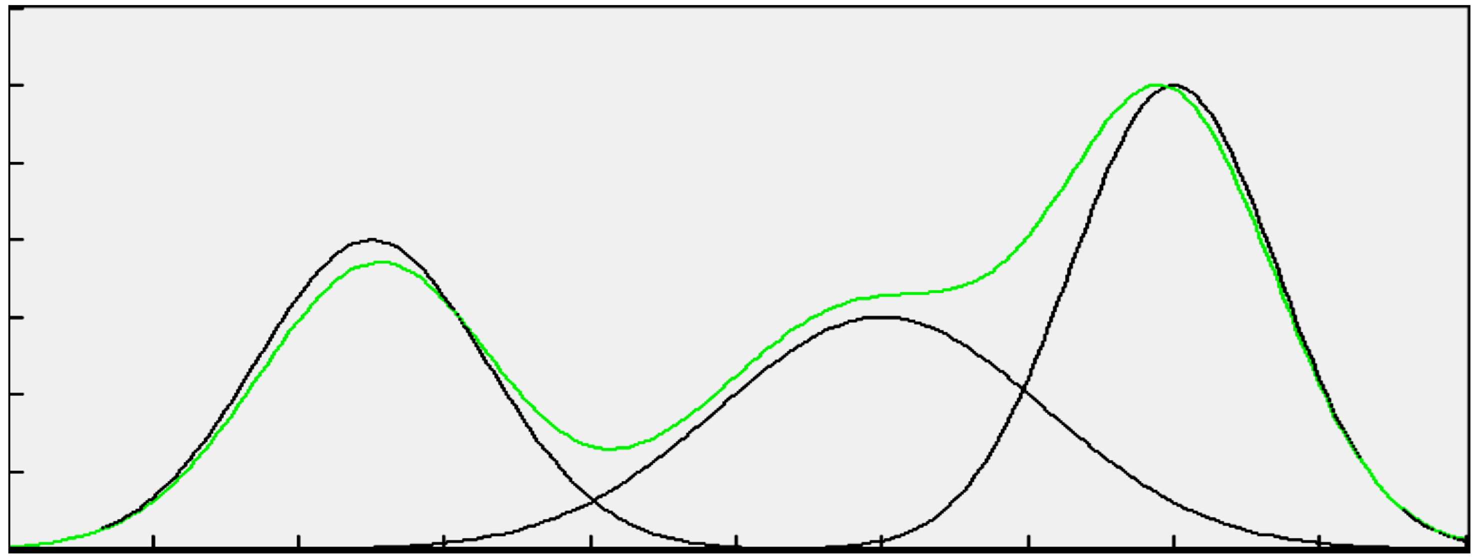




Mixture Decomposition

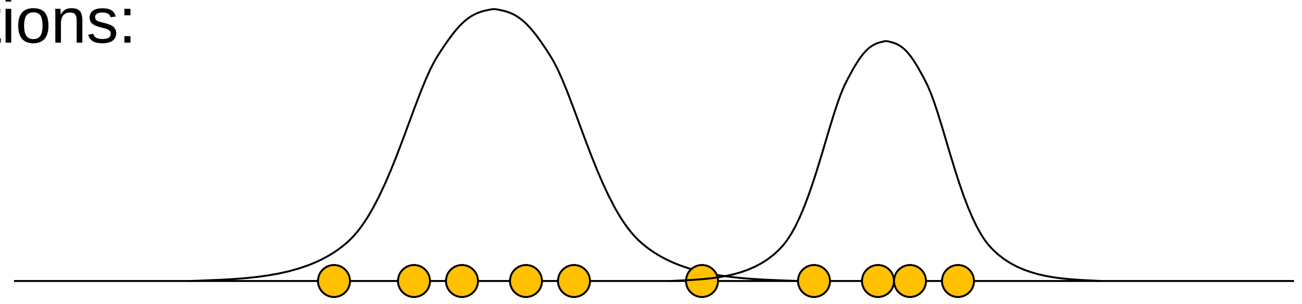
Mixture decomposition, where we want to estimate a probability distribution as the linear combination of a set of parametric components, can also be considered as a method of clustering, i.e., each component as a cluster.

Example: Modeling a 1-D distribution with mixture of Gaussians:

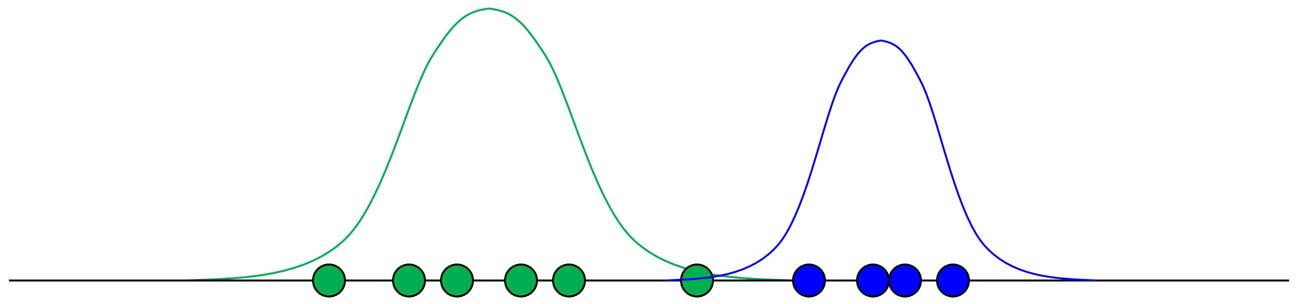


Incomplete Data Problem

Assume that we want to model the following 1-D data distribution with two clusters represented as Gaussian distributions:



If we know the “cluster labels” of the individual points, we can use maximum-likelihood to easily estimate the parameters of the two Gaussians:



The problem is that the cluster labels are not known. This is the “incomplete data problem”.

Mixture Model

- Each cluster is represented by a parametric probability distribution function (pdf), $p(\boldsymbol{x}|C_j)$. Gaussian is the most common form of pdf here.

$$p(\boldsymbol{x}) = \sum_{j=1}^{m} p(\boldsymbol{x} \mid C_j) P_j$$

- The number of clusters is assumed to be known, just like k-means.
- The EM algorithm is the most common method.
- If a crisp partition is desired, when the algorithm converges, each $\boldsymbol{x}$ is assigned to the cluster with the largest $P(C_j|\boldsymbol{x})$.

EM Algorithm

If we know the cluster label j_i of each sample $\boldsymbol{x}_i$, (that is, we have the complete data), the likelihood function associated with $\boldsymbol{x}_i$ can be expressed as

$$p(\boldsymbol{x}_i \mid C_{j_i}; \boldsymbol{\theta}_{j_i}) P_{j_i}$$

The overall likelihood becomes

$$\prod_{i=1}^{N} p(\boldsymbol{x}_i \mid C_{j_i}; \boldsymbol{\theta}_{j_i}) P_{j_i}$$

As the standard practice, we use the loglikelihood:

$$L(\boldsymbol{\theta}) = \sum_{i=1}^{N} \ln\left[p(\boldsymbol{x}_i \mid C_{j_i}; \boldsymbol{\theta}_{j_i}) P_{j_i} \right]$$

Maximum likelihood estimation: The parameters are selected such that the loglikelihood is maximized.

EM Algorithm

However, we do not know the cluster labels of the observed samples $\boldsymbol{x}_i$, so maximum likelihood estimation can not be applied directly. What we can try to do then is to maximize the ***expectation*** of the loglikelihood function:

$$E\left\{\sum_{i=1}^{N} \ln\left[p(\boldsymbol{x}_i \mid C_j;\boldsymbol{\theta}_j)P_j\right]\right\} = \sum_{i=1}^{N} E\left\{\ln\left[p(\boldsymbol{x}_i \mid C_j;\boldsymbol{\theta}_j)P_j\right]\right\}$$

The computation of the expectation is over the unknown cluster labels:

$$\sum_{i=1}^{N}\sum_{j=1}^{m} P(C_j \mid \boldsymbol{x}_i)\ln\left[p(\boldsymbol{x}_i \mid C_j;\boldsymbol{\theta}_j)P_j\right]$$

EM Algorithm

Now, how do we know $P(C_j|\boldsymbol{x}_i)$? If we have a guess of the mixture composition, we can compute all $P(C_j|\boldsymbol{x}_i)$ based on this guess. The expectation becomes

$$\sum_{i=1}^{N}\sum_{j=1}^{m} P(C_j \mid \boldsymbol{x}_i;\boldsymbol{\Theta}) \ln\left[p(\boldsymbol{x}_i \mid C_j;\boldsymbol{\theta}_j)P_j\right]$$

Parameters of the mixture, including $\boldsymbol{\theta}$ and all P_j.

with

$$P(C_j \mid \boldsymbol{x}_i;\boldsymbol{\Theta}) = \frac{p(\boldsymbol{x}_i \mid C_j;\boldsymbol{\theta}_j)P_j}{\sum_{k=1}^{m} \ln\left[p(\boldsymbol{x}_i \mid C_k;\boldsymbol{\theta}_k)P_k\right]}$$

$$P_j = \frac{1}{N}\sum_{i=1}^{N} P(C_j \mid \boldsymbol{x}_i;\boldsymbol{\Theta})$$

EM Algorithm

Expectation-Maximization Algorithm:

- Iterative method.
- Need to choose the number of mixture components, as well as their initial weights (P_j) and parameters (θ_j).

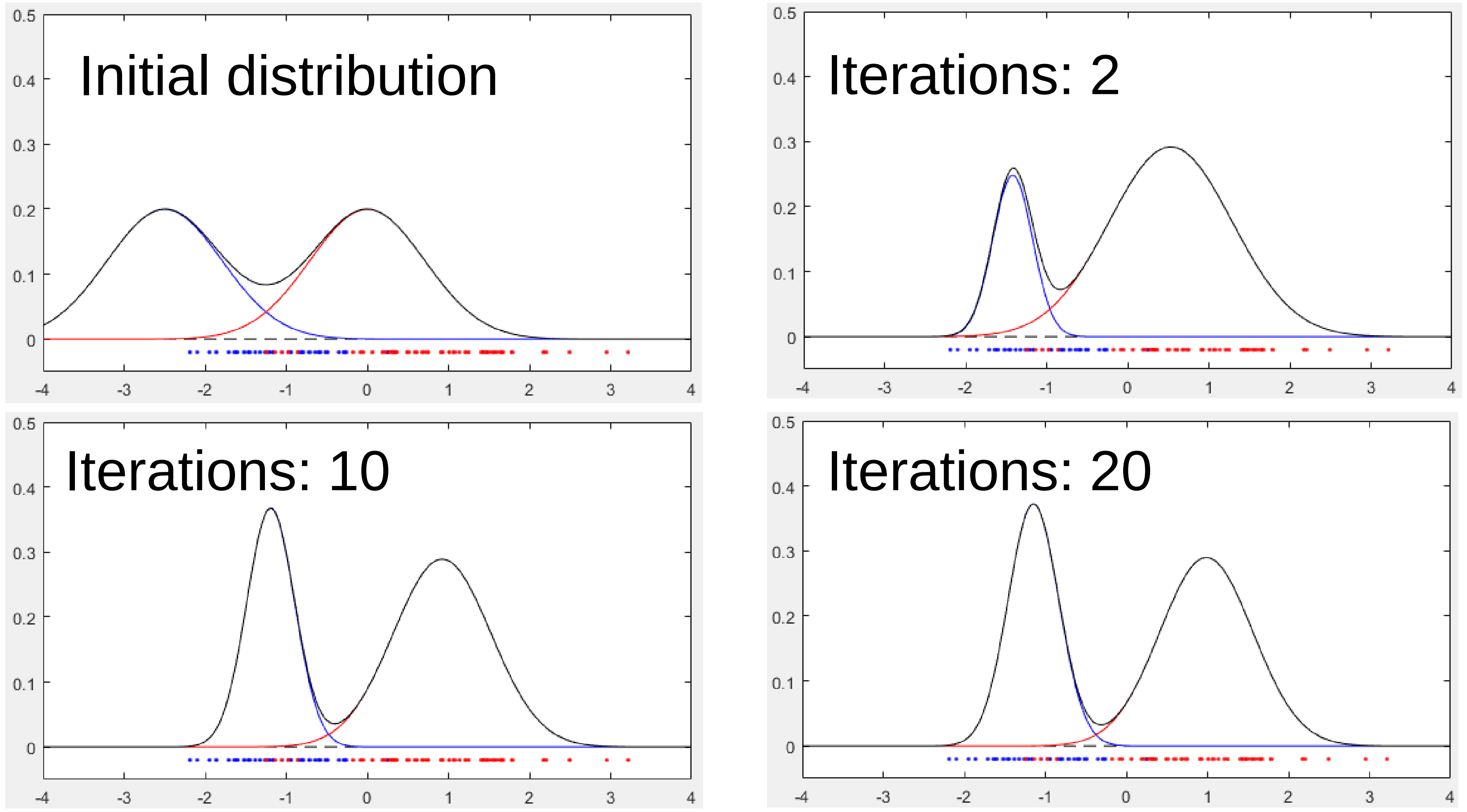
In each iteration:

- ***E-step***: Compute the expectation of $L(\boldsymbol{\theta})$, given the current guess of parameters ($\boldsymbol{\theta}$ and all P_j).
- ***M-step***: Find the new set of parameters ($\boldsymbol{\theta}$ and all P_j) that maximizes the expectation.

