CS 886
Applied Machine Learning
Clustering

Dan Lizotte

University of Waterloo

30 May 2012

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 or similarity 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, at least for small dimension.
- Assumes the objects (instances) to be clustered are p-dimensional vectors, $\mathbf{x}_i$.
- Uses a distance measure between the instances (typically Euclidian distance)
- The goal is to *partition* the data into K disjoint subsets

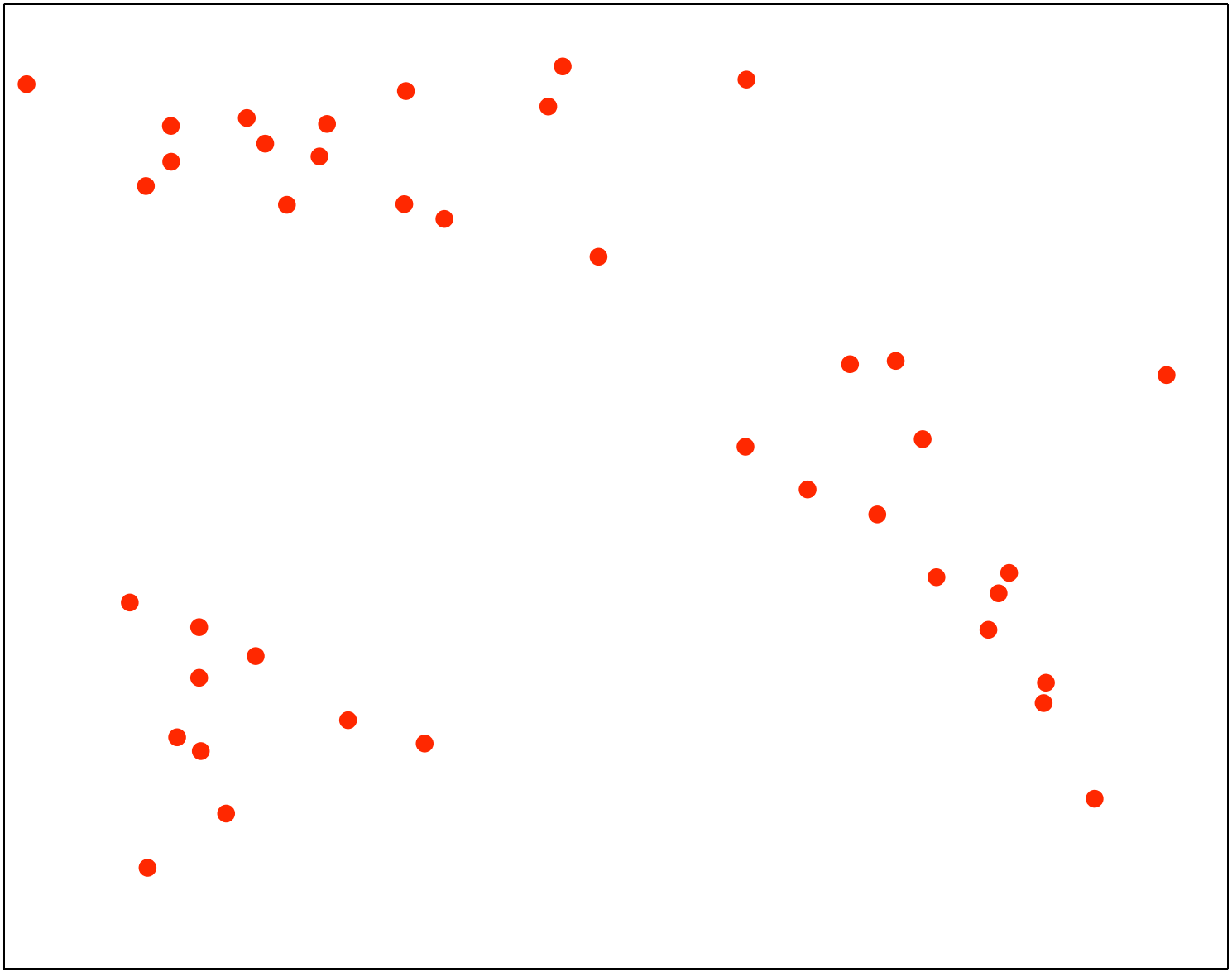
K-means clustering with real-valued data

