Lecture 12: Unsupervised learning. Clustering

- Introduction to unsupervised learning
- Clustering problem
- K-means clustering
- Hierarchical clustering

Unsupervised learning

- In supervised learning, data is in the form of pairs $\langle \mathbf{x}, y \rangle$, where $y = f(\mathbf{x})$, and the goal is to approximate f well.
- In *unsupervised learning*, the data just contains $\mathbf{x}$!
- Goal is to "summarize" or find "patterns" or "structure" in the data
- A variety of problems and uses:
  - Clustering: "Flat" clustering or partitioning, hierarchical clustering
  - Density estimation
  - Dimensionality reduction, for: visualization, compression, pre-processing
- The definition of "ground truth" is often missing: no clear error function, or at least many reasonable alternatives
- Often useful in exploratory data analysis, and as a pre-processing step for supervised learning

What is clustering?

- Clustering is grouping similar objects together.
  - To establish prototypes, or detect outliers.
  - To simplify data for further analysis/learning.
  - To visualize data (in conjunction with dimensionality reduction)
- Clusterings are usually not "right" or "wrong" – different clusterings/clustering criteria can reveal different things about the data.
- Some clustering criteria/algorithms have natural probabilistic interpretations
- Clustering algorithms:
  - Employ some notion of distance between objects
  - Have an explicit or implicit criterion defining what a good cluster is
  - Heuristically optimize that criterion to determine the clustering

K-means clustering

- One of the most commonly-used clustering algorithms, because it is easy to implement and quick to run.
- Assumes the objects (instances) to be clustered are n-dimensional vectors, $\mathbf{x}_i$.
- Uses Euclidean distance
- The goal is to *partition* the data into K disjoint subsets

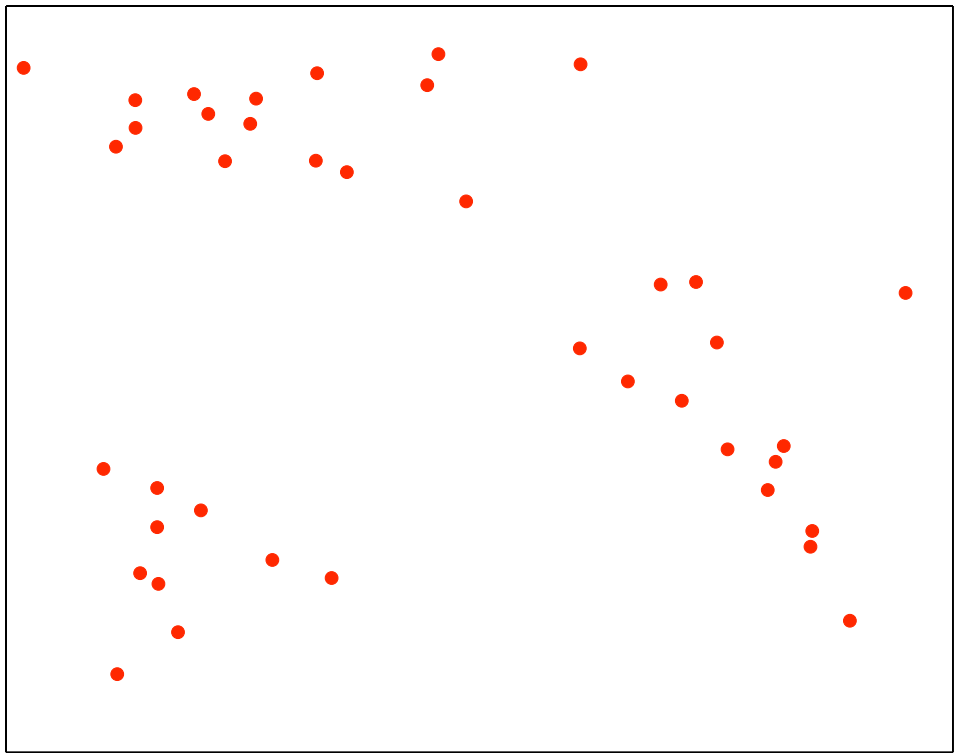
K-means clustering with real-valued data

- Inputs:
  - A set of n-dimensional real vectors $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$.
  - K, the desired number of clusters.
- Output: A mapping of the vectors into K clusters (disjoint subsets), $C : \{1, \ldots, m\} \mapsto \{1, \ldots, K\}$.

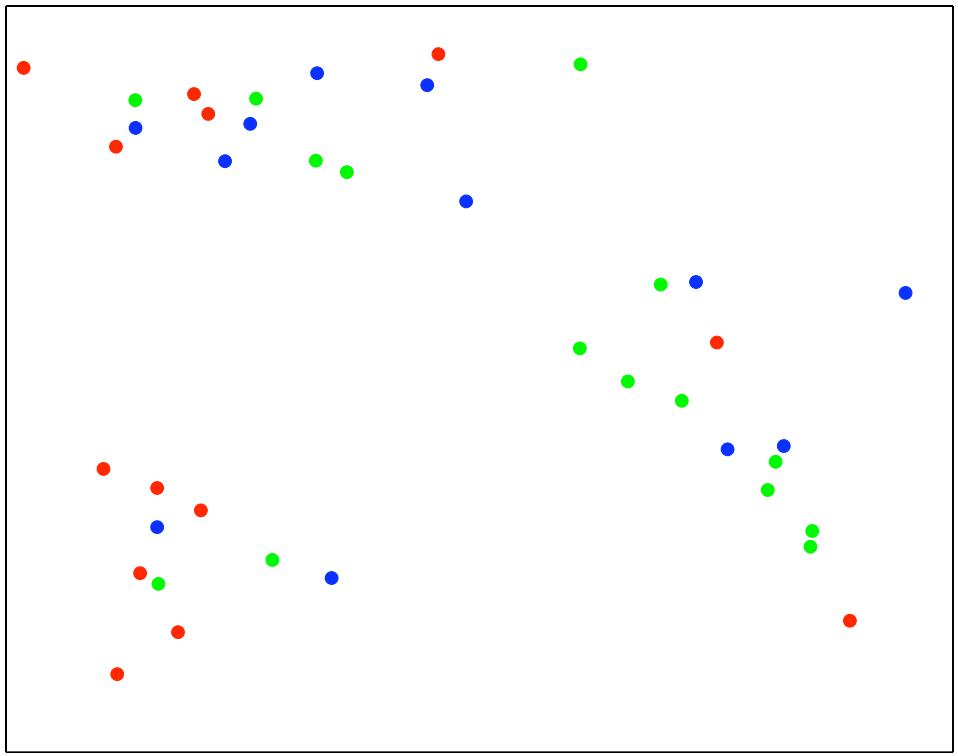
1. Initialize C randomly.
2. Repeat
   (a) Compute the *centroid* of each cluster (the mean of all the instances in the cluster)
   (b) Reassign each instance to the cluster with closest centroid

   until C stops changing.

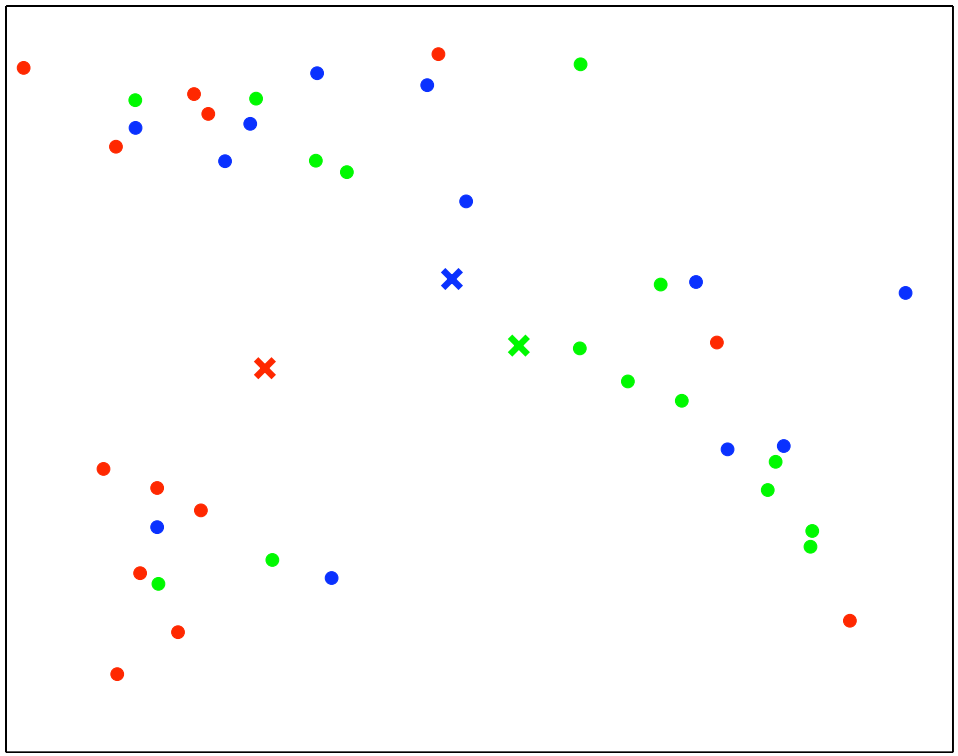
Example: initial data



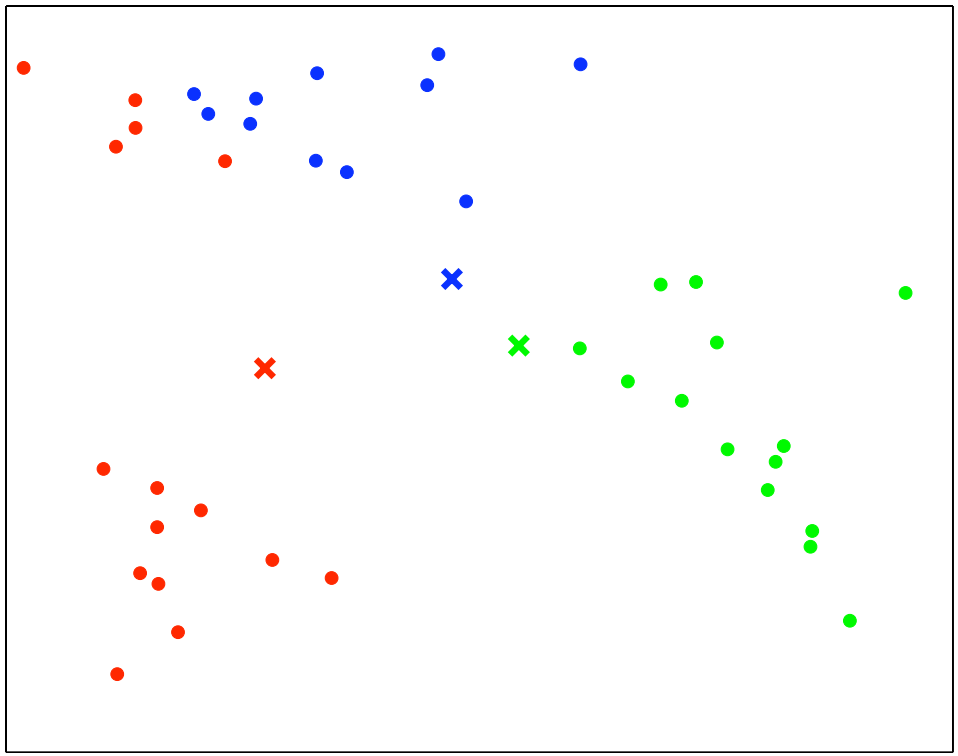
Example: assign into 3 clusters randomly