This process always converges to a local maximum of the expectation of $L(\boldsymbol{\theta})$, although may be slow, and the result is initialization dependent.

EM Algorithm Example

For a set of 1-D points, estimate their distribution as the linear combination of two Gaussians:

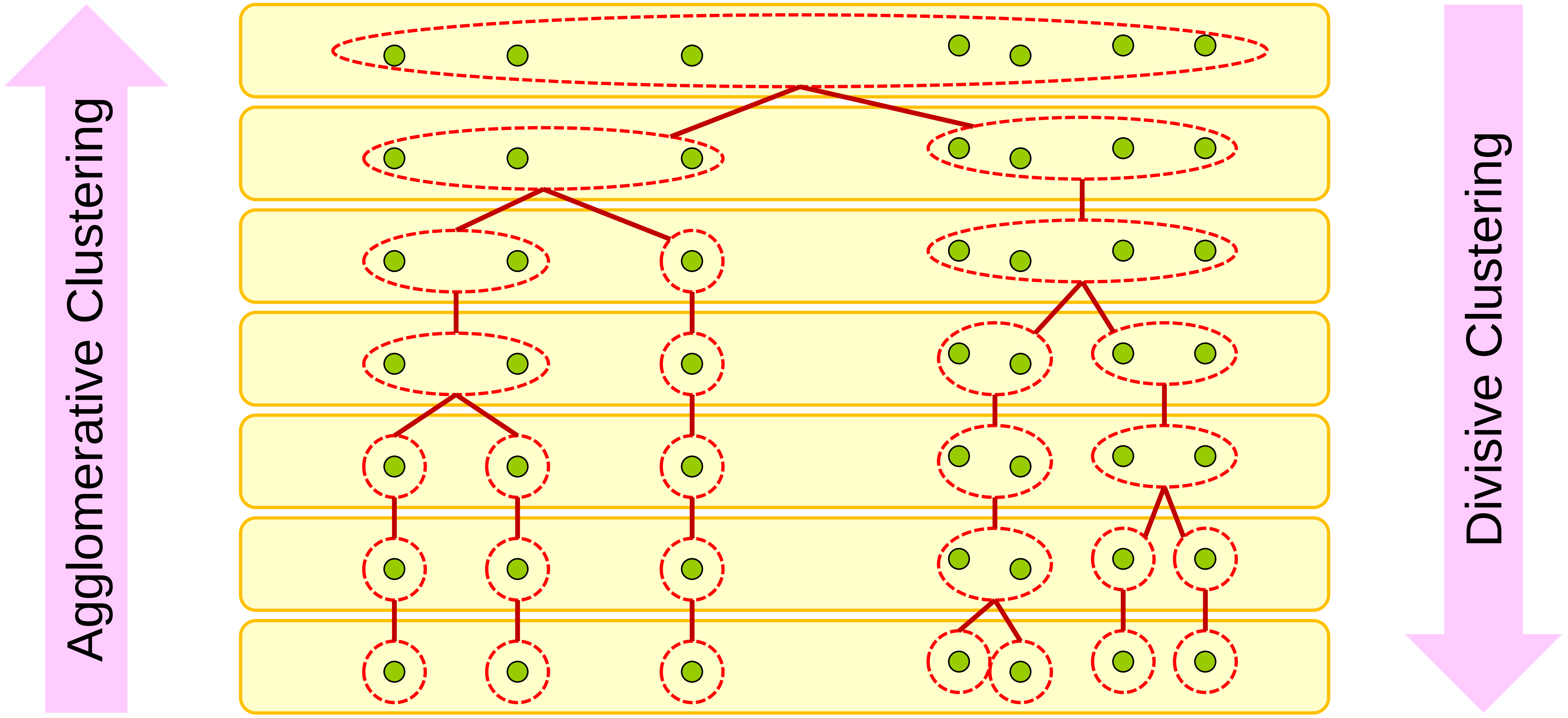


Hierarchical Clustering

- A group of algorithms that produce a hierarchy of possible clusterings.
- Common for clustering relational data.
- It is not required to have a pre-specified number of clusters.
- In some applications, the whole hierarchy is useful.
- In many applications, we select one clustering from the hierarchy according to some criteria.
- Two main categories:
  - **Agglomerative clustering**: Bottom-up; repeated cluster merging
  - **Divisive clustering**: Top-down; repeated cluster splitting

Nested Clusterings

The output of a hierarchical clustering algorithm is a series of nested clusterings.



Drawback: We can not recover from a "bad" decision in the process.

Agglomerative Algorithm

The term linkage represents how cluster-cluster proximities are computed from their members.

Assume that we are using a dissimilarity measure. Commonly used cluster-cluster dissimilarity:

$$d_{min}(C_1, C_2) = \min_{\boldsymbol{x} \in C_1, \boldsymbol{y} \in C_2} d(\boldsymbol{x}, \boldsymbol{y})$$ (**single link**)

$$d_{max}(C_1, C_2) = \max_{\boldsymbol{x} \in C_1, \boldsymbol{y} \in C_2} d(\boldsymbol{x}, \boldsymbol{y})$$ (**complete link**)

These can be computed recursively as part of the clustering algorithm. Let C_q be the cluster obtained by merging C_i and C_j, then for all C_s,

$$d_{min}(C_q, C_s) = \min\{d_{min}(C_s, C_i), d_{min}(C_s, C_j)\}$$ (**single link**)

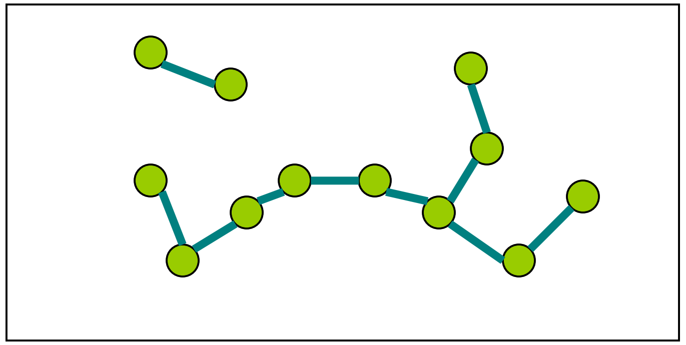
$$d_{max}(C_q, C_s) = \max\{d_{max}(C_s, C_i), d_{max}(C_s, C_j)\}$$ (**complete link**)

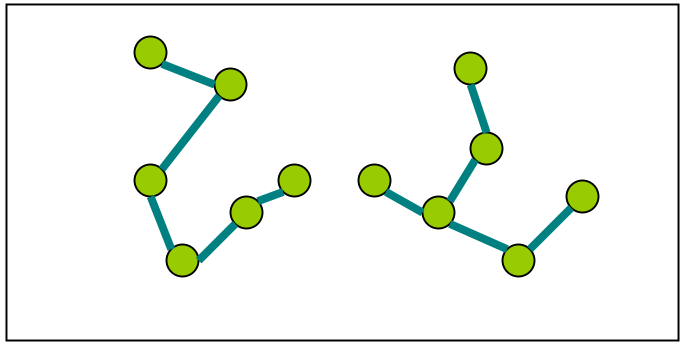
2-D Example

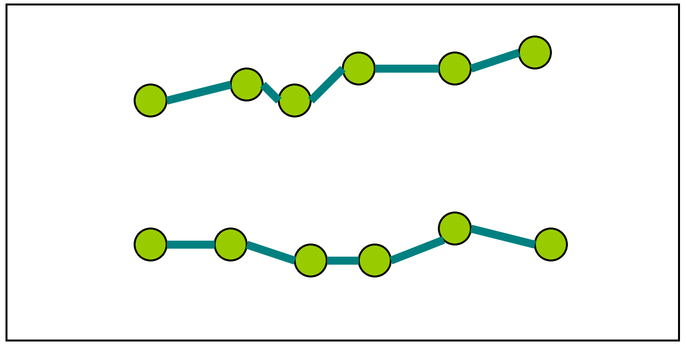
For each dataset, repeat the process until there are only two (or the desired number of) clusters left.