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

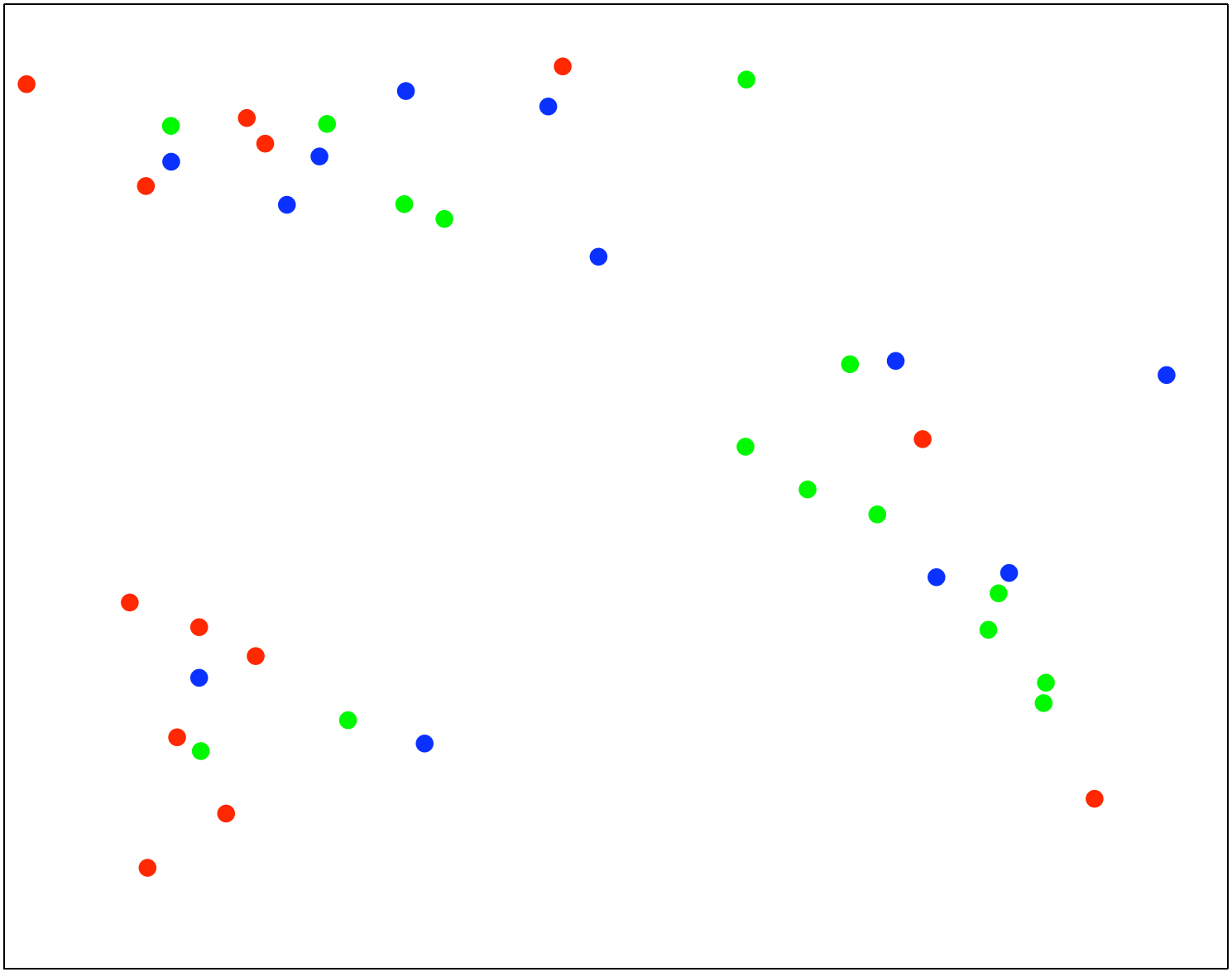
1. Initialize C randomly.
2. Repeat
   1. Compute the *centroid* of each cluster (the mean of all the instances in the cluster)
   2. Reassign each instance to the cluster with closest centroid

   until C stops changing.