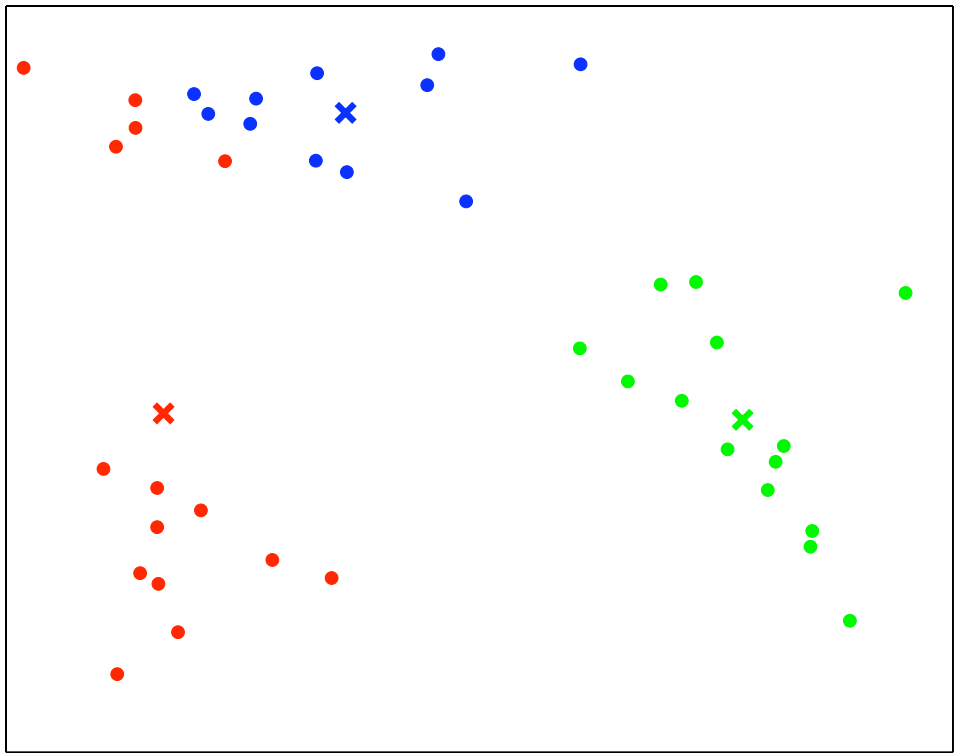
Example: compute centroids



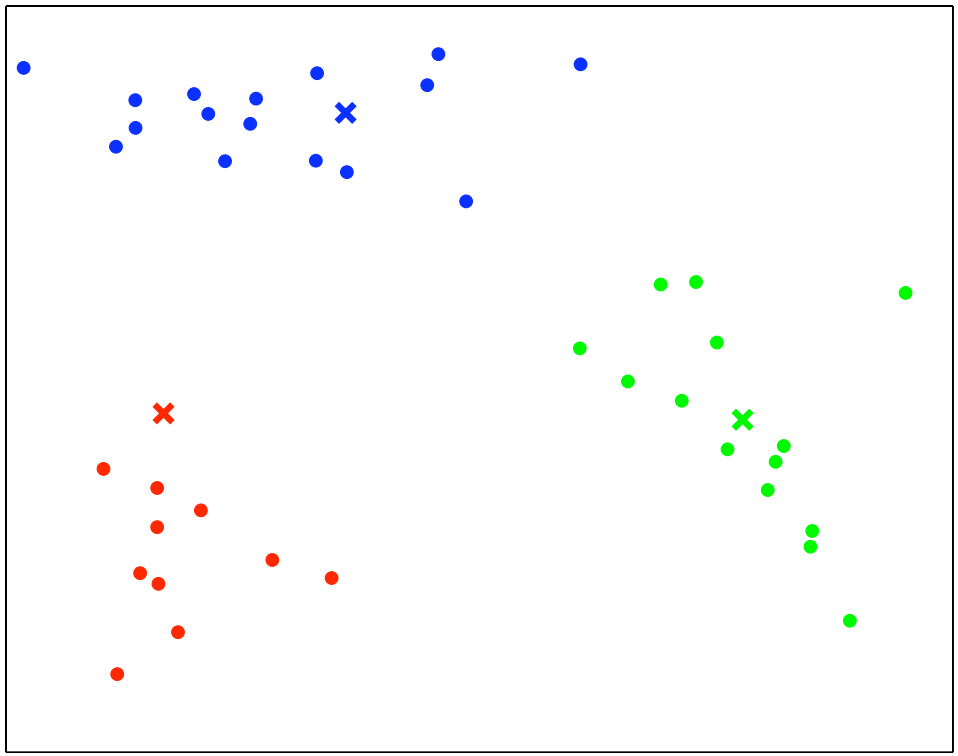
Example: reassign clusters



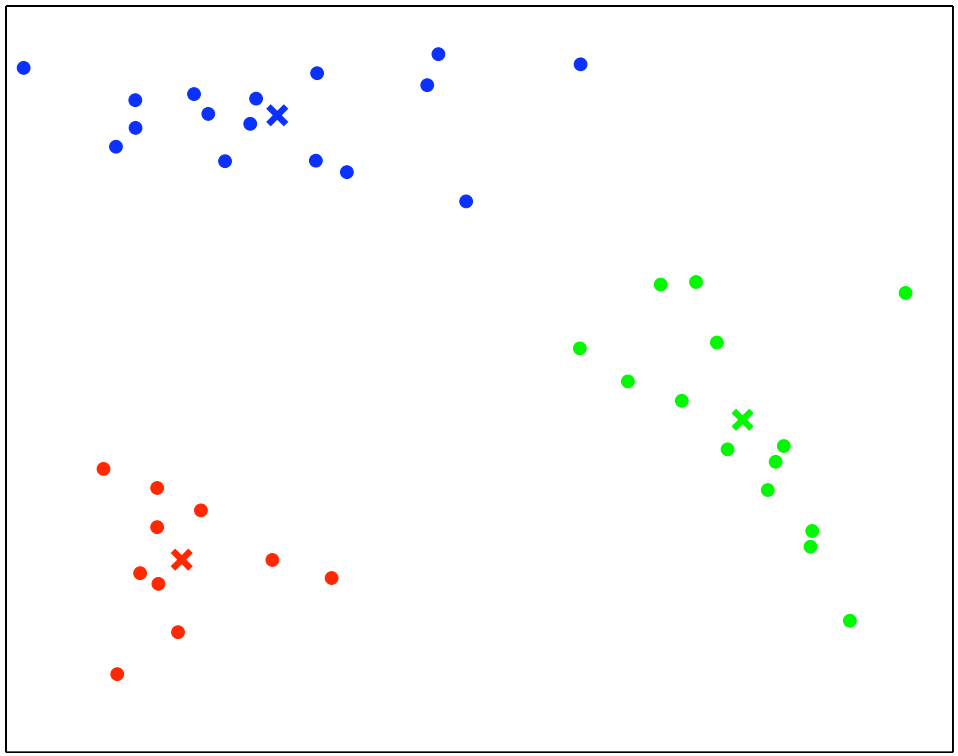
Example: recompute centroids



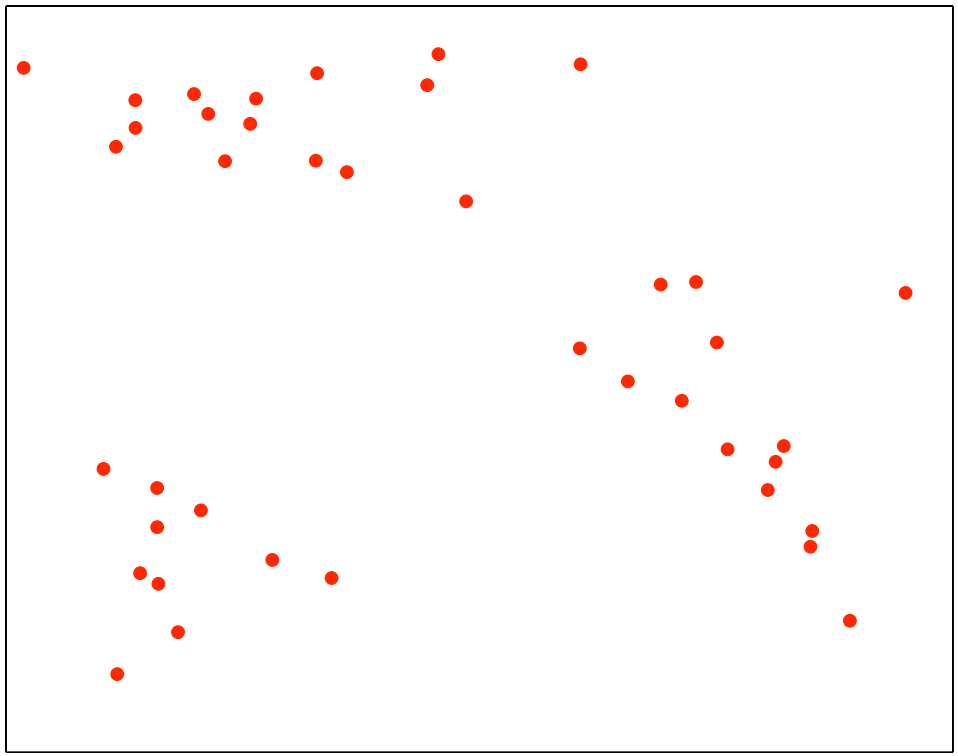
Example: reassign clusters



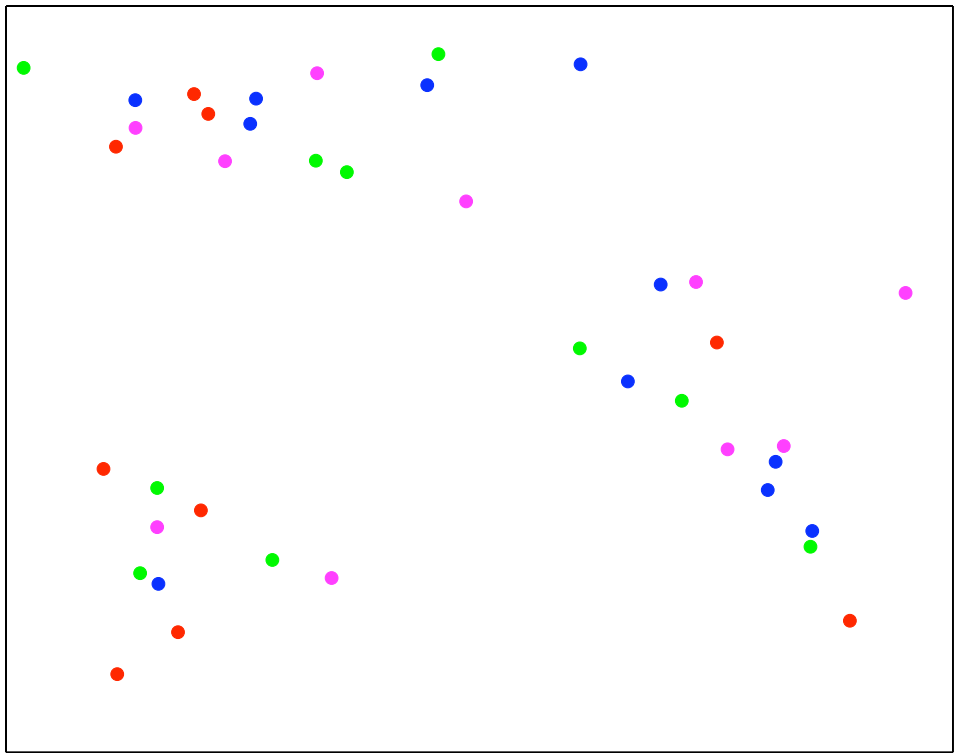
Example: recompute centroids – done!