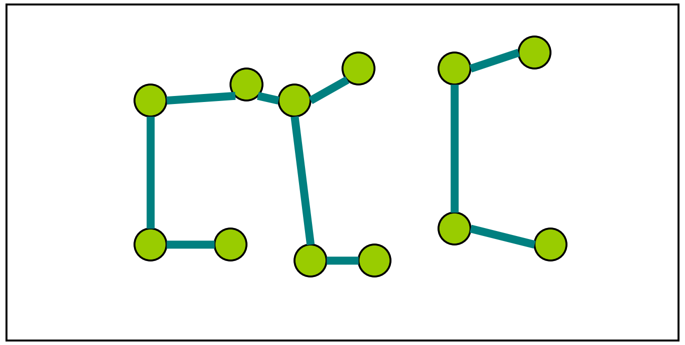
single-link

complete-link

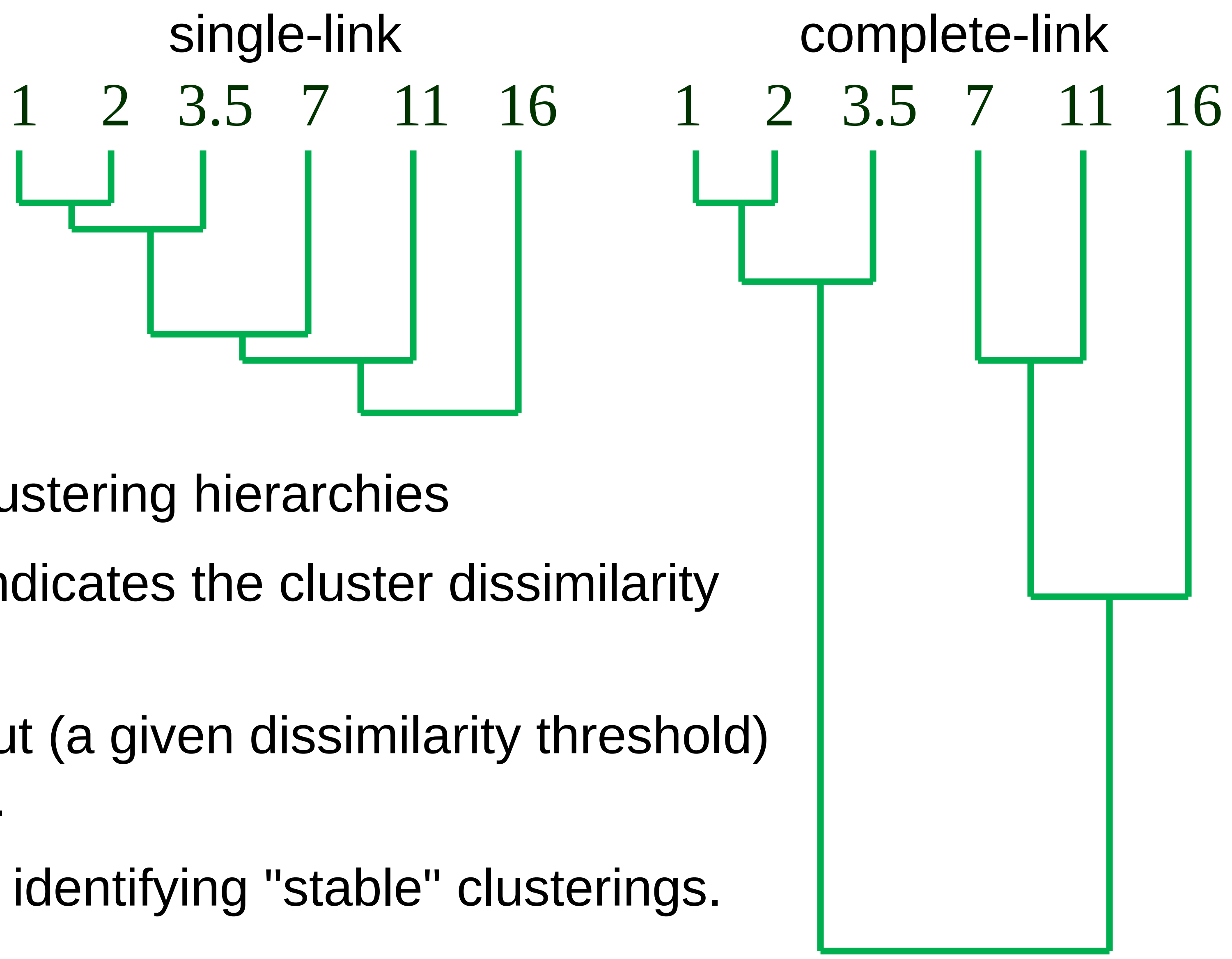








Dendograms (1-D Example)



- Visualization of clustering hierarchies
- Vertical position indicates the cluster dissimilarity at merging.
- Each horizontal cut (a given dissimilarity threshold) gives a clustering.
- Useful for visually identifying "stable" clusterings.

Linkage in Agglomerative Clustering

- Different ways of computing cluster-cluster proximity (i.e., linkage criteria) give different properties of the resulting clusters:
  - "Single linkage" works better for elongated clusters, and are prone to wrong cluster linking due to outliers.
  - "Complete linkage" works better for compact, spherical, similar sized clusters (similar to k-means).
- While these two are the standard and “extreme” ones, there are other commonly used options:
  - Average linkage
  - Ward linkage

Average Linkage

Assume that we are using a dissimilarity measure. Cluster-cluster dissimilarity in average linkage is the average pairwise dissimilarity between their points:

$$d_{avg}(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{\boldsymbol{x} \in C_1, \boldsymbol{y} \in C_2} d(\boldsymbol{x}, \boldsymbol{y})$$

These can be computed recursively as part of the clustering algorithm. Let C_q be the cluster obtained by merging C_i and C_j, then for all C_s,

$$d_{avg}(C_q, C_s) = \frac{|C_i|}{|C_i| + |C_j|} d(C_s, C_i) + \frac{|C_j|}{|C_i| + |C_j|} d(C_s, C_j)$$

Ward Linkage

Ward linkage assumes that a distance measure between clusters is used (a quite common case) and the cluster centroids can be computed. Cluster-cluster dissimilarity is defined as the increase of total squared distances of the points to the cluster centroids:

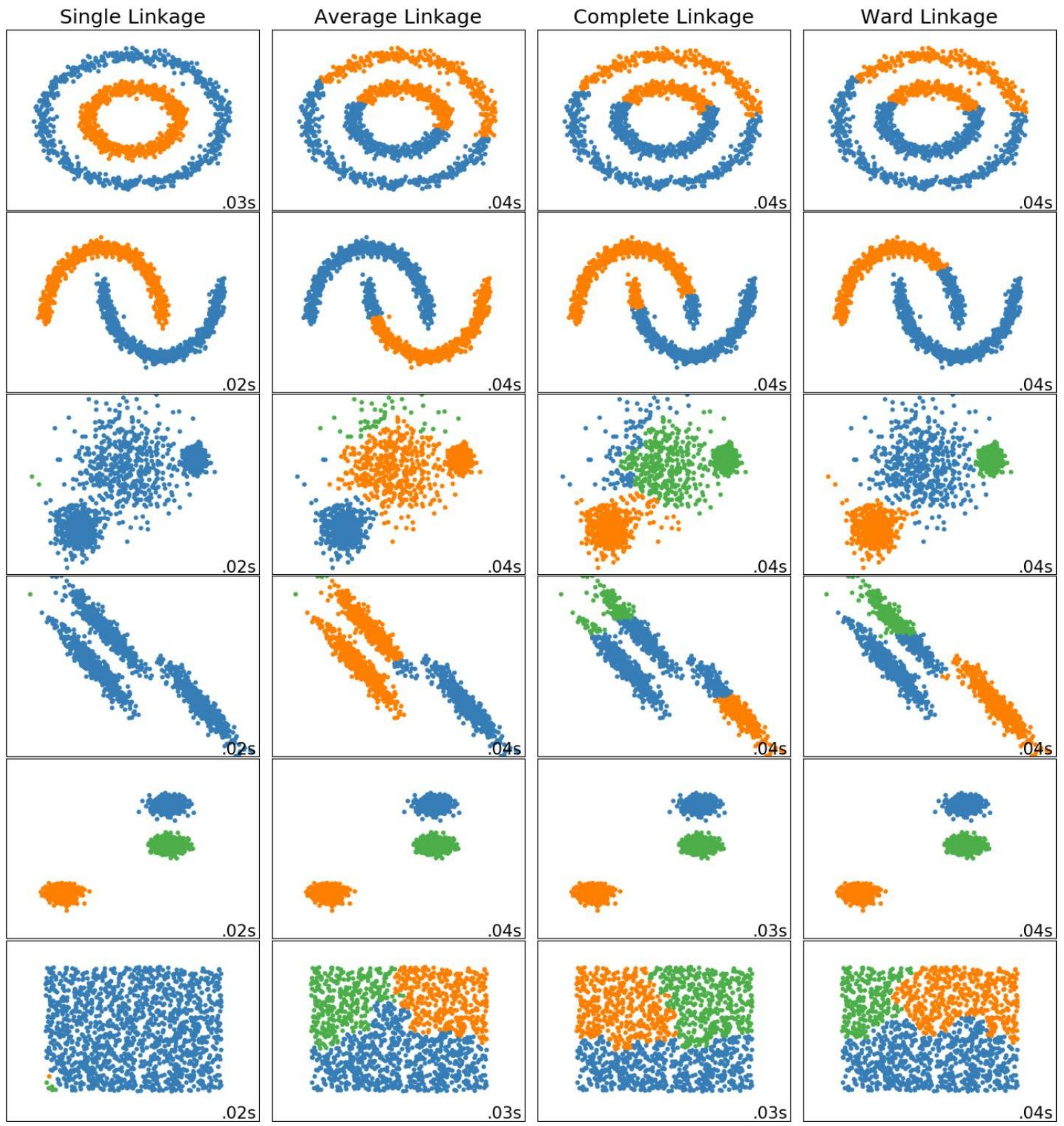
$$d_{Ward}(C_i, C_j) = \sum_{\boldsymbol{x} \in C_q} \left\| \boldsymbol{x} - \boldsymbol{m}_q \right\|^2 - \sum_{\boldsymbol{x} \in C_i} \left\| \boldsymbol{x} - \boldsymbol{m}_i \right\|^2 - \sum_{\boldsymbol{x} \in C_j} \left\| \boldsymbol{x} - \boldsymbol{m}_j \right\|^2$$

Here the vectors $\boldsymbol{m}$ represents the cluster centroids, and C_q is the cluster obtained by merging C_i and C_j. With a little derivation we can see that

$$d_{Ward}(C_i, C_j) = \frac{|C_i||C_j|}{|C_i| + |C_j|} \left\| \boldsymbol{m}_i - \boldsymbol{m}_j \right\|^2$$

Examples of Different Linkages

Some examples with common 2-D simulated datasets:



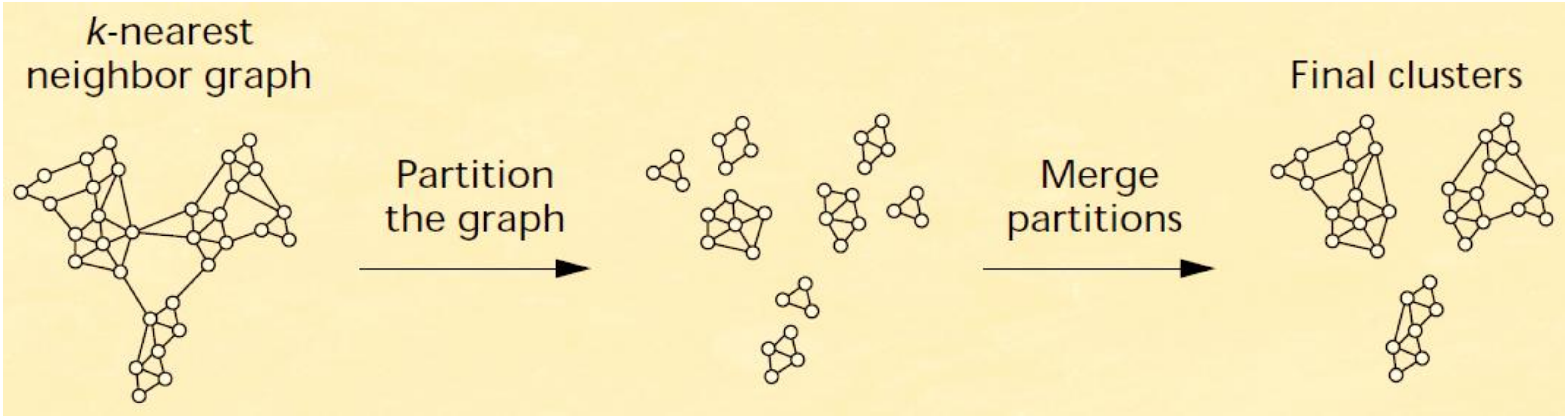
https://stats.stackexchange.com/questions/426760/should-we-most-of-the-time-use-wards-method-for-hierarchical-clustering

Computational Complexity

- Brute-force: $O(N^3)$ for obtaining the whole hierarchy of clusterings.
- Faster versions:
  - Single-link: $O(N^2)$: This is because single-link is equivalent to minimum-spanning-tree if there is no duplicate value in the original similarity/dissimilarity matrix.
  - Complete/average/Ward-link: $O(N^2 \log N)$: Use a heap (as a priority queue).
- To speed up the process, a common approach is to run the clustering on a reduced and approximate representation of the whole dataset.
  - BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)
  - CURE (Clustering Using REpresentatives)