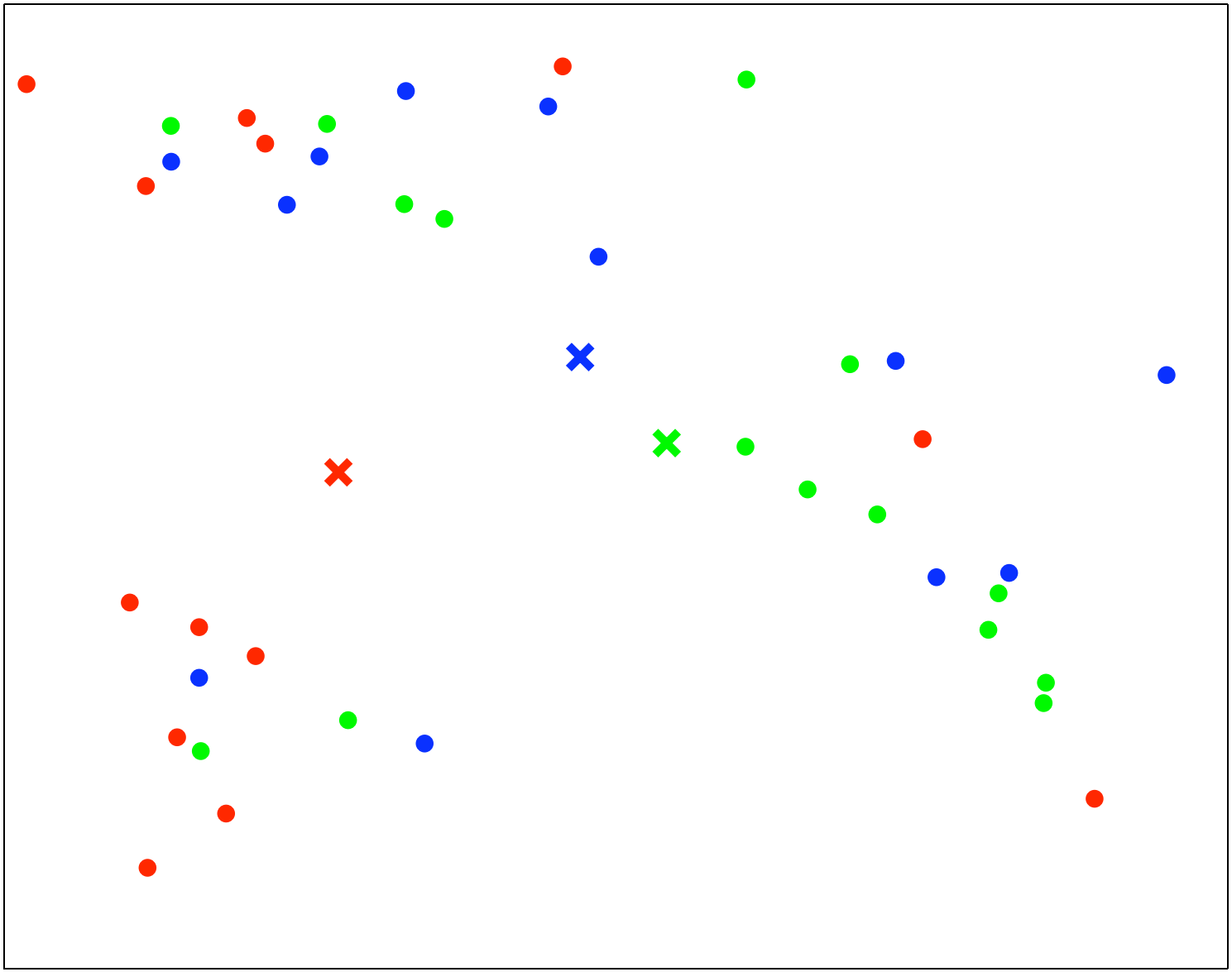
Example: initial data



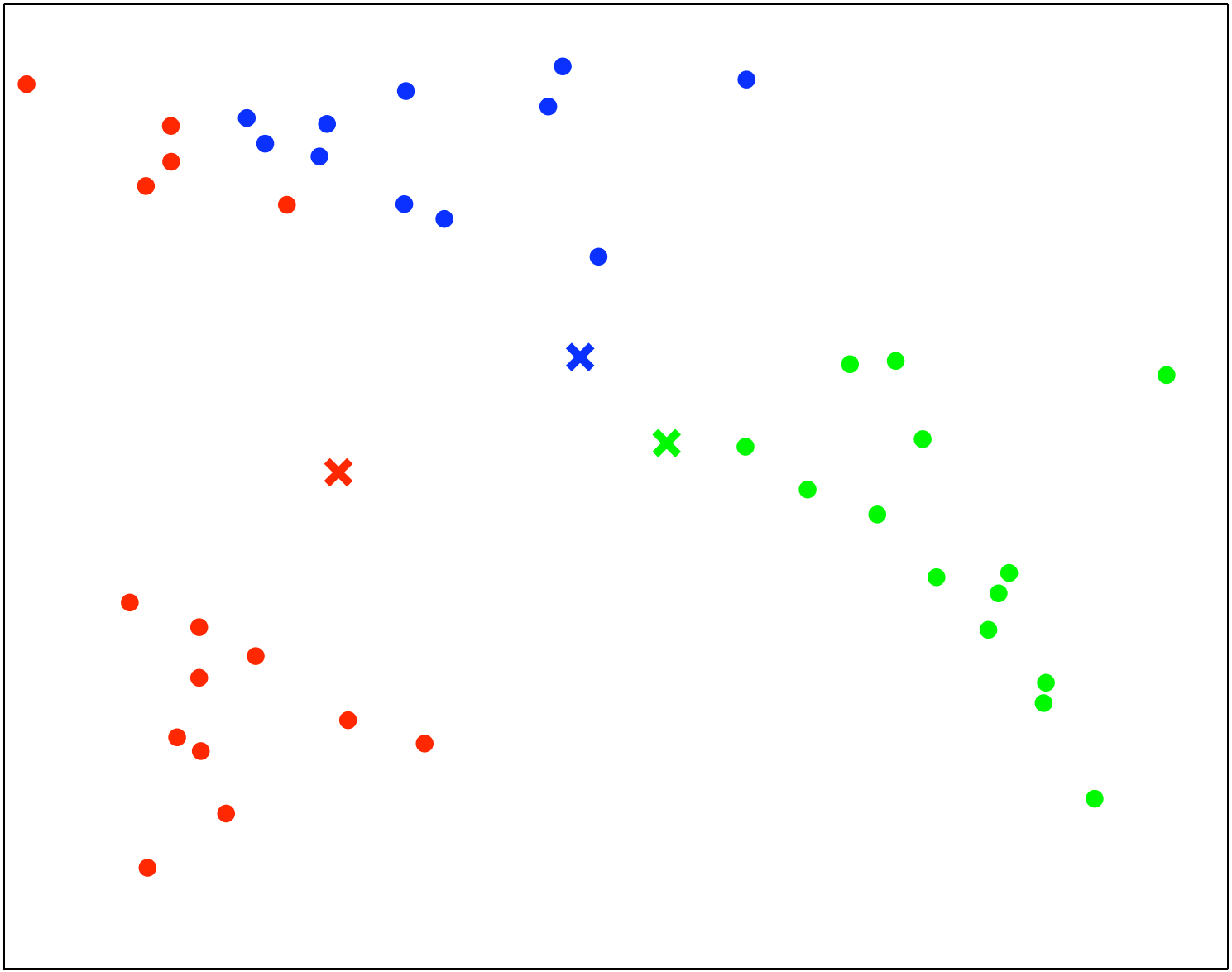
Example: assign into 3 clusters randomly