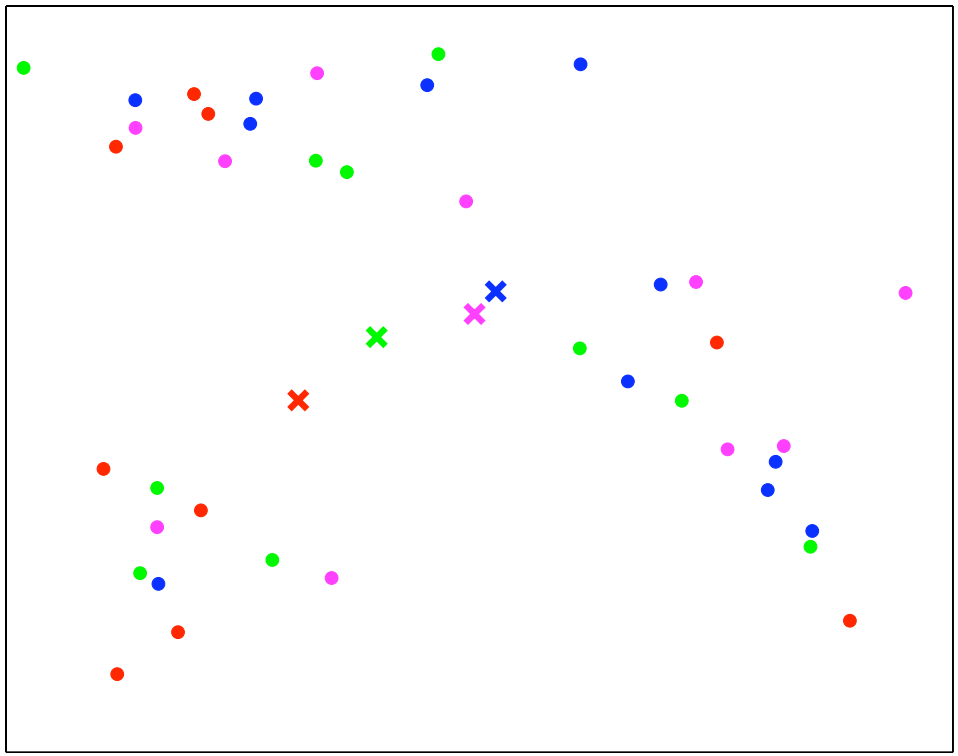
What if we do not know the right number of clusters?



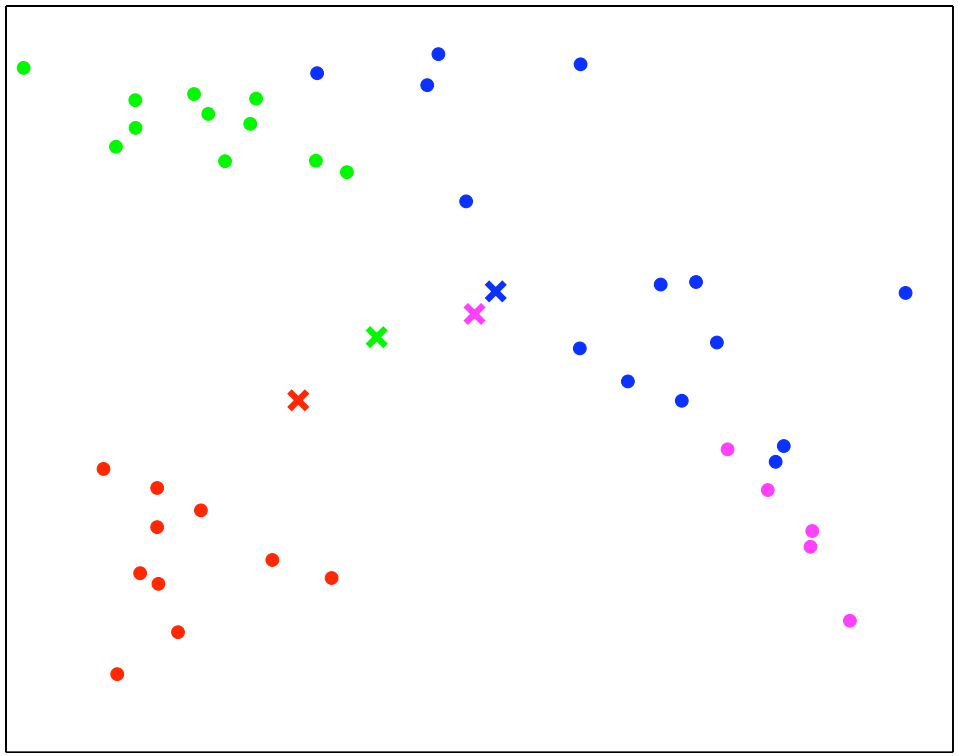
Example: assign into 4 clusters randomly