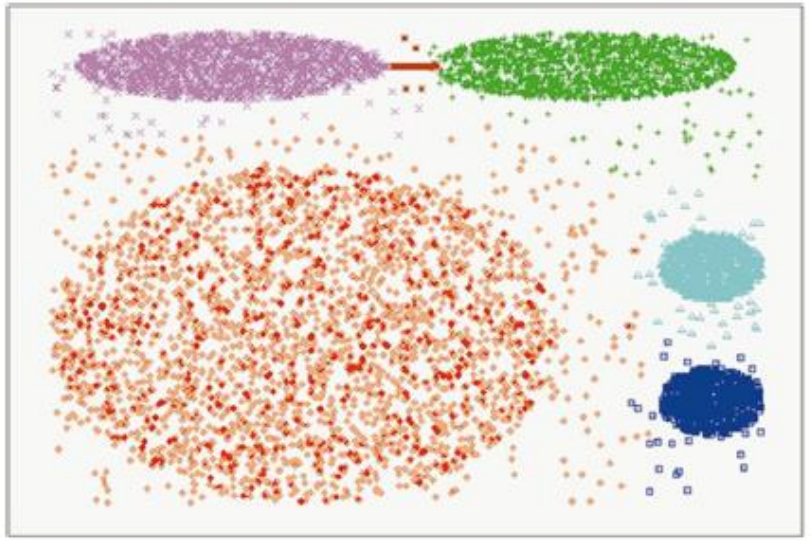
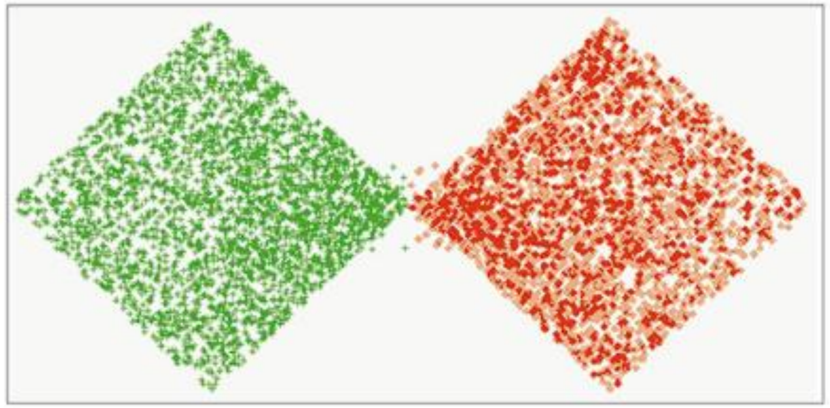
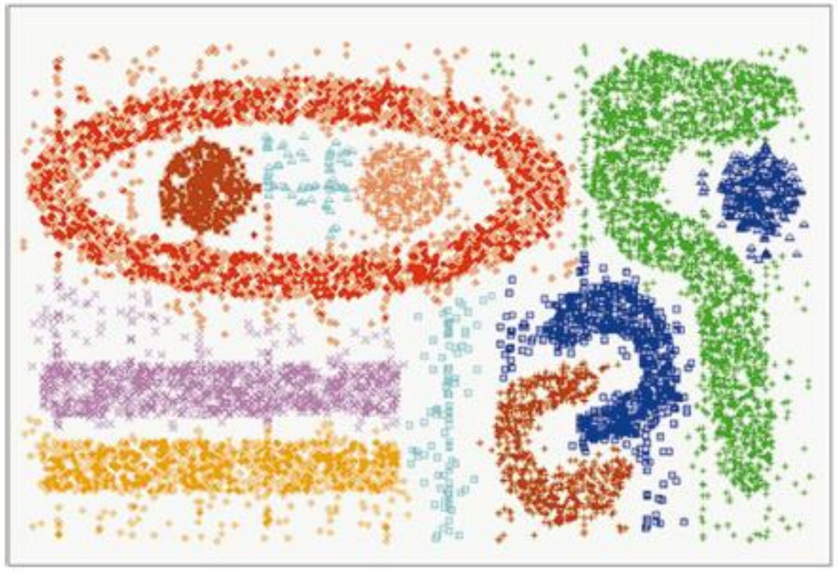
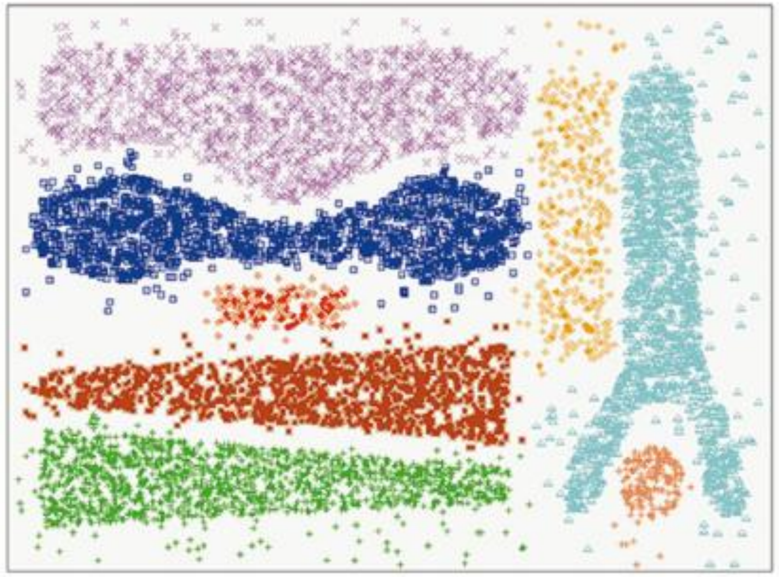
Chameleon

- Motivation: To be able to cluster data with highly variable cluster structures.
- Pre-clustering: A partition of the sparse kNN graph.
- Hierarchical agglomeration is then applied to this pre-clustering



Chameleon

Some results of synthetic datasets from the original paper:

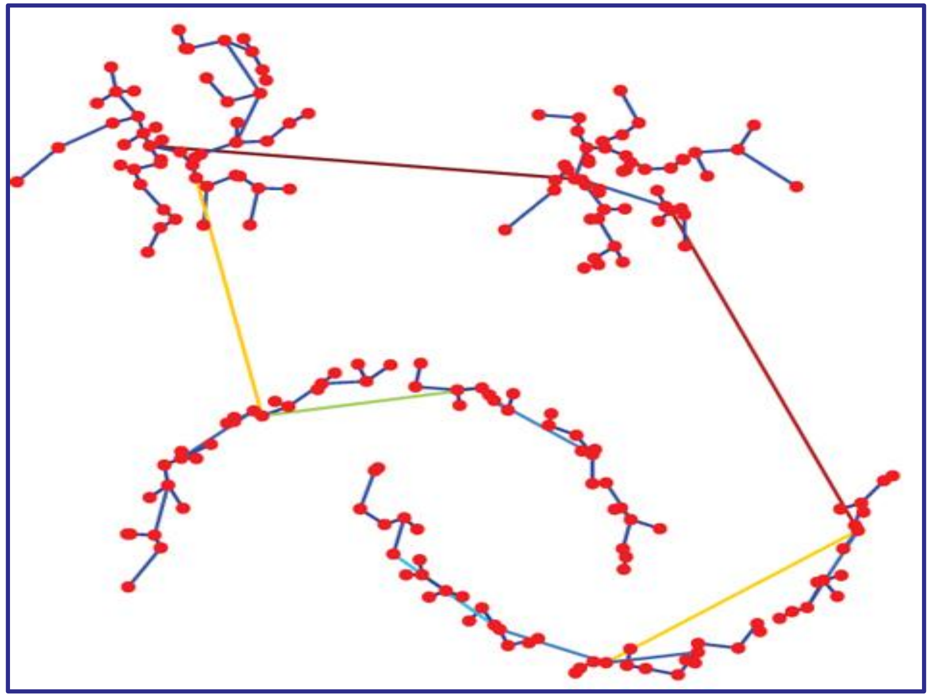
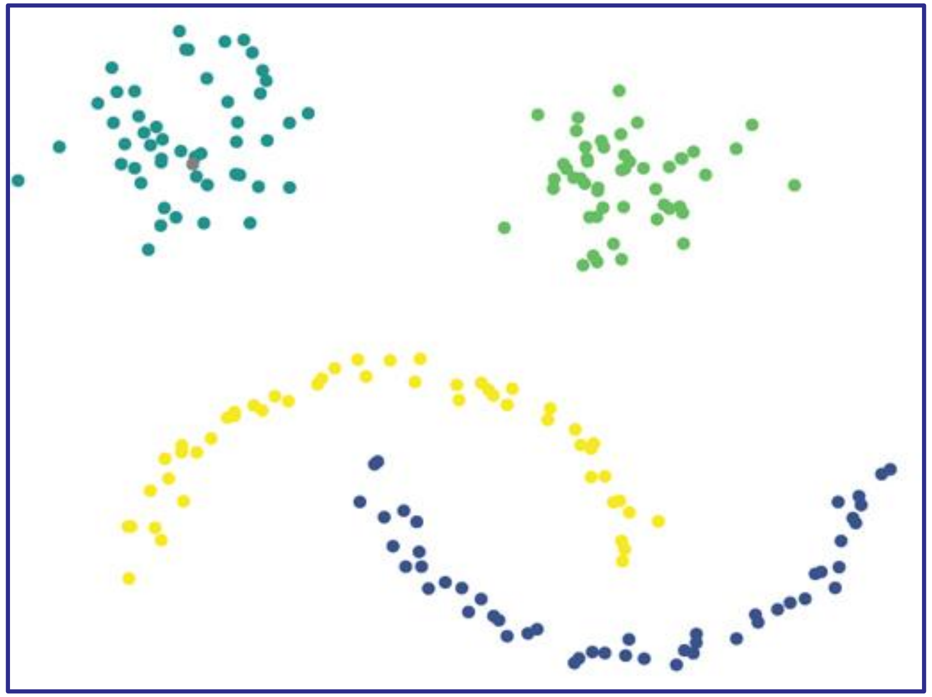





Divisive Algorithms

- In each iteration, We need to determine
  - Which cluster to split?
  - How to split it?
- Idea: Find the splitting that gives the most dissimilar (or least similar) pair of new clusters. This depends on the proximity measure used.
- To split any given cluster, it is computationally very expensive to find the "optimal" splitting. Approximations to the optimal splitting are more practical.
- Common approaches:
  - Bisecting k-means algorithm (k=2)
  - EM algorithm (2 components)
  - Spectral clustering (graph cut, to be discussed later)

Divisive Clustering with Density Peaks

- Idea: Build a tree of data points based on local densities.
  - Parent = nearest neighbor with higher density
  - Parent-child edge weight = dissimilarity
- Clustering: Iterative removal of edges with larger to smaller weights
- Example results (synthetic dataset) from the original paper:

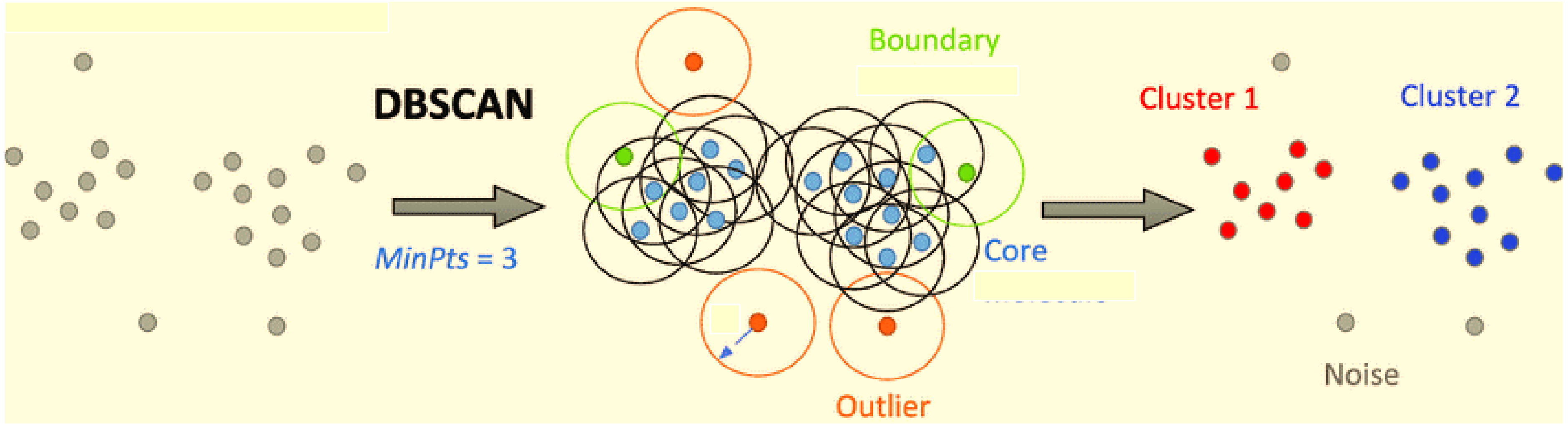



Density Based Clustering

- The basic ideas:
  - Each cluster is a somewhat connected high-density region.
  - Clusters are separated by low-density regions.
  - Samples in low-density regions are considered noise.