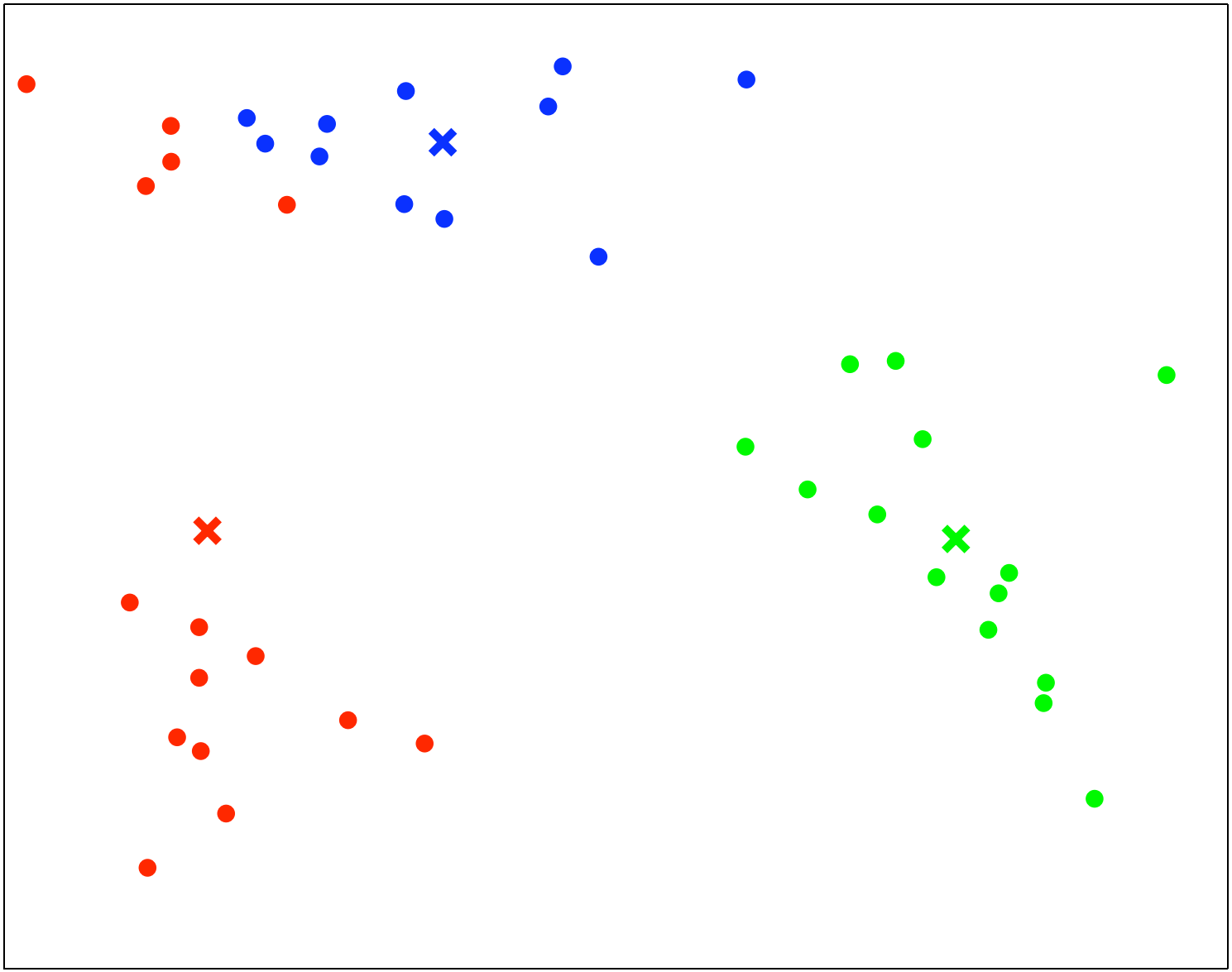
Example: compute centroids



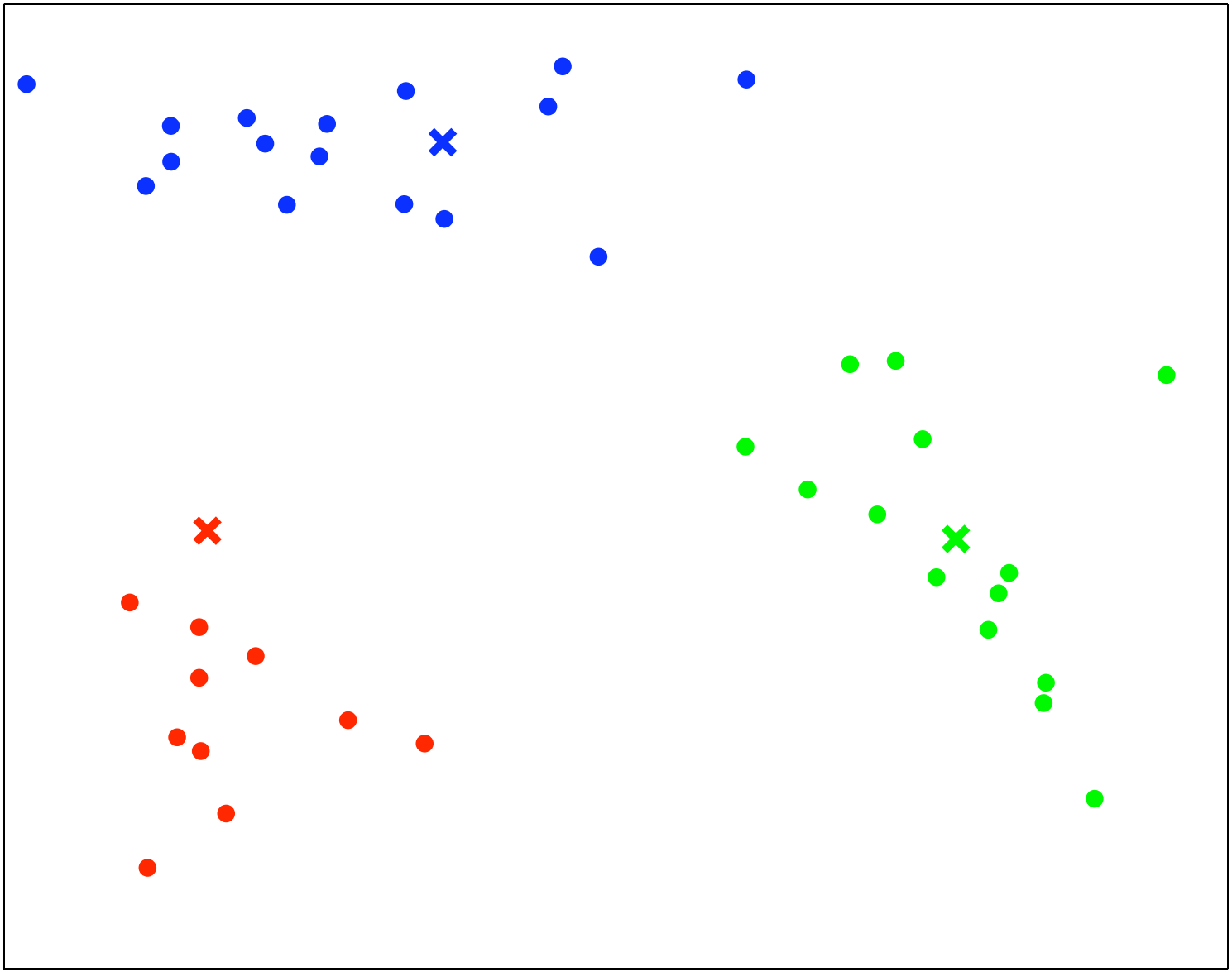