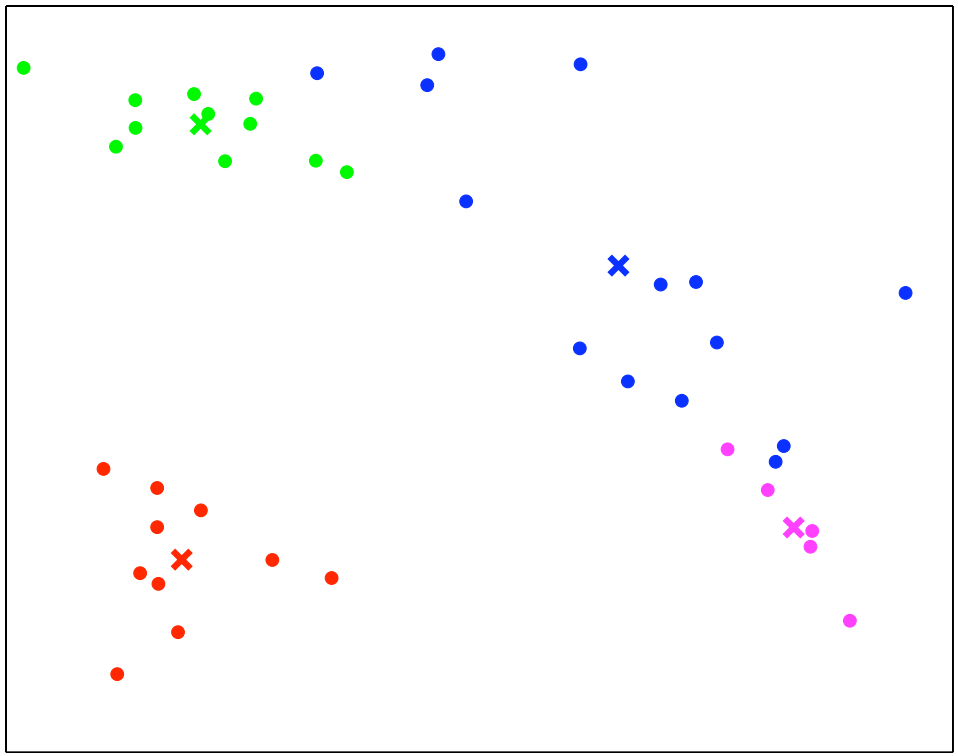
Example: compute centroids



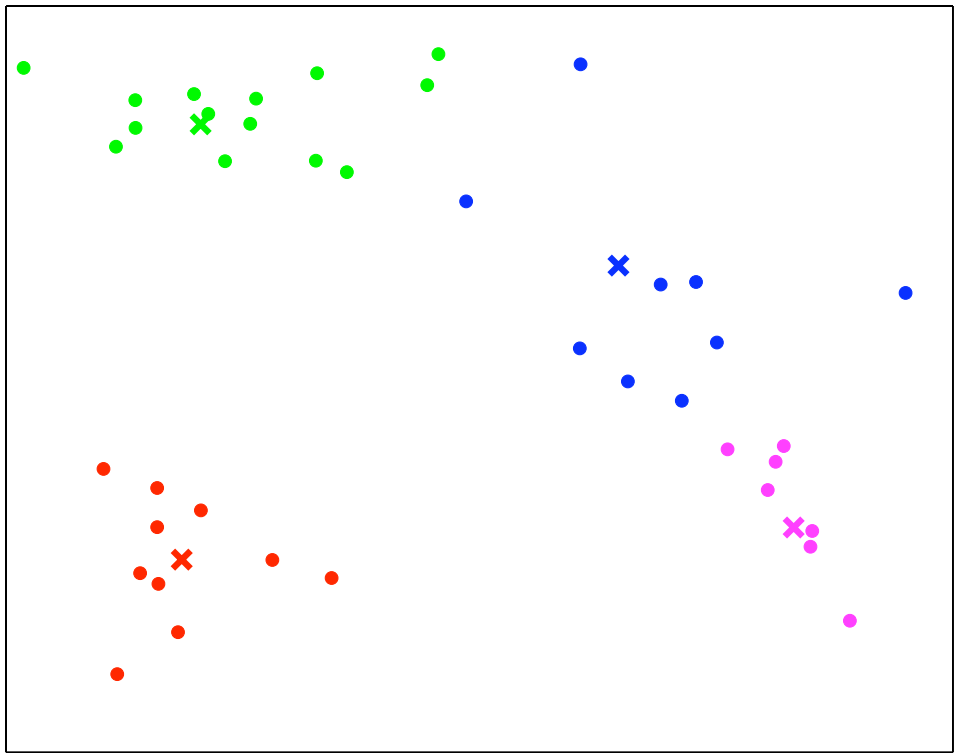
Example: reassign clusters



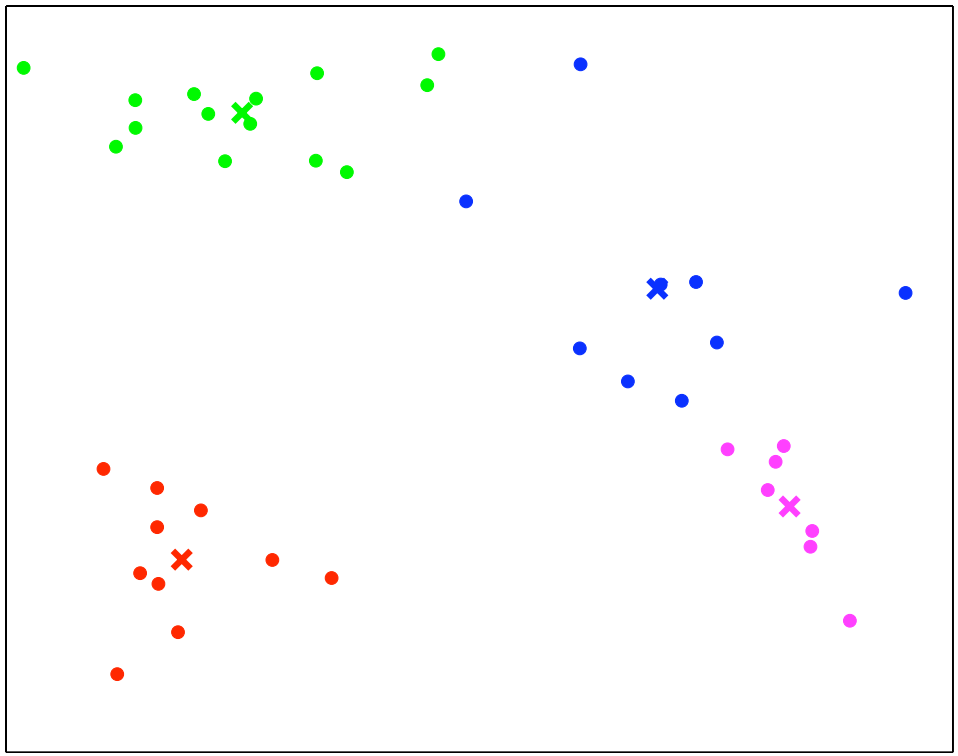
Example: recompute centroids



Example: reassign clusters



Example: recompute centroids – done!



Questions

- What is K-means trying to optimize?
- Will it terminate?
- Will it always find the same answer?
- How should we choose the initial cluster centers?
- Can we automatically choose the number of centers?

Does K-means clustering terminate?

- For given data $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and a clustering C, consider the sum of the squared Euclidian distance between each vector and the center of its cluster:

$$J = \sum_{i=1}^{m} \|\mathbf{x}_i - \mu_{C(i)}\|^2 ,$$

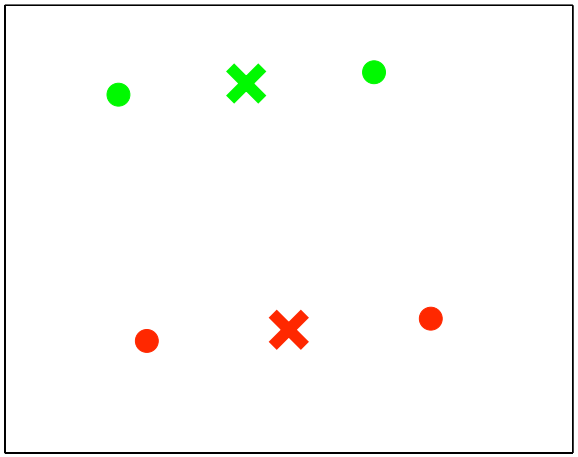
  where $\mu_{C(i)}$ denotes the centroid of the cluster containing $\mathbf{x}_i$.

- There are finitely many possible clusterings: at most K^m.
- Each time we reassign a vector to a cluster with a nearer centroid, J decreases.
- Each time we recompute the centroids of each cluster, J decreases (or stays the same.)
- Thus, the algorithm must terminate.

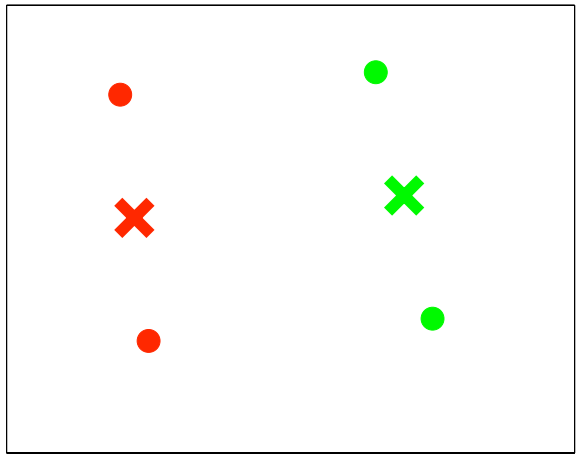
Does K-means always find the same answer?

- K-means is a version of coordinate descent, where the parameters are the cluster center coordinates, and the assignments of points to clusters.
- It minimizes the sum of squared Euclidean distances from vectors to their cluster centroid.
- This error function has many local minima!
- The solution found is *locally optimal*, but *not globally optimal*
- Because the solution depends on the initial assignment of instances to clusters, random restarts will give different solutions

Example



$J = 0.22870$



$J = 0.3088$

Example application: Color quantization

- Suppose you have an image stored with 24 bits per pixel
- You want to compress it so that you use only 8 bits per pixel (256 colors)
- You want the compressed image to look *as similar as possible* to the original image

$\Rightarrow$ Perform $K-$means clustering on the original set of color vectors with $K = 256$ colors.

  - Cluster centers (rounded to integer intensities) form the entries in the 256-color colormap
  - Each pixel repesented by 8-bit index into colormap

Example (Bishop)

$K = 2$ | $K = 3$ | $K = 10$ | Original image

More generally: Vector quantization with Euclidean loss

- Suppose we want to send all the instances over a communication channel
- In order to compress the message, we cluster the data and *encode each instance as the center of the cluster* to which it belongs
- The *reconstruction error* for real-valued data can be measured as Euclidian distance between the true value and its encoding
- An optimal (J-minimizing) K-means clustering minimizes the reconstruction error among all possible codings of the same type

Finding good initial configurations

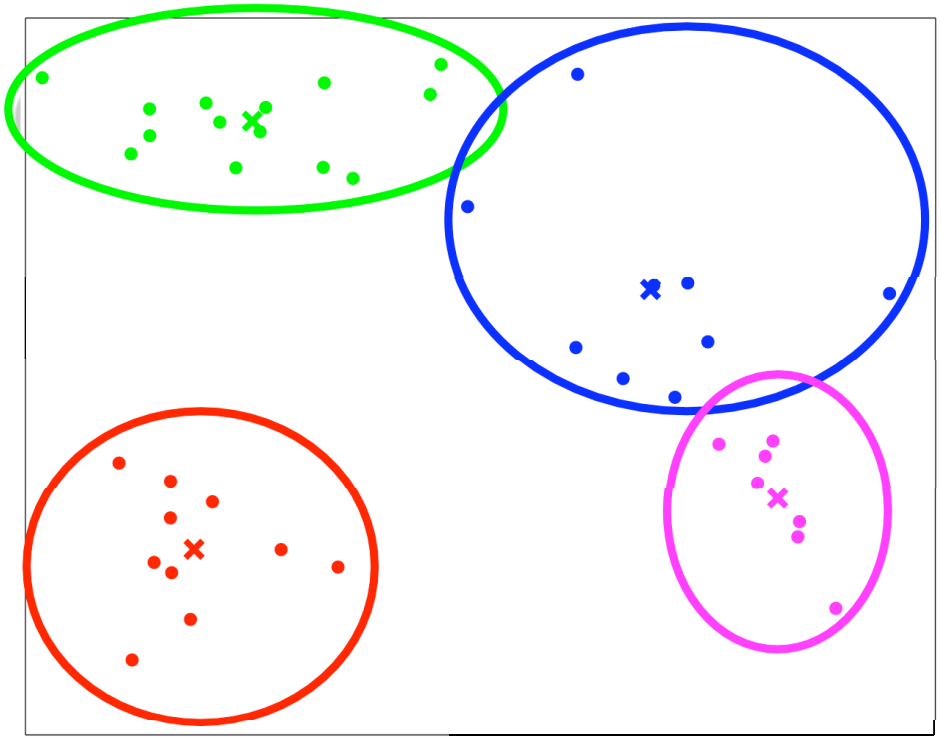
- The initial configuration can influence the final clustering
- Assigning each item to random cluster in $\{1, \ldots, K\}$ is unbiased. . . but typically results in cluster centroids near the centroid of all the data in the first round.
- A different heuristic tries to spread the initial centroids around as much as possible:
  - Place first center on top of a randomly chosen data point
  - Place second center on a data point as far away as possible from the first one
  - Place the i-th center as far away as possible from the closest of centers 1 through $i - 1$
- K-means clustering typically runs quickly. With a randomlized intialization step, you can run the algorithm multiple times and take the clustering with smallest J.

Choosing the number of clusters

- A difficult problem, ideas are floating around
- Delete clusters that cover too few points
- Split clusters that cover too many points
- Add extra clusters for “outliers”
- Minimum description length: minimize loss + complexity of the clustering
- Use a hierarchical method first (see in a bit)

Why the sum of squared Euclidean distances?

Subjective reason: It produces nice, round clusters.



Why the sum of squared Euclidean distances?

Objective reason: Maximum Likelihood Principle

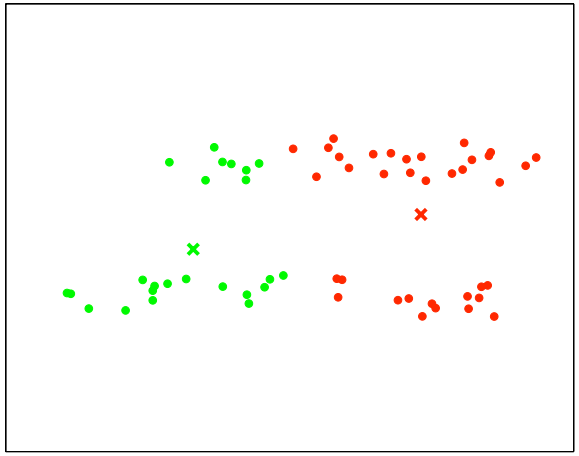
- Suppose the data really does divide into K clusters.
- Suppose the data in each cluster is generated by independent samples from a multivariate Gaussian distribution, where:
  - The mean of the Gaussian is the centroid of the cluster
  - The covariance matrix is of the form $\sigma^2 I$
- Then the probability of the data is highest when the sum of squared Euclidean distances is smallest.