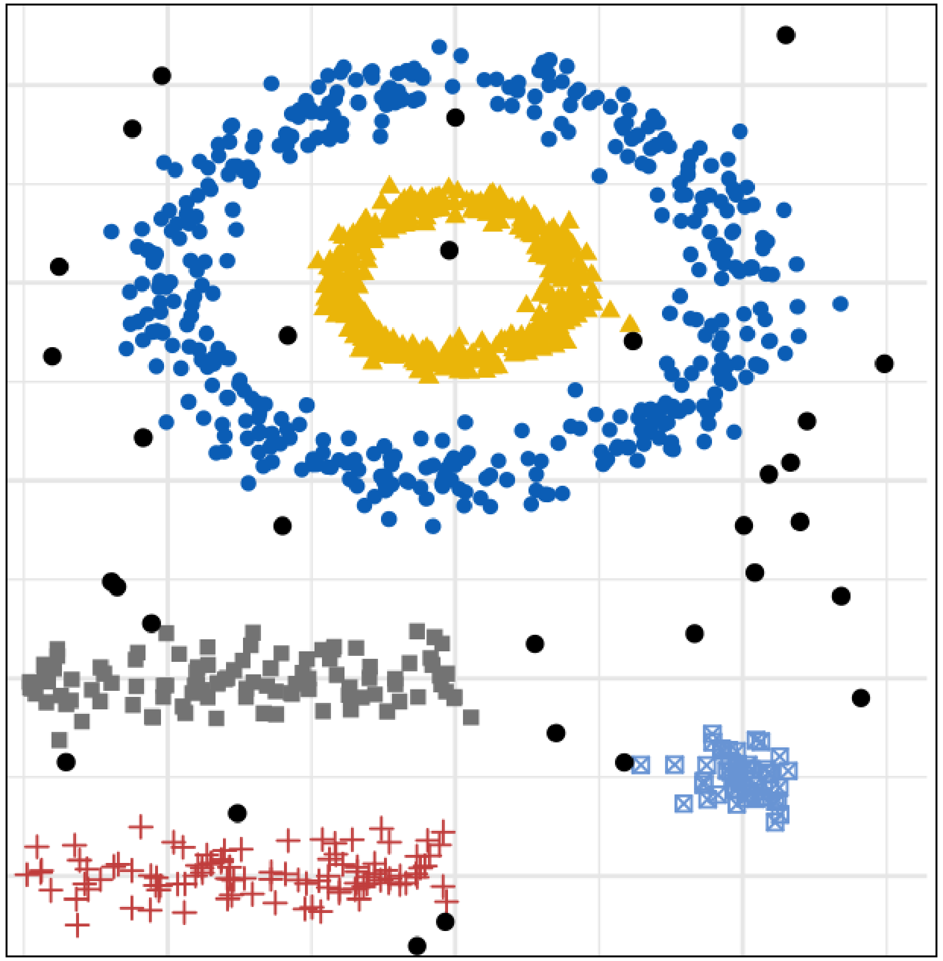
DBSCAN

- **Density-Based Spatial Clustering of Applications with Noise** (**DBSCAN**) :
- Core points are those points with sufficiently high density, i.e., #samples-in-neighborhood (within radius ε) $\geq$ $MinPts$.
- Make a directed graph: For each core point, generate one edge to each of its neighbors within the radius ε.
- Make connected components of the core points. Each component forms a cluster.
- Include in a cluster the non-core-point neighbors its core points. (Each non-core point only belongs to one cluster.)



DBSCAN

- DBSCAN has become quite popular given its robustness to noise and flexibility of cluster shapes.
- Challenges:
  - Efficiency (for finding the neighbors)
  - How to determine the radius?
  - Does not adapt well to clusters with different densities



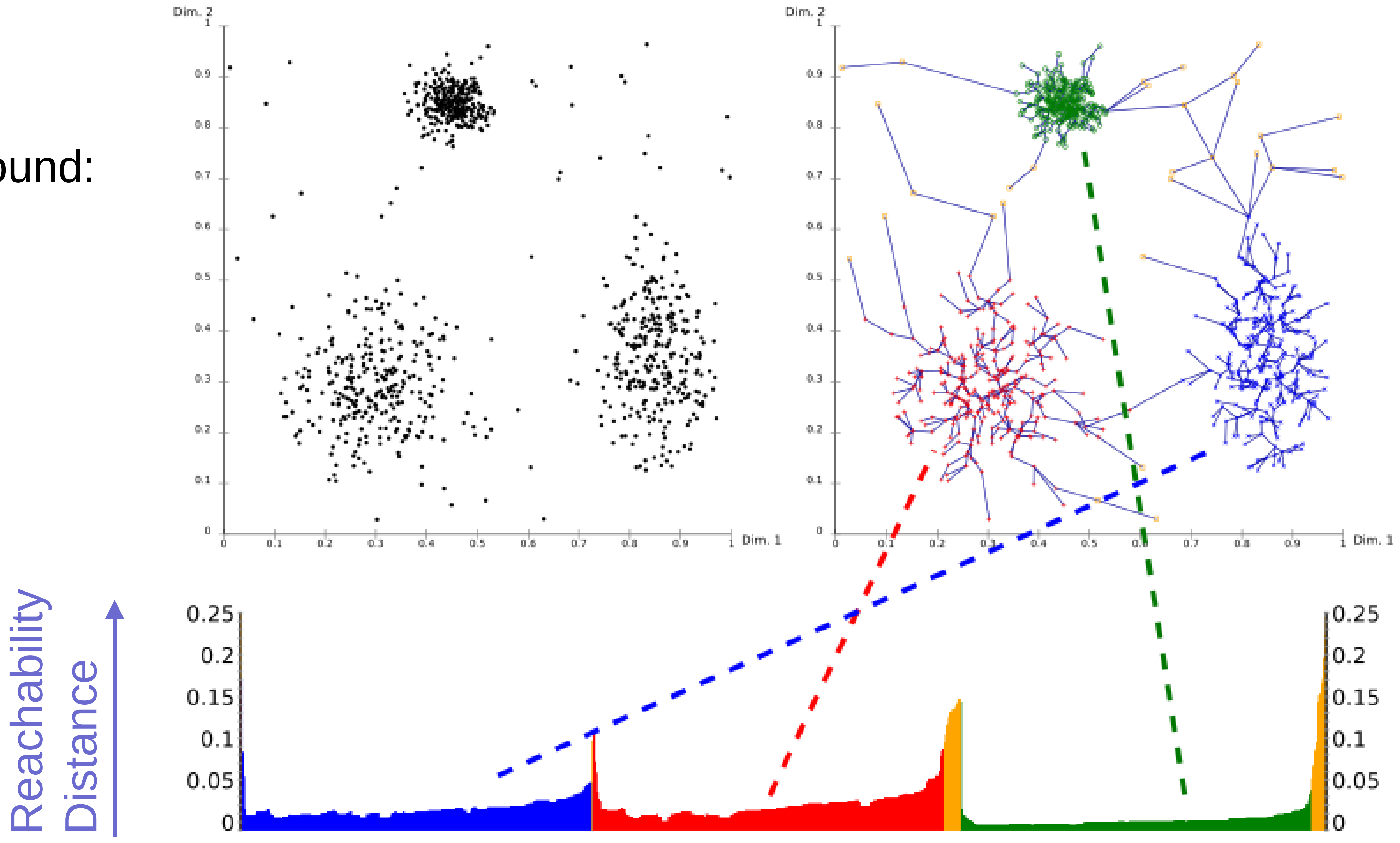
www.datanovia.com

OPTICS

- Ordering Points To Identify the Clustering Structure (OPTICS) is a clustering tool with basic concepts taken from DBSCAN.
- Only an ordering of points is created, not an actual clustering. However, the ordering exhibits the cluster structures in the data.
- Concepts:
  - The core distance of a point *p*: For the given *MinPts*, the smallest ε for *p* to be a core point.
  - The reachability distance of a point *p* from a point *o*: Maximum of the core-distance of *o* and the distance between *p* and *o*.
- Use a priority queue ordered by increasing reachability distance
  - Each time a point *o* is extracted, insert its neighbors *p* with their reachability distances from *o*.

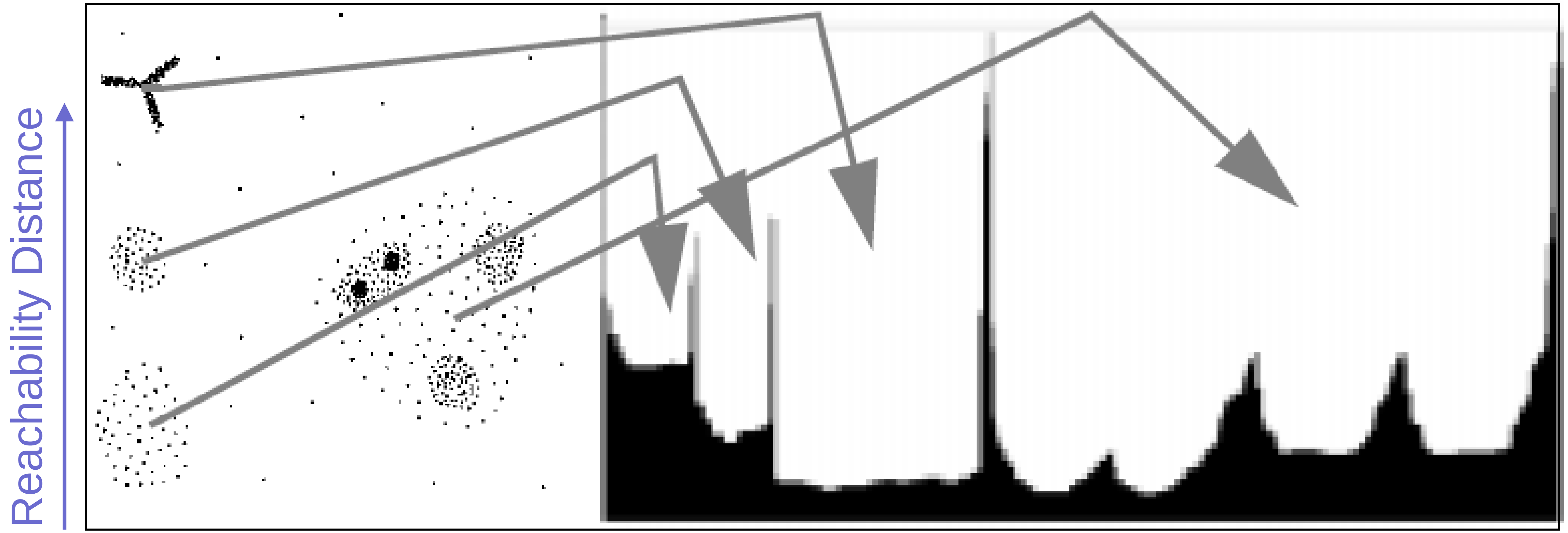
OPTICS - Example

With clusters found:



OPTICS - Example

- Example result from the original paper.