Example: reassign clusters



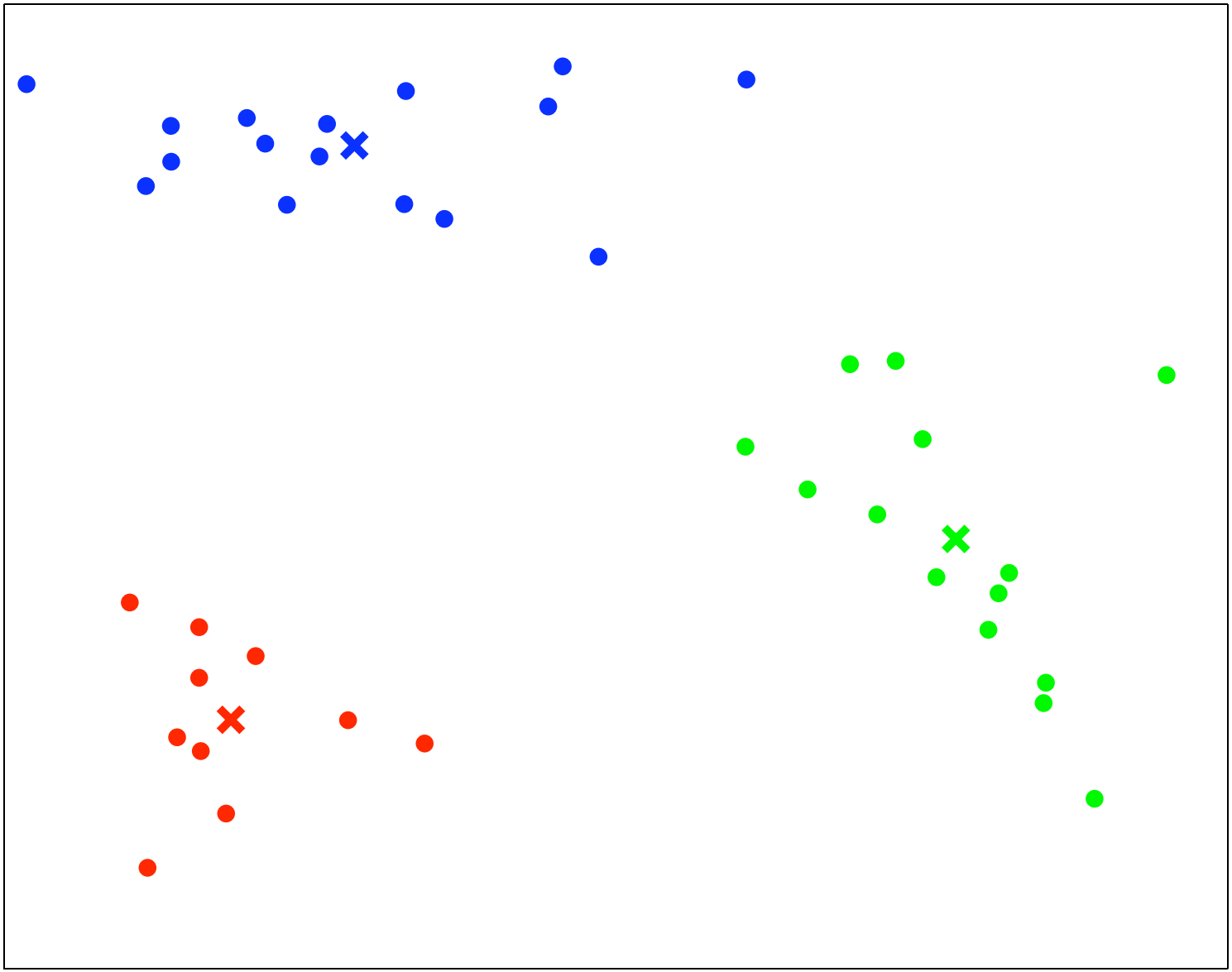
Example: recompute centroids