Derivation: similar to MSE motivation in supervised learning

$$
\begin{aligned}
l(\mathbf{x}_1, \ldots, \mathbf{x}_m | C(i), \mu_j) &= \prod_{i=1}^{m} l(\mathbf{x}_i | C(i), \mu_j) \\
&= \prod_{i=1}^{m} \frac{1}{(2\pi)^{n/2}\sigma^n} \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mu_{C(i)}\|^2\right) \\
\log l(\mathbf{x}_1, \ldots, \mathbf{x}_m | C(i), \mu_j) &\propto -\sum_{i=1}^{m} \|\mathbf{x}_i - \mu_{C(i)}\|^2 = J
\end{aligned}
$$

Why *not* the sum of squared Euclidean distances?

1. It produces nice round clusters!



2. Differently scaled axes can dramatically affect results.
3. There may be symbolic attributes, which have to be treated differently

K-means-like clustering in general

- Given a set of objects (need not be real vectors),
  - Choose a notion of pairwise distance / similarity between the objects.
  - Choose a scoring function for the clustering
  - Optimize the scoring function, to find a good clustering.
- For most choices, the optimization problem will be intractable. Local optimization is often necessary.

Distance metrics

- Euclidean distance
- Hamming distance (number of mismatches between two strings)
- Travel distance along a manifold (e.g. for geographic points)
- Tempo / rhythm similarity (for songs)
- Shared keywords (for web pages), or shared in-links
- . . .

Scoring functions

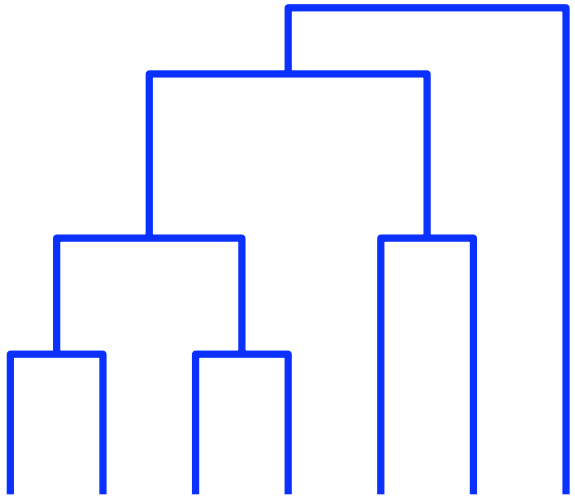
- Minimize: Summed distances between all pairs of objects in the same cluster. (Also known as ”within-cluster scatter.”)
- Minimize: Maximum distance between any two objects in the same cluster. (Can be hard to optimize.)
- Maximize: Minimum distance between any two objects in different clusters.

Common uses of K-means

- Often used in exploratory data analysis
- Often used as a pre-processing step before supervised learning
- In one-dimension, it is a good way to discretize real-valued variables into non-uniform buckets
- Used in speech understanding/recognition to convert wave forms into one of k categories (vector quantization)

Hierarchical clustering

- Organizes data instances into trees.
- For visualization, exploratory data analysis.
- *Agglomerative methods* build the tree bottom-up, successively grouping together the clusters deemed most similar.
- *Divisive methods* build the tree top-down, recursively partitioning the data.

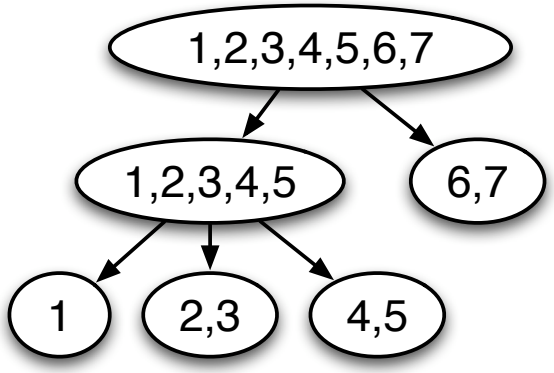


What is a hierarchical clustering?

- Given instances $D = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$.
- A hierarchical clustering is a set of subsets (clusters) of D, $C = \{C_1, \ldots, C_K\}$, where
  - Every element in D is in at least one set of C (the root)
  - The C_j can be assigned to the nodes of a tree such that the cluster at any node is precisely the union of the clusters at the node's children (if any).

Example of a hierarchical clustering

- Suppose $D = \{1, 2, 3, 4, 5, 6, 7\}$. A hierarchical clustering is $C = \{\{1\}, \{2, 3\}, \{4, 5\}, \{1, 2, 3, 4, 5\}, \{6, 7\}, \{1, 2, 3, 4, 5, 6, 7\}\}$.



- In this example:
  - Leaves of the tree need not correspond to single instances.
  - The branching factor of the tree is not limited.
- However, most hierarchical clustering algorithms produce binary trees, and take single instances as the smallest clusters.

Agglomerative clustering

- Input: Pairwise distances $d(\mathbf{x}, \mathbf{x}')$ between a set of data objects $\{\mathbf{x}_i\}$.
- Output: A hierarchical clustering
- Algorithm:
  1. Assign each instance as its own cluster on a working list W.
  2. Repeat
     (a) Find the two clusters in W that are most "similar".
     (b) Remove them from W.
     (c) Add their union to W.
     until W contains a single cluster with all the data objects.
  3. Return *all clusters* appearing in W at any stage of the algorithm.

How many clusters?

- How many clusters are generated by the agglomerative clustering algorithm?
- Answer: $2m - 1$, where m is the number of data objects.
- Why? A binary tree with m leaves has $m - 1$ internal nodes, thus $2m - 1$ nodes total.
- More explicitly:
  - The working list W starts with m singleton clusters
  - Each iteration removes two clusters from W and adds one new one
  - The algorithm stops when W has one cluster, which is after $m - 1$ iterations

How do we measure dissimilarity between clusters?

- Distance between nearest objects ("Single-linkage" agglomerative clustering, or "nearest neighbor"):

$$\min_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

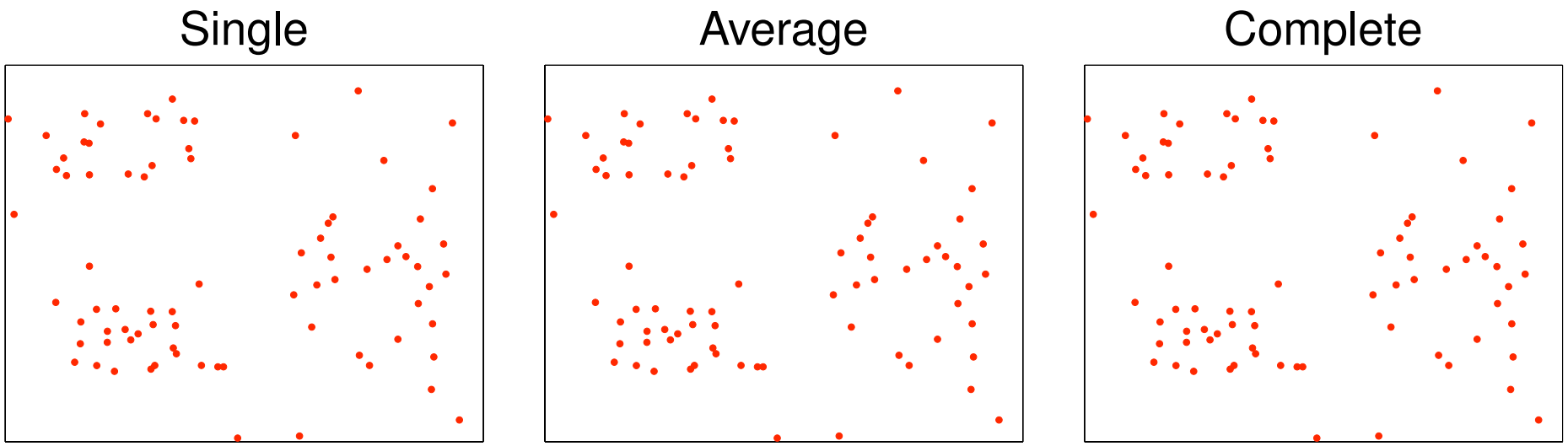
- Distance between farthest objects ("Complete-linkage" agglomerative clustering, or "furthest neighbor"):

$$\max_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

- Average distance between objects ("Group-average" agglomerative clustering):

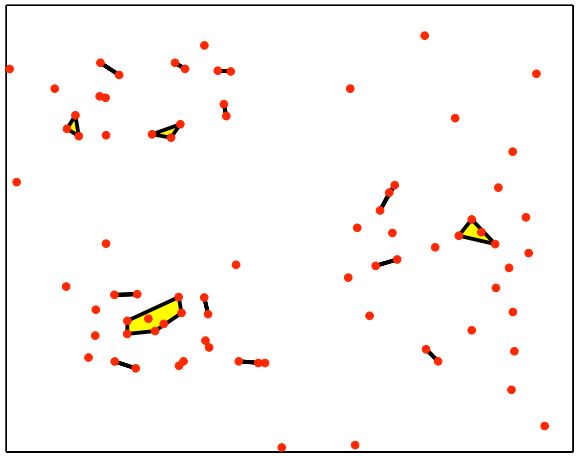
$$\frac{1}{|C||C'|} \sum_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