Mode-Seeking Algorithms

- Mode (in statistics):
  - For a discrete variable, its mode is the value(s) with the most occurrences in a set of samples.
  - For a continuous variable, its mode(s) are the values of the **local** maxima of its probability distribution.
- Mode-seeking: To (iteratively) move a point in the feature space towards a mode (local maxima of density).
- Mode-seeking for clustering: Use an iterative procedure to identify the modes (local maxima of estimated density), and treat each distinct mode as a cluster.

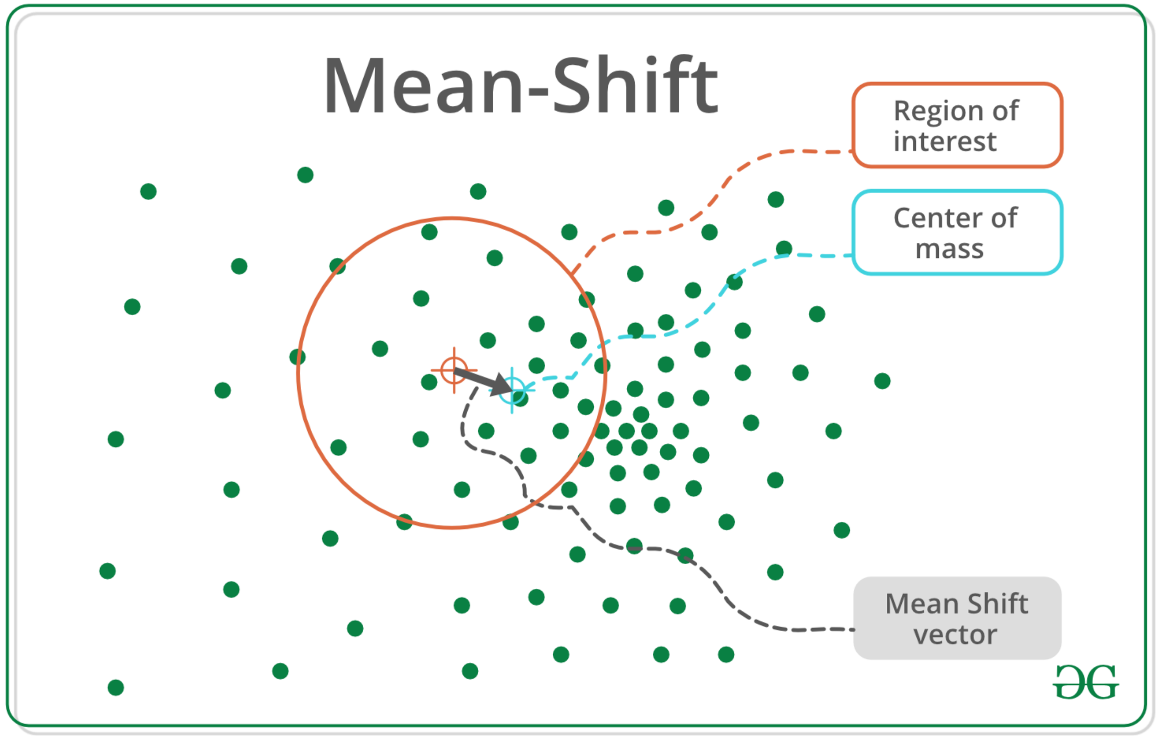
Mean-Shift

- A **kernel function** is needed for local density estimation from discrete samples. A typical kernel function is isotropic and has one hyper-parameter: its **bandwidth** (range of interest).

$$density(\mathbf{x}) = \sum_{\mathbf{x}_i} K\left(\frac{\|\mathbf{x}_i - \mathbf{x}\|}{h}\right)$$

- Each step of mean-shift involves moving a point $\mathbf{x}$ to a new location that maximizes the estimated density according to the kernel centered at $\mathbf{x}$.

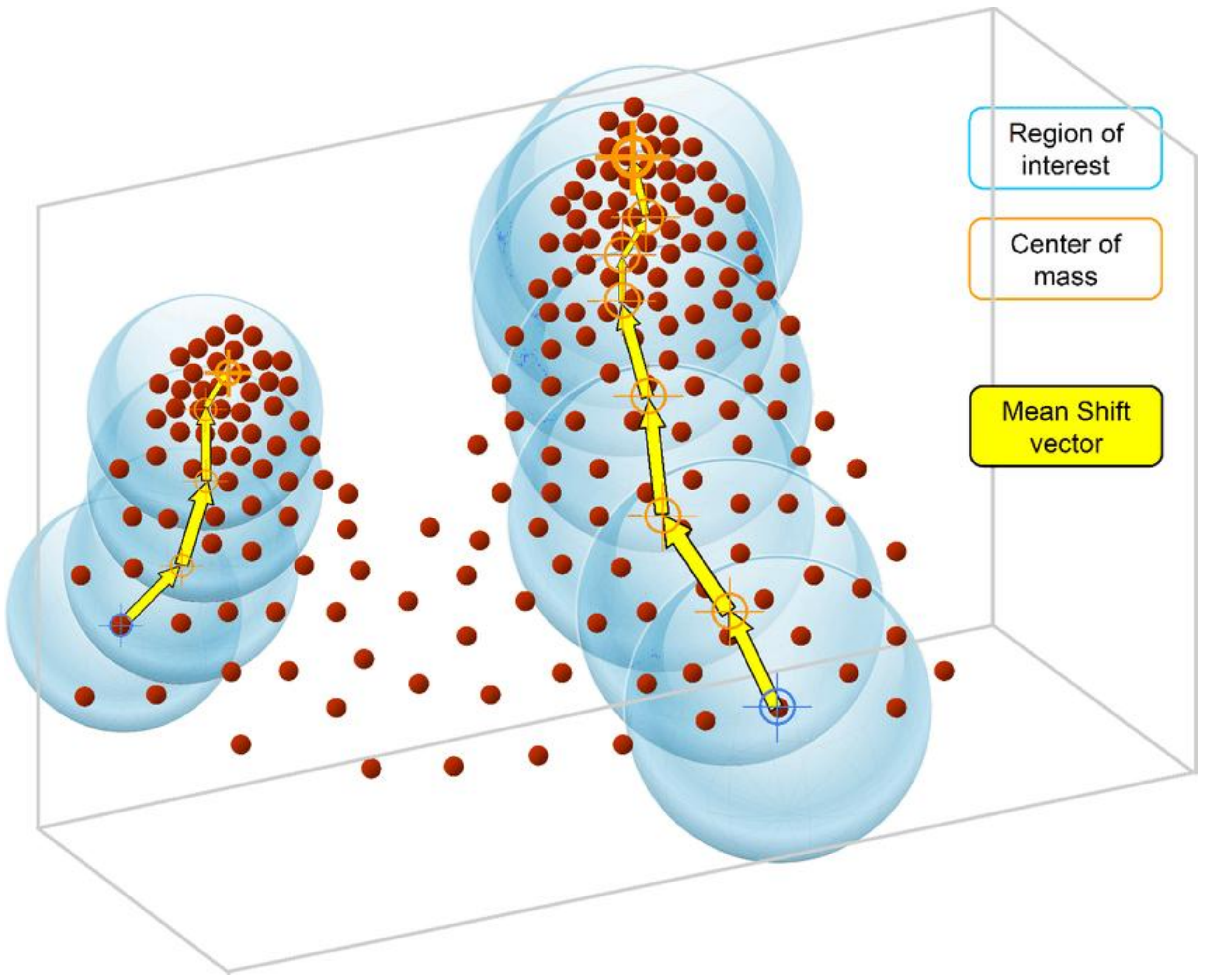
$$\mathbf{x} \leftarrow \frac{\sum_{\mathbf{x}_i} K(\mathbf{x}_i - \mathbf{x})\mathbf{x}_i}{\sum_{\mathbf{x}_i} K(\mathbf{x}_i - \mathbf{x})}$$



https://www.geeksforgeeks.org/ml-mean-shift-clustering/

Mean-Shift for Clustering

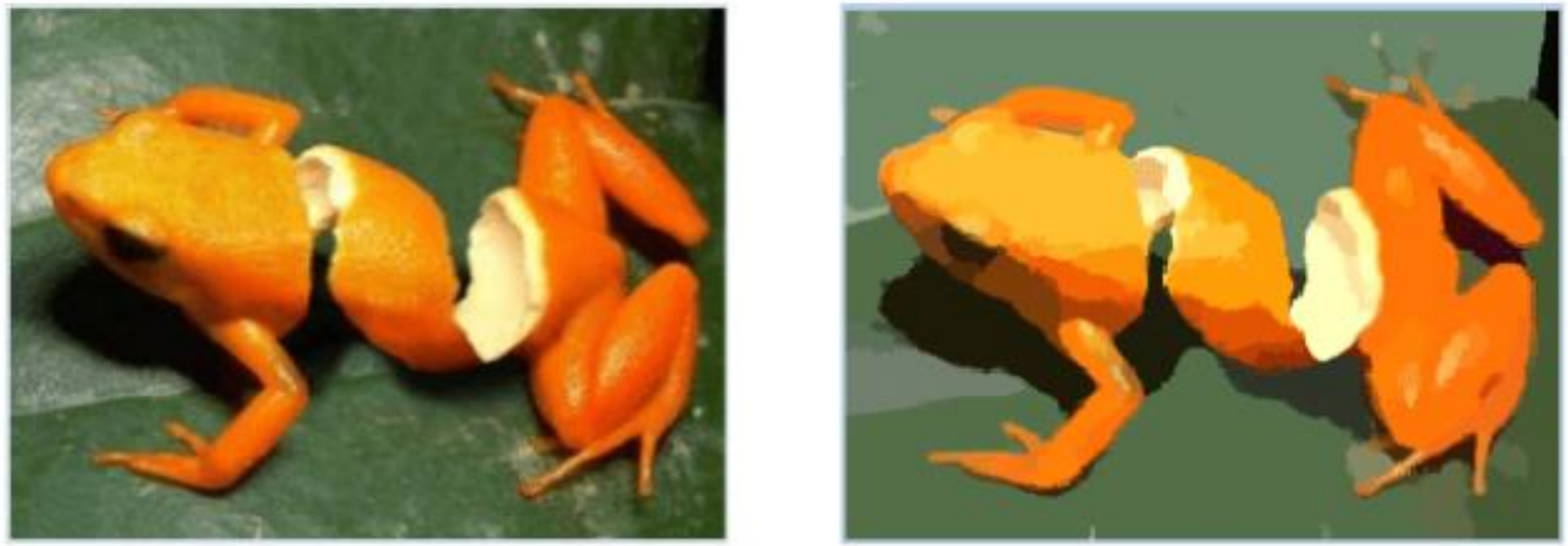
- The update of each x stops when its location converges (a mode is found). Repeat this multiple times to find multiple modes.
- Merge the modes that are within the bandwidth of each other.
- Each remaining mode represents cluster, and the points that move towards this mode is assigned to this cluster.