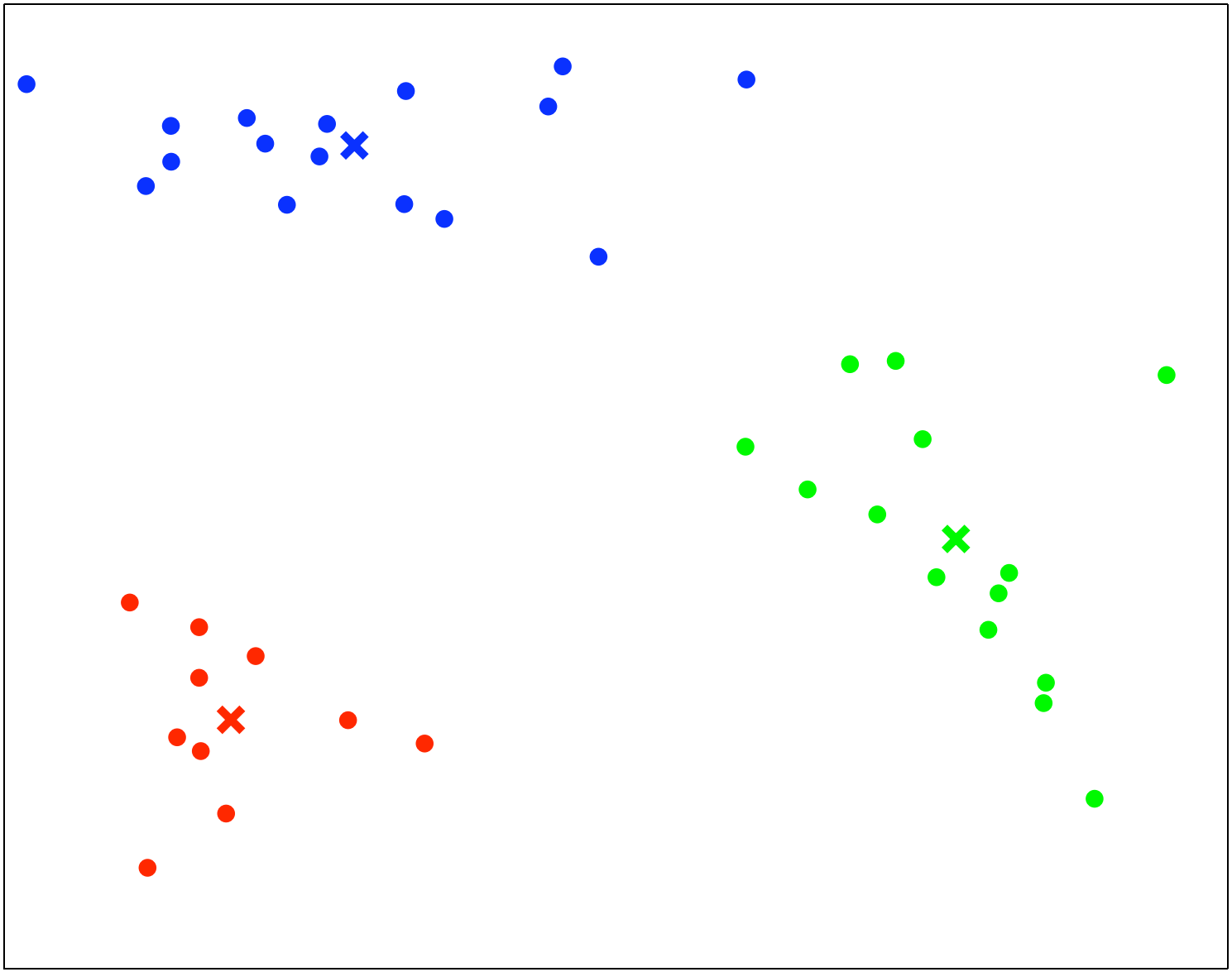
Example: reassign clusters



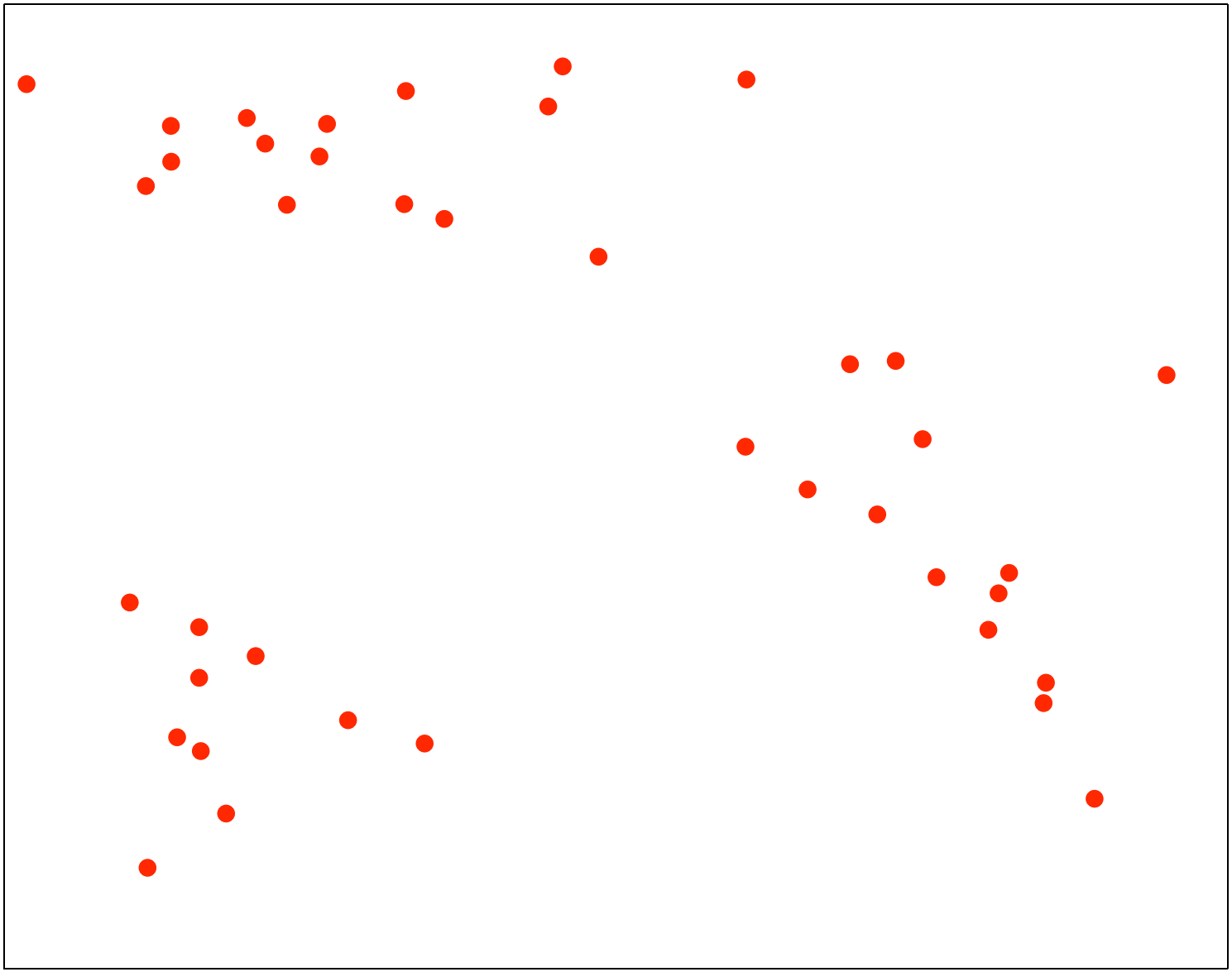