Example 1: Data

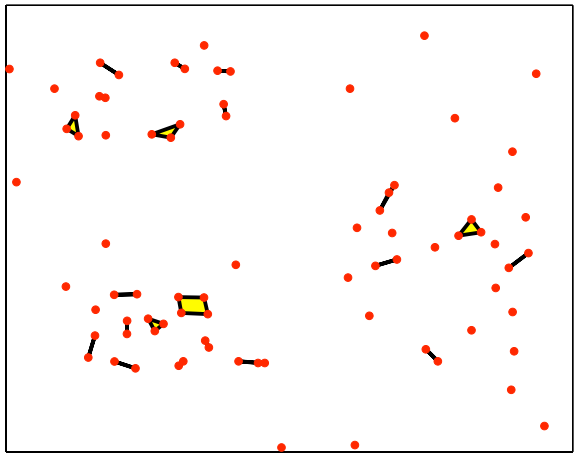


Example 1: Iteration 30

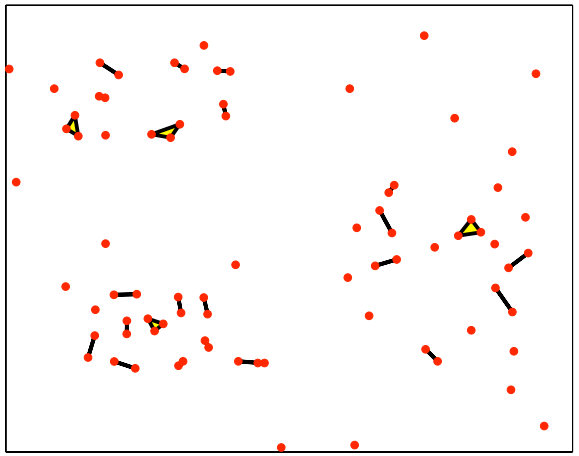
Single



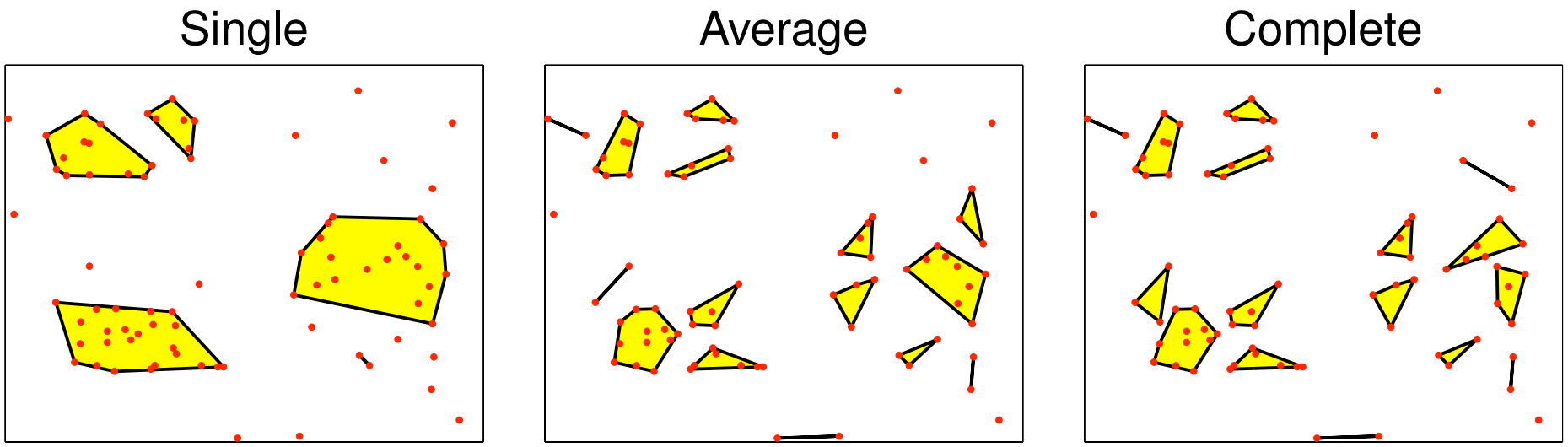
Average



Complete



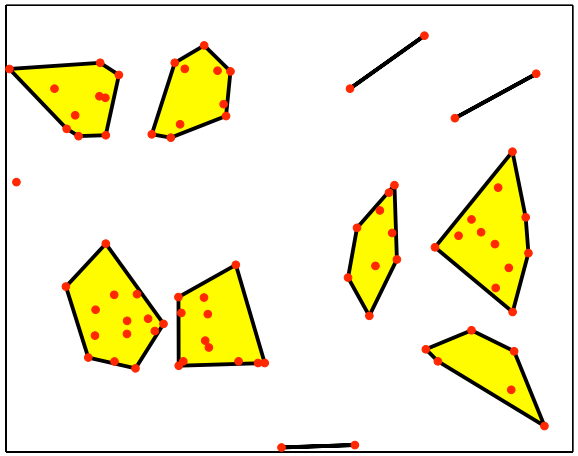
Example 1: Iteration 60



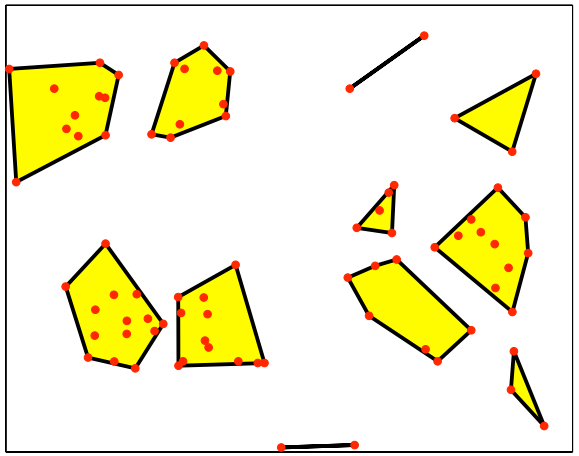
Example 1: Iteration 70

Single

Average

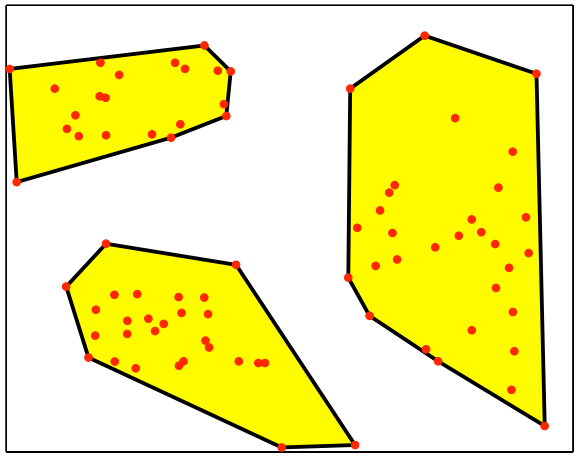


Complete

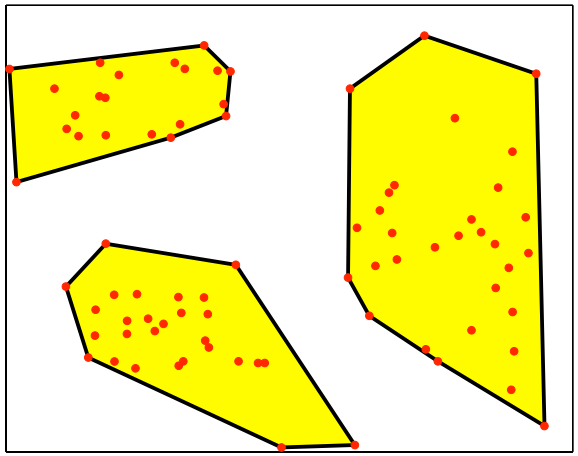


Example 1: Iteration 78

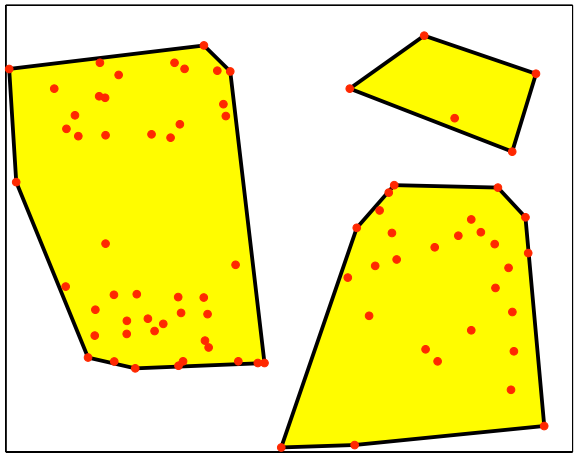
Single



Average

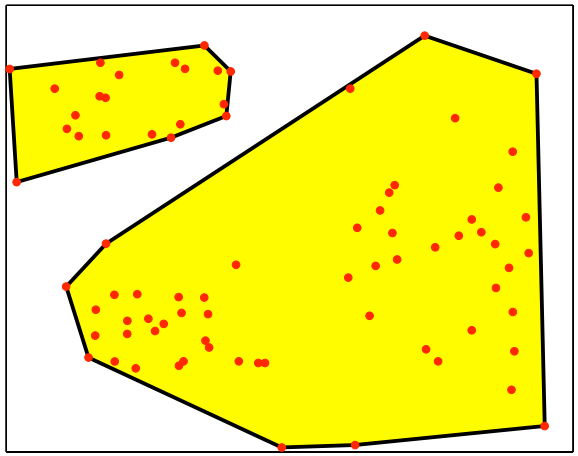


Complete



Example 1: Iteration 79

Single

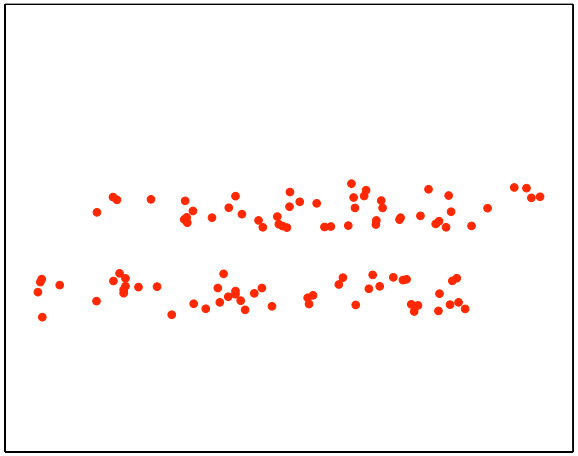


Average

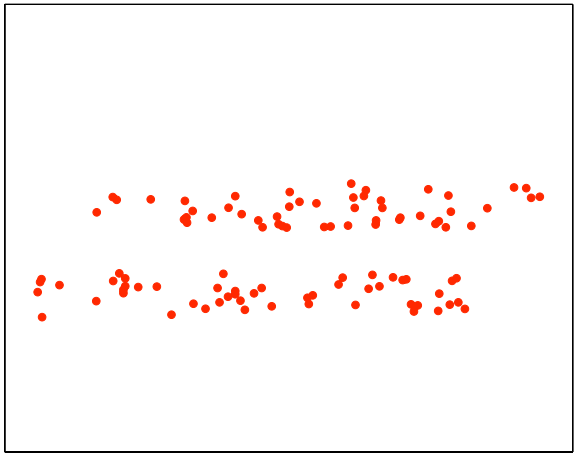
Complete

Example 2: Data

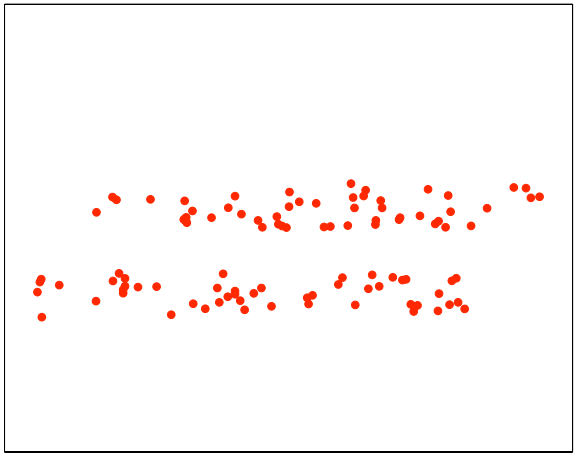
Single



Average

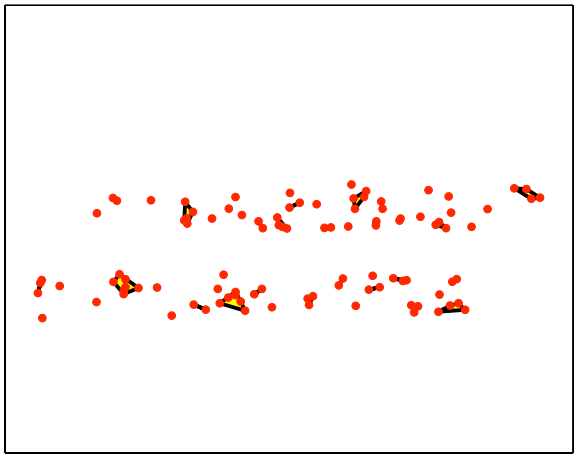


Complete

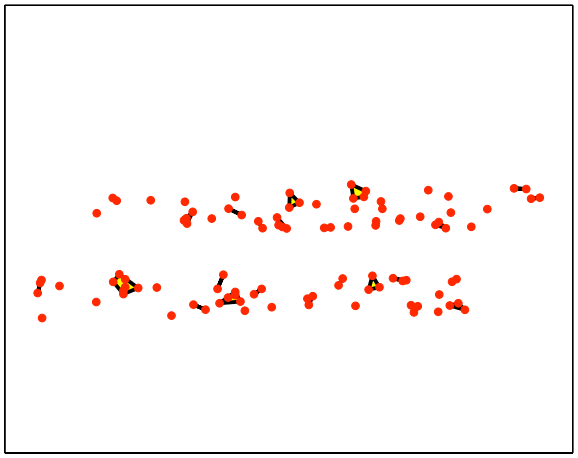


Example 2: Iteration 50

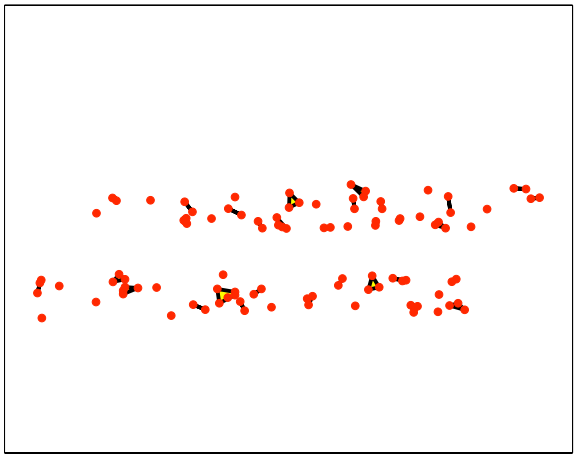
Single



Average

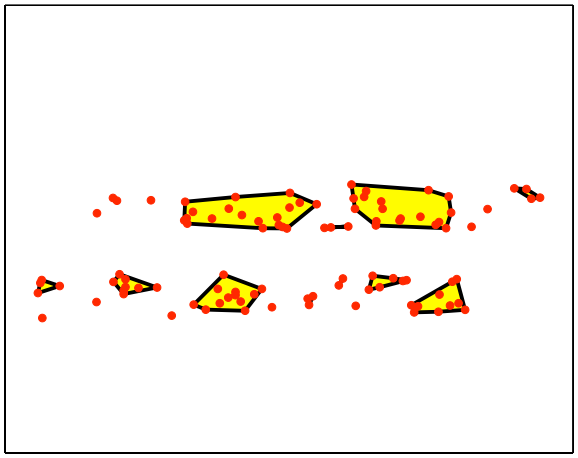


Complete

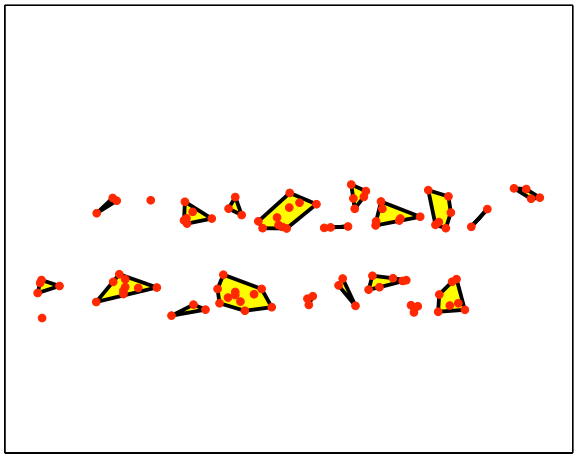


Example 2: Iteration 80

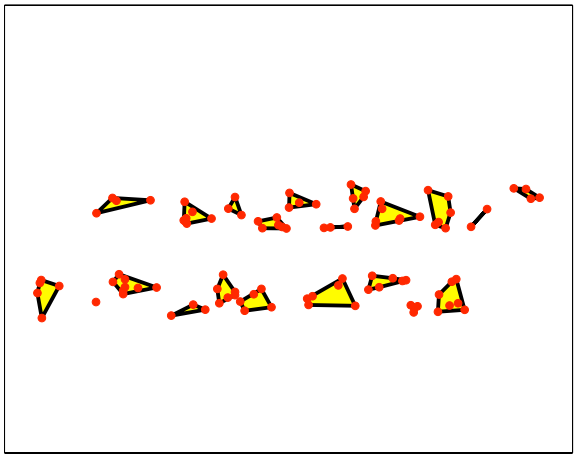
Single



Average

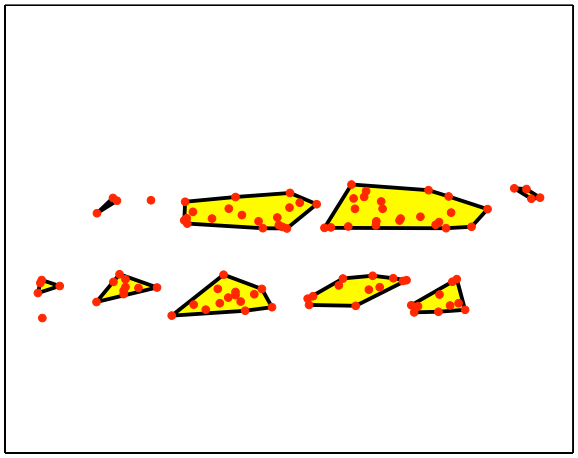


Complete

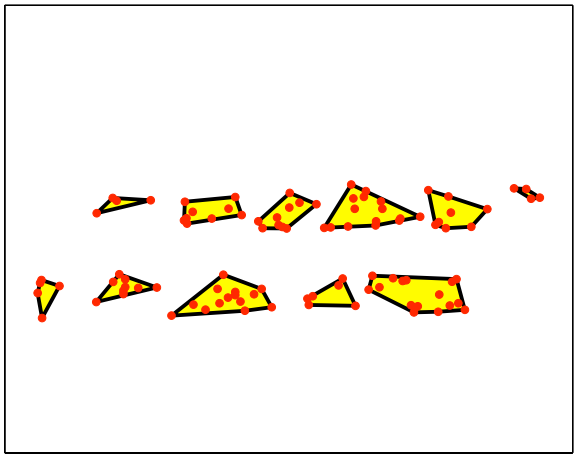


Example 2: Iteration 90

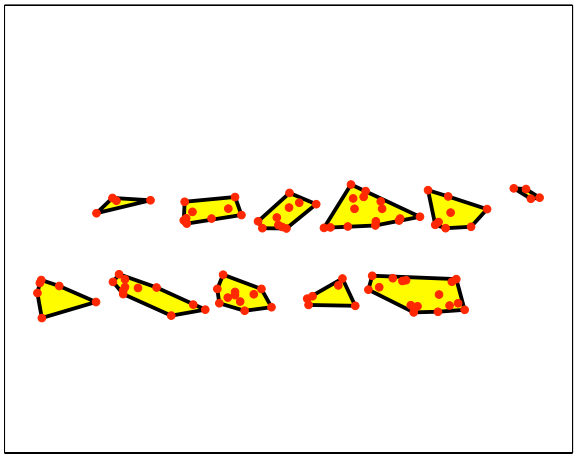
Single



Average

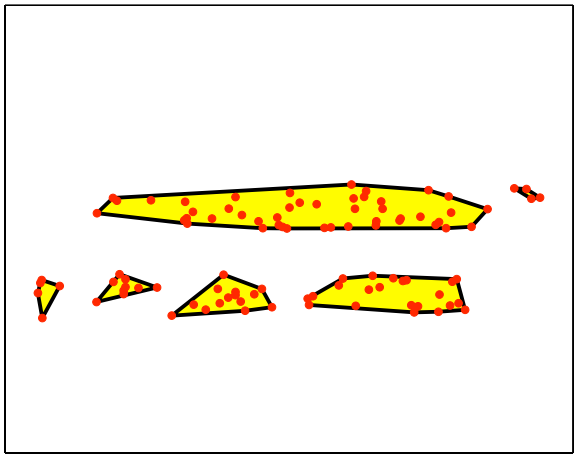


Complete

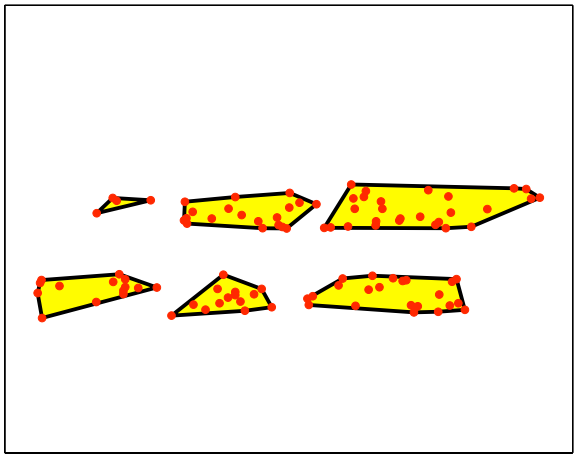


Example 2: Iteration 95

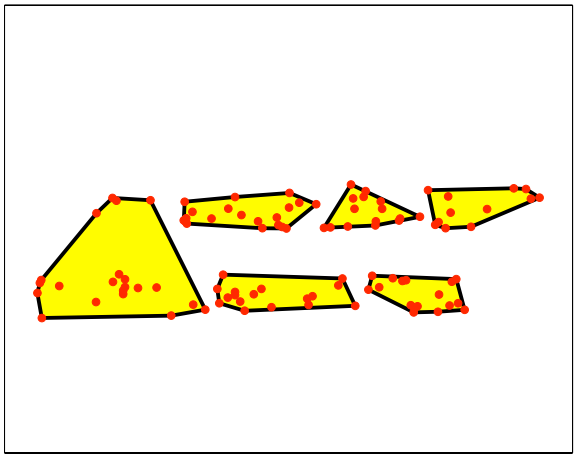
Single



Average

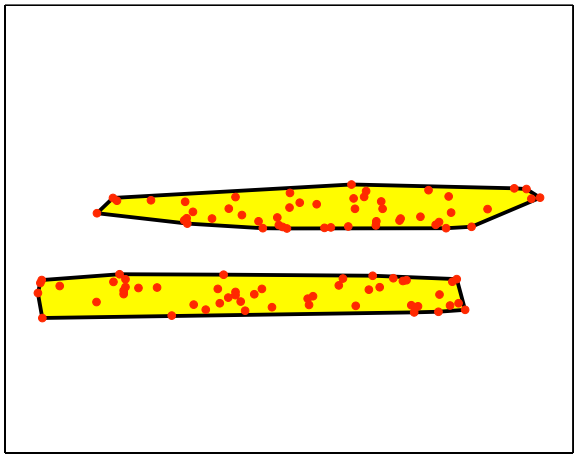