https://www.researchgate.net/figure/Intuitive-description-of-the-mean-shift-procedure-find-the-densest-regions-in-the_fig3_326242239

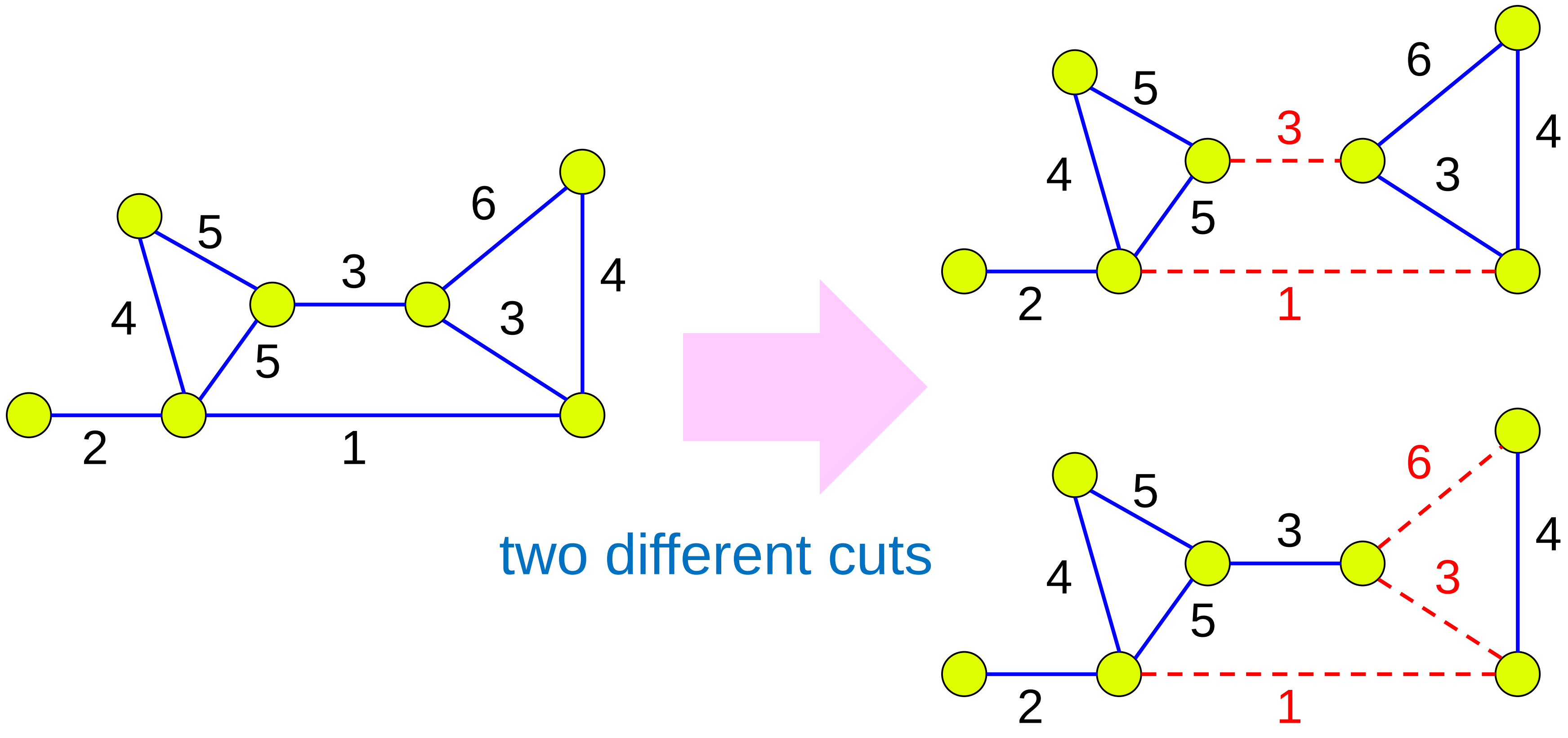
Mean-Shift for Clustering

- No constraint on the cluster shapes (similar to DBSCAN and unlike k-means).
- We can obtain the cluster prototypes (the modes).
- It can adapt to clusters with different densities.
- Hyperparameter: Bandwidth (similar to the role of k for k-means)
- Example application in color image segmenration:


https://www.researchgate.net/figure/Mean-Shift-Color-Segmentation-Result-Sample_fig2_281268167

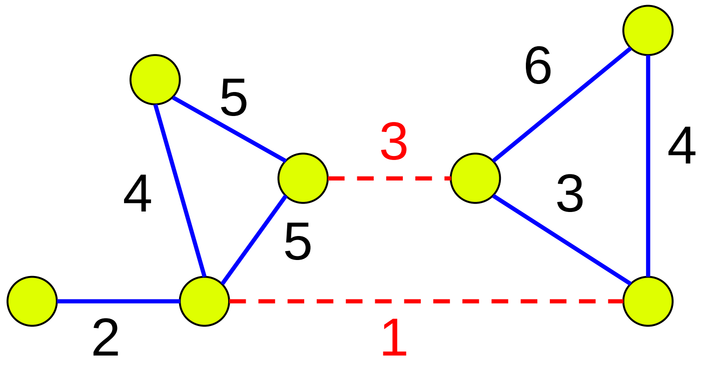
Graph-Cut Based Clustering

- Treat the samples to be clustered as a weighted graph. Use a **similarity measure** between samples as the weights.
- Idea: To cut the graph into two sub-graphs while minimizing the total weights of the removed edges.

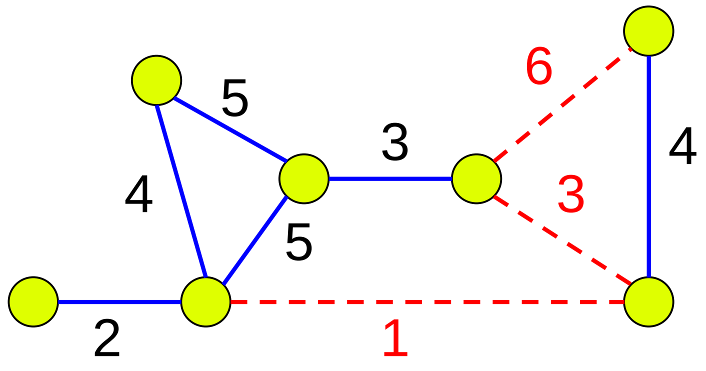


Graph-Cut Based Clustering

- Let V be the original set of vertices, and A and B be two subsets of V.
- Define $w(A,B)$ be the total weights of edges between vertices in A and B.
- A min-cut of a graph into two subgraphs minimizes $w(A,V\backslash A)$.



$w(A,V\backslash A) = 4$



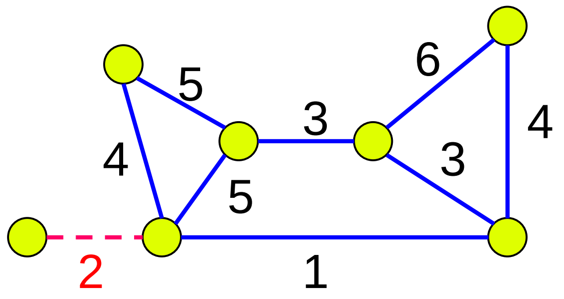
$w(A,V\backslash A) = 10$

- A min-cut of graph into k disjoint subsets minimizes

$$Cut(A_i,\cdots,A_k) = \sum_{i=1}^{k} w(A_i,\ V \setminus A_i)$$

Ideas of Balanced Graph Cuts

A simple min-cut is prone to generating results with very unbalanced sub-graphs, so we need to consider balance in our criteria. Examples:



- **Ratio cut**:

$$RatioCut(A_1,\cdots,A_k)=\sum_{i=1}^{k}\frac{w(A_i,\bar{A}_i)}{|A_i|}$$

total edge weight between the interior and exterior of a cluster

  - Attempt to balance based on cluster cardinalities.

- **Normalized cut**:

$$NCut(A_1,\cdots,A_k)=\sum_{i=1}^{k}\frac{w(A_i,\bar{A}_i)}{w(A_i,V)}$$

total weighted degree of the vertices in a cluster

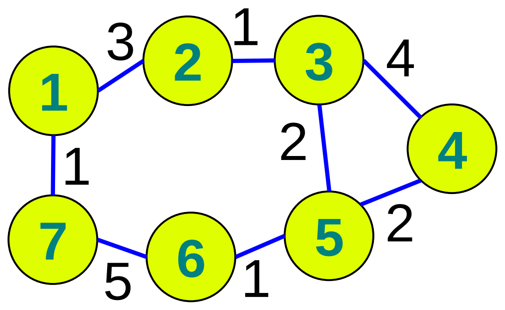
  - Attempt to balance based on cluster degrees. (The "weighted degree" of a vertex is the total weight of edges incident on the vertex, and the degree of a cluster is the total weighted degree of its vertices.)

Laplacian Matrix

- The derivation of spectral clustering starts with the **Laplacian matrix** of the graph:

$$L = D - W$$

  - W: Weighted similarity matrix (diagonal = 0)
  - D: A diagonal matrix where D_{ii} is the degree of the i^{th} vertex (the total weights of its incident edges).
  - L is symmetric and semi-positive definite.
  - The smallest eigenvalue of L is zero, with its eigenvector being any unit vector.



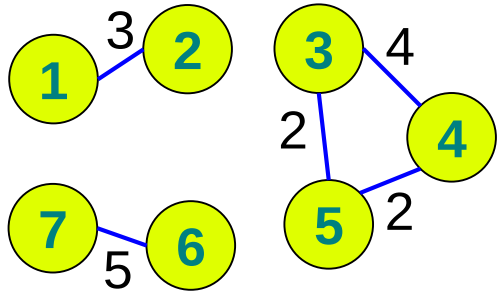
Laplacian matrix:

```
  4   -3    0    0    0    0   -1
 -3    4   -1    0    0    0    0
  0   -1    7   -4   -2    0    0
  0    0   -4    6   -2    0    0
  0    0   -2   -2    5   -1    0
  0    0    0    0   -1    6   -5
 -1    0    0    0    0   -5    6
```

Laplacian Matrix

Special case: A graph with exactly k connected components:

- L can be arranged to be block-diagonal.
- L has exactly k zero eigenvalues, one originating from each connected component.
- The eigenvectors of the k zero eigenvalues are orthogonal.
- The space spanned by these eigenvectors is the same as the one spanned by the indicator vectors of the k connected components.
  - The kth indicator vector is a length-n vector with binary elements indicating whether a vertex belongs to the kth component.



Laplacian matrix:

3	-3	0	0	0	0	0
-3	3	0	0	0	0	0
0	0	6	-4	-2	0	0
0	0	-4	6	-2	0	0
0	0	-2	-2	4	0	0
0	0	0	0	0	5	-5
0	0	0	0	0	-5	5

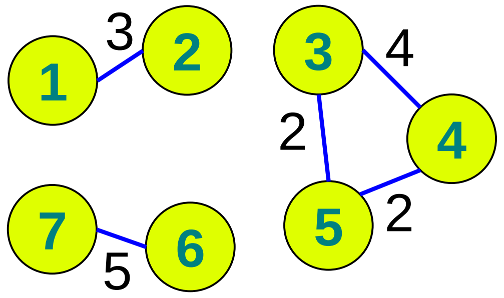
Laplacian Matrix and Graph Cut

- In the special case (a graph with k connected components), projection of the Laplacian matrix rows (one per vertex) to a space spanned by those first k eigenvectors lead to a perfect partition into k subgraphs.
- For this special case, the projection matrix F (F is the matrix composed of the normalized indicator vectors, which are eigenvectors of the k zero eigenvalues) minimizes

$$trace(F^T LF)$$

- It can be proved that this objective function is the same as the <u>**ratio cut**</u> in graph cut.