Example: recompute centroids



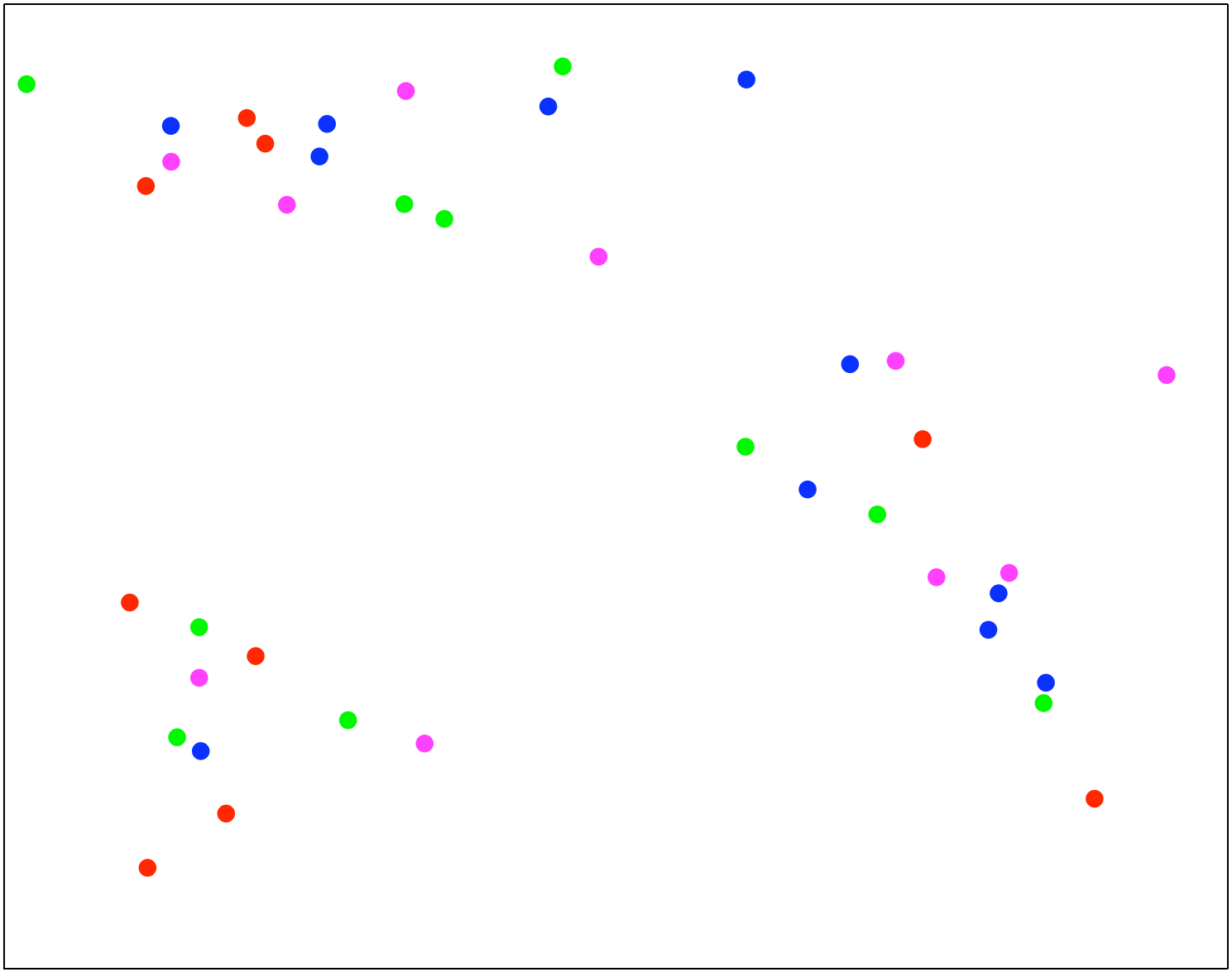
Example: reassign clusters – done!