Complete

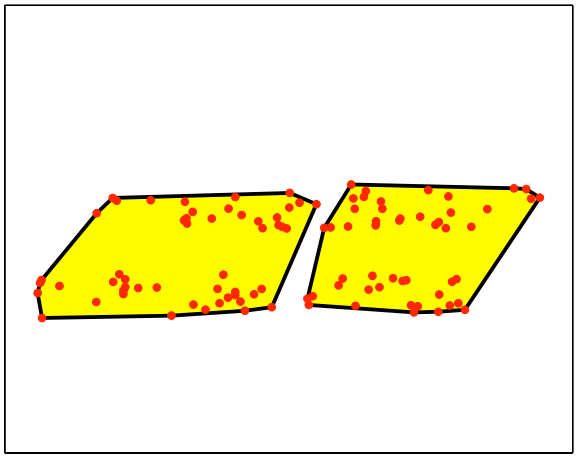


Example 2: Iteration 99

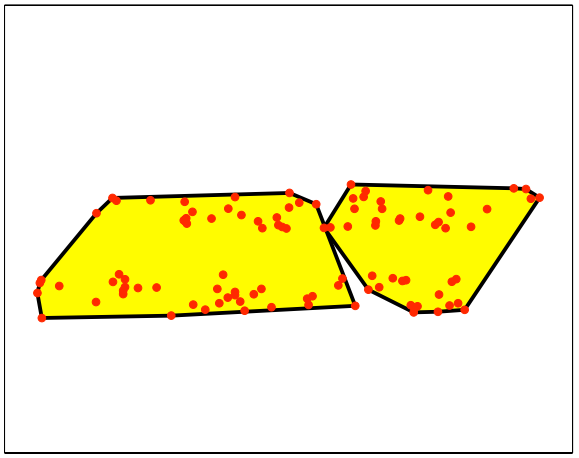
Single



Average



Complete

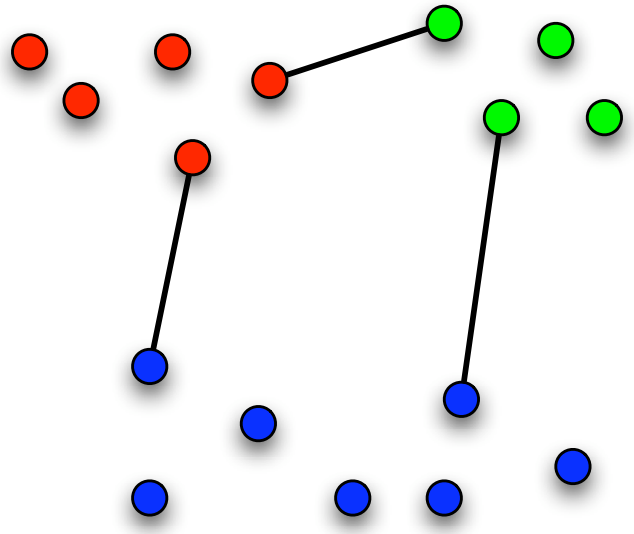


Intuitions about cluster similarity

- Single-linkage
  - Favors spatially-extended / filamentous clusters
  - Often leaves singleton clusters until near the end
- Complete-linkage favors compact clusters
- Average-linkage is somewhere in between

Monotonicity

- Let A, B, C be clusters.
- Let d be one of the dissimilarity measures: single-linkage (see below), average linkage or complete linkage
- If $d(A, B) \leq d(A, C)$ and $d(A, B) \leq d(B, C)$, then $d(A, B) \leq d(A \cup B, C)$.



Monotonicity of single-linkage criterion: Proof

- Suppose that that $d(A,B) \leq d(A,C)$ and $d(A,B) \leq d(B,C)$
- Then:

$$\begin{aligned} d(A \cup B, C) &= \min_{x \in A \cup B, x' \in C} d(x, x') \\ &= \min \left(\min_{x_a \in A, x' \in C} d(x_a, x'), \min_{x_b \in B, x' \in C} d(x_a, x') \right) \\ &= \min \left(d(A,C), d(B,C) \right) \\ &\geq \min \left(d(A,B), d(A,B) \right) \\ &= d(A,B) \end{aligned}$$

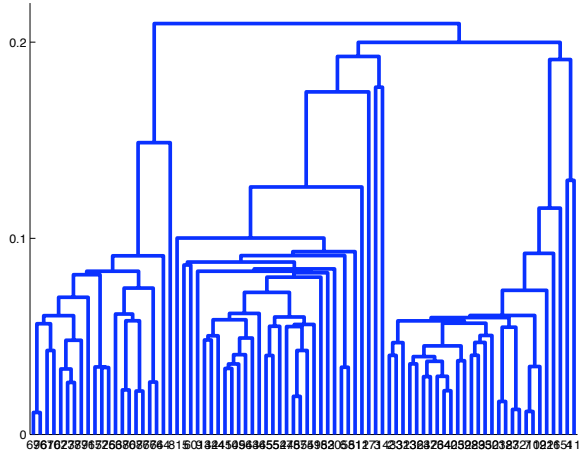
- Proofs for group-average and complete-linkage are similar.

Dendrograms

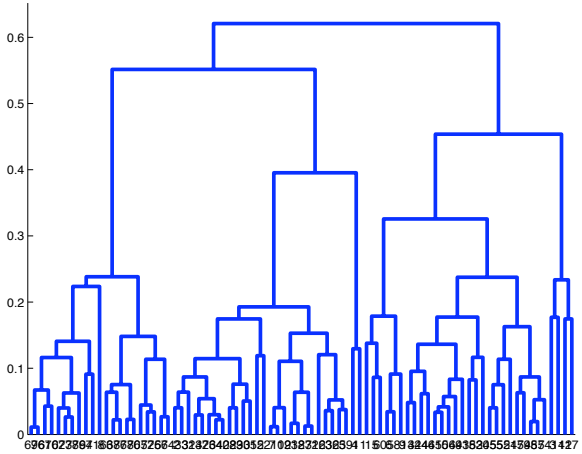
- The monotonicity property implies that every time agglomerative clustering merges two clusters, the dissimilarity of those clusters is $\geq$ the dissimilarity of all previous merges.
- Dendrograms (trees depicting hierarchical clusterings) are often drawn so that the height of a node corresponds to the dissimilarity of the merged clusters.