Laplacian Matrix (Example)



Laplacian matrix:

3	-3	0	0	0	0	0
-3	3	0	0	0	0	0
0	0	6	-4	-2	0	0
0	0	-4	6	-2	0	0
0	0	-2	-2	4	0	0
0	0	0	0	0	5	-5
0	0	0	0	0	-5	5

Eigenvalues:

0	0	0	6	6	10	10

Eigenvectors: (normalized) indicator vectors

0	-0.71	0	0	-0.71	0	0
0	-0.71	0	0	0.71	0	0
0.58	0	0	-0.41	0	0	-0.71
0.58	0	0	-0.41	0	0	0.71
0.58	0	0	0.82	0	0	0
0	0	-0.71	0	0	-0.71	0
0	0	-0.71	0	0	0.71	0

projections ➔ perfect clustering

0	-0.71	0	0	-0.71	0	0
0	-0.71	0	0	0.71	0	0
0.58	0	0	-0.41	0	0	-0.71
0.58	0	0	-0.41	0	0	0.71
0.58	0	0	0.82	0	0	0
0	0	-0.71	0	0	-0.71	0
0	0	-0.71	0	0	0.71	0

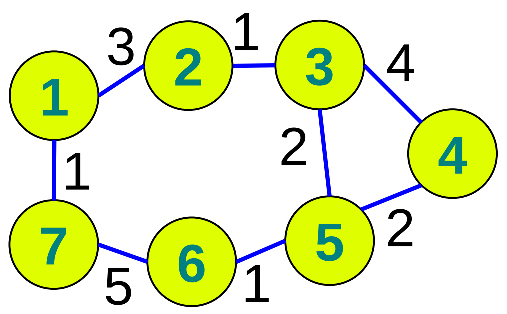
Laplacian Matrix and Graph Cut

- The task of identifying the optimal set of indicator vectors in general cases is NP-hard. However, we can relax the condition that the eigenvectors are indicator vectors, allowing them to be real-valued.
- With the relaxed condition, we can simply select the orthonormal eigenvectors corresponding to the $\underline{k \text{ smallest eigenvalues of } L}$ to form the projection matrix. This minimizes

$$\operatorname{tr}(F^T L F) \quad s.t. \quad F^T F = I$$

- We can then use any clustering algorithm with pre-specified number of clusters (e.g., k-means) in this new space to obtain the partition.
- Relaxed versions of ratio cut or normalized cut lead to a more practical class of clustering algorithms called **spectral clustering**. (This name comes from the analysis of the spectrum, or the set of the eigenvalues, of the Laplacian matrix.)

Laplacian Matrix (Example)



Laplacian matrix:

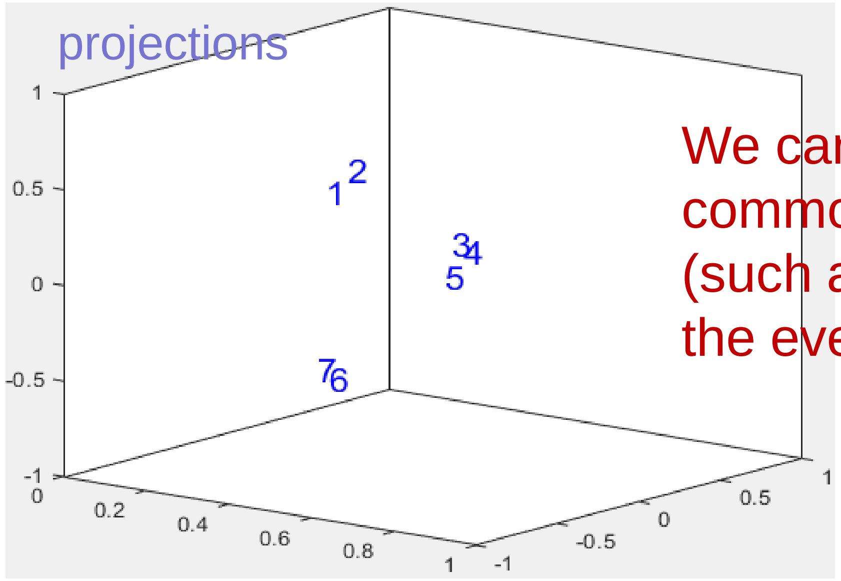
4	-3	0	0	0	0	-1
-3	4	-1	0	0	0	0
0	-1	7	-4	-2	0	0
0	0	-4	6	-2	0	0
0	0	-2	-2	5	-1	0
0	0	0	0	-1	6	-5
-1	0	0	0	0	-5	6

Eigenvalues:

0.00	1.04	1.38	6.76	7.00	10.64	11.19

Eigenvectors:

0.38	-0.35	0.48	-0.39	-0.58	0.05	0.12
0.38	-0.21	0.56	0.31	0.61	-0.13	-0.07
0.38	0.42	0.04	0.31	-0.10	0.75	0.12
0.38	0.49	-0.02	0.31	-0.34	-0.63	-0.04
0.38	0.38	-0.13	-0.73	0.37	-0.02	-0.14
0.38	-0.33	-0.50	0.06	0.14	-0.10	0.69
0.38	-0.40	-0.43	0.14	-0.10	0.09	-0.69



We can simply apply a common clustering method (such as k-means) to obtain the eventual clusters.

Laplacian Matrix and Normalized Cut

- For **normalized cut**, we need a normalized version of the Laplacian matrix:

$$L_{norm} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

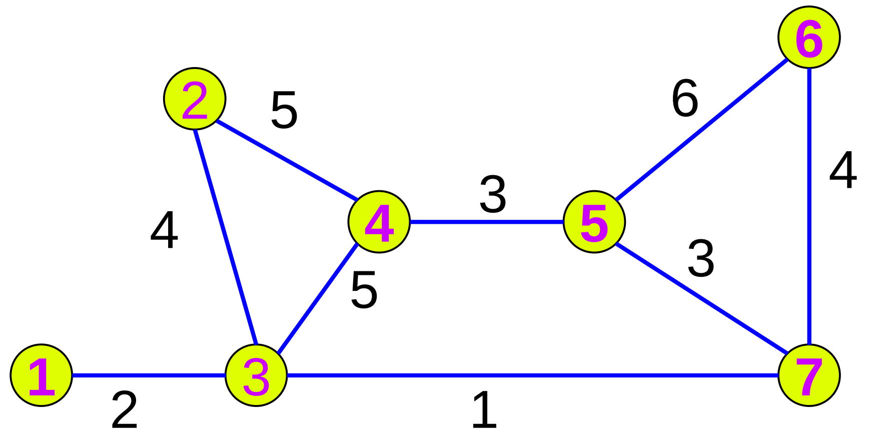
- Again, with the relaxed condition that allows real-valued basis vectors, we can simply select the orthonormal eigenvectors corresponding to the k smallest eigenvalues of L_{norm} to form the projection matrix. This minimizes

$$\mathrm{tr}(F^T L_{norm} F) \quad s.t. \quad F^T F = I$$

- We can then use any clustering algorithm with pre-specified number of clusters (e.g., k-means) in this new space to obtain the partition.

Bisection Spectral Clustering with Normalized Cut

Example:



L_{norm}

1.00	0	-0.41	0	0	0	0
0	1.00	-0.38	-0.46	0	0	0
-0.41	-0.38	1.00	-0.40	0	0	-0.10
0	-0.46	-0.40	1.00	-0.24	0	0
0	0	0	-0.24	1.00	-0.55	-0.31
0	0	0	0	-0.55	1.00	-0.45
0	0	-0.10	0	-0.31	-0.45	1.00

Eigenvector of the second smallest eigenvalue (with a single connected component, the first eigenvector is not useful):

0.21	0.38	0.41	0.31	-0.39	-0.49	-0.38

As an approximate “indicator vector”, we can use the signs of its elements to divide the vertices into two subsets.

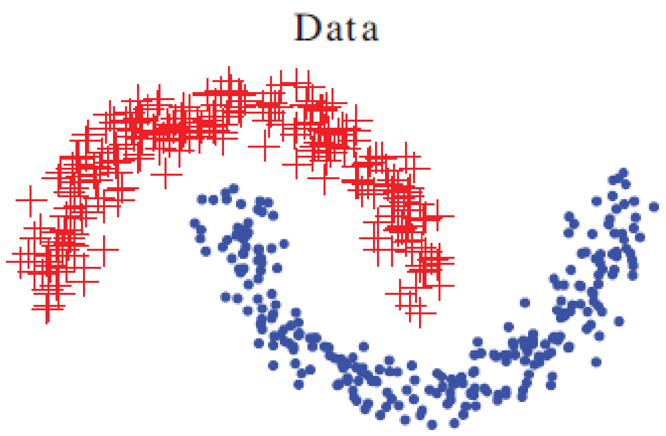
Building the Graph

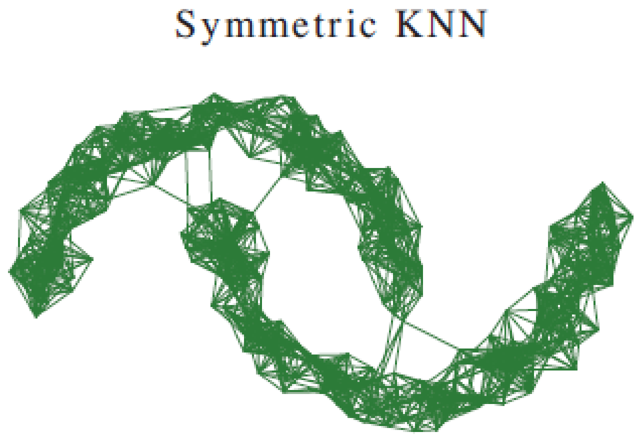
This step is needed if we start with data points in a feature space. This is a weighted graph with edge weights representing similarities between data points. Three typical methods are

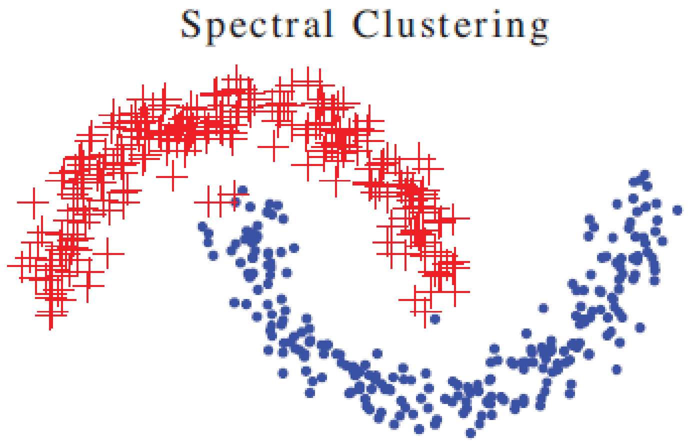
- k-nearest-neighbor graph (regular or symmetric)
- ε-neighborhood graph (edges based on distance thresholding)
- Fully connected graph (edges weighted by a function of distances)

Spectral Clustering with kNN Graphs

- This is an example result:







Additional Graph Partitioning Methods

- Agglomerative and divisive graph clustering (similar to hierarchical clustering, with samples as graph vertices)
- Graph growing (similar to region growing)
- Graph-cut refinement (iterative vertex pair swapping between clusters to minimize the cut)
- Multi-level graph partitioning
  - Multi-level graph coarsening (iterative edge collapsing)
  - Apply a standard graph-based clustering method.
  - Repeatedly "projecting up" to the original graph, possibly applying graph-cut refinement along the way.
  - The most well-known algorithm of this type is METIS and its derivatives.