- We can form a flat clustering by cutting the tree at any height.
- Jumps in the height of the dendrogram can suggest natural cutoffs.

Dendrograms for Example 1

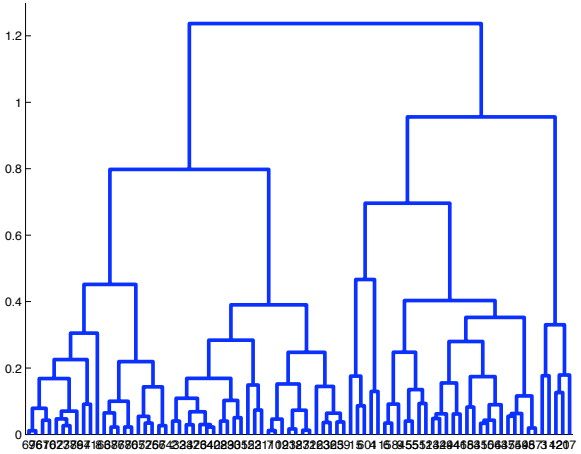
Single



Average

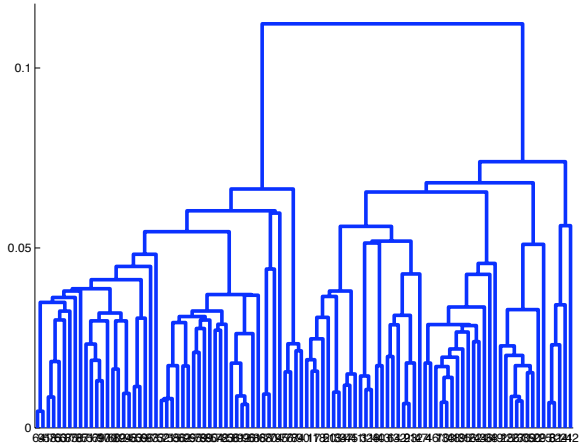


Complete

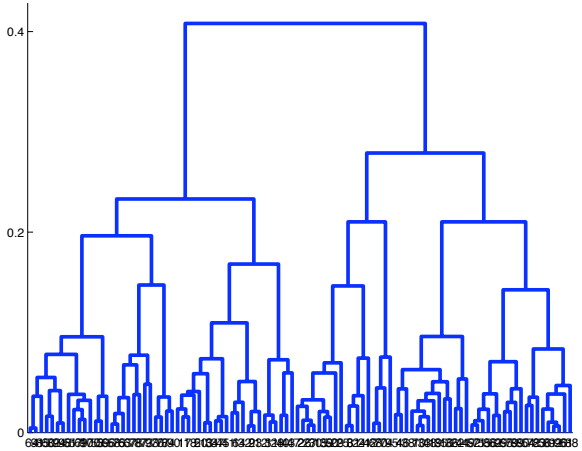


Dendrograms for Example 2

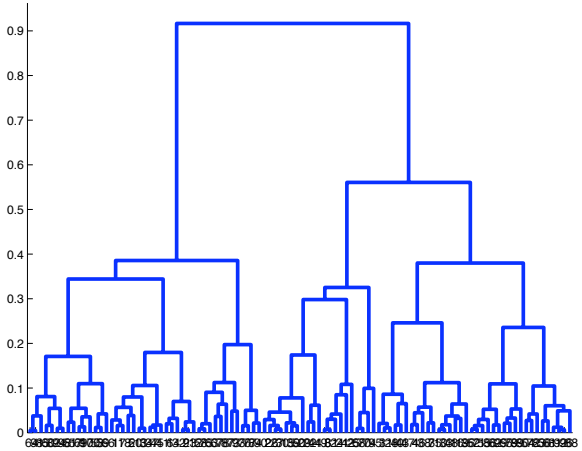
Single



Average



Complete



Divisive clustering

- Works by recursively partitioning the instances.
- How might you do that?
  - K-means?
  - Max weighted cut on graph where edges are weighted by pairwise distances?
  - Maximum margin?
- Many heuristics for partitioning the instances have been proposed . . . but many are computationally hard and/or violate monotonicity, making it hard to draw dendrograms.

Summary

- K-means
  - Fast way of partitioning data into K clusters
  - It minimizes the sum of squared Euclidean distances to the clusters centroids
  - Different clusterings can result from different initializations
  - Can be interpreted as fitting a mixture distribution
- Hierarchical clustering
  - Organizes data objects into a tree based on similarity.
  - Agglomerative (bottom-up) tree construction is most popular.
  - There are several choices of distance metric (linkage criterion)
  - Monotonicity allows us to draw dendrograms in which the height of a node corresponds to the dissimilarity of the clusters merged.
  - Trees can be cut off at some level, to generate a flat partitioning of the data.