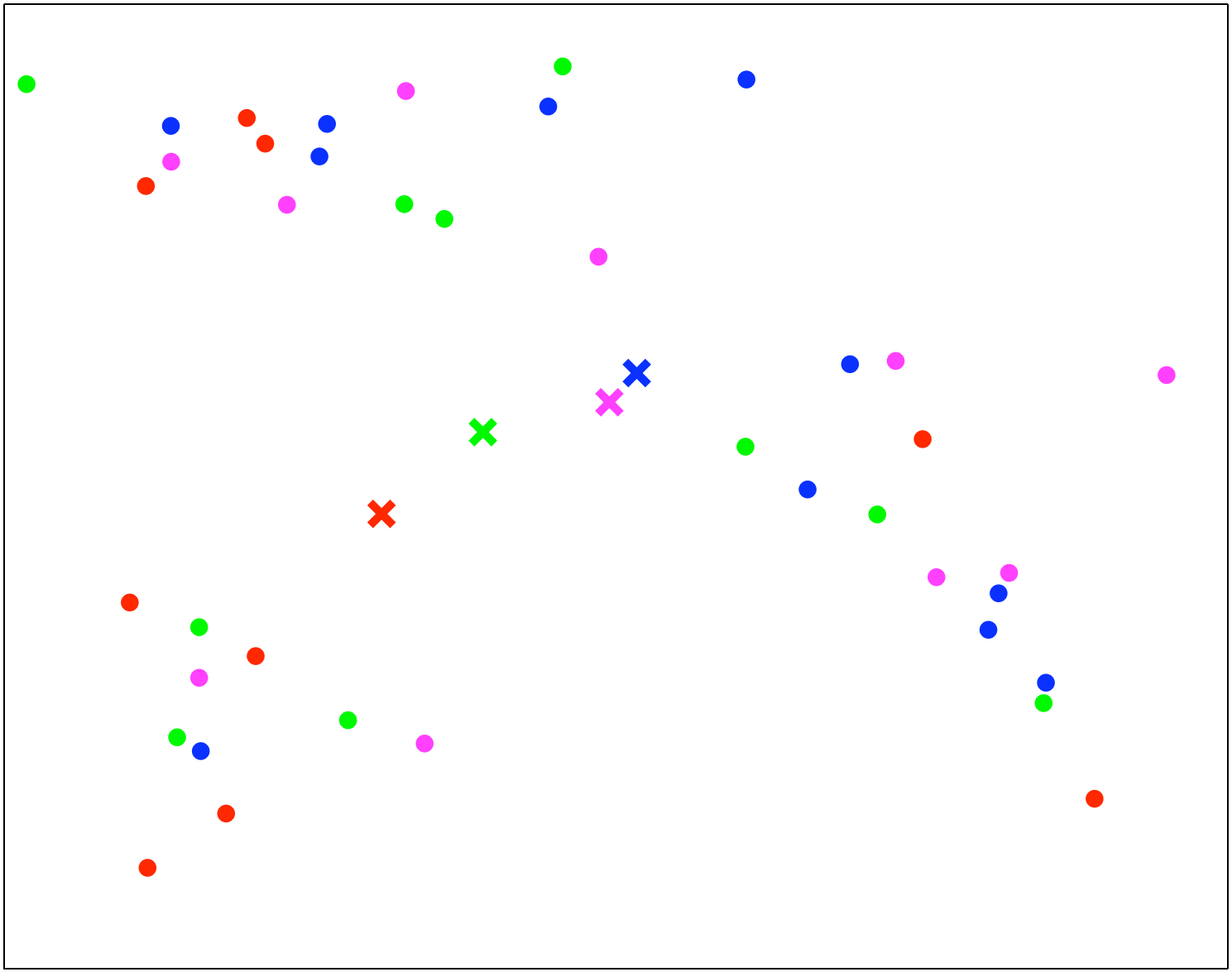
What if we do not know the right number of clusters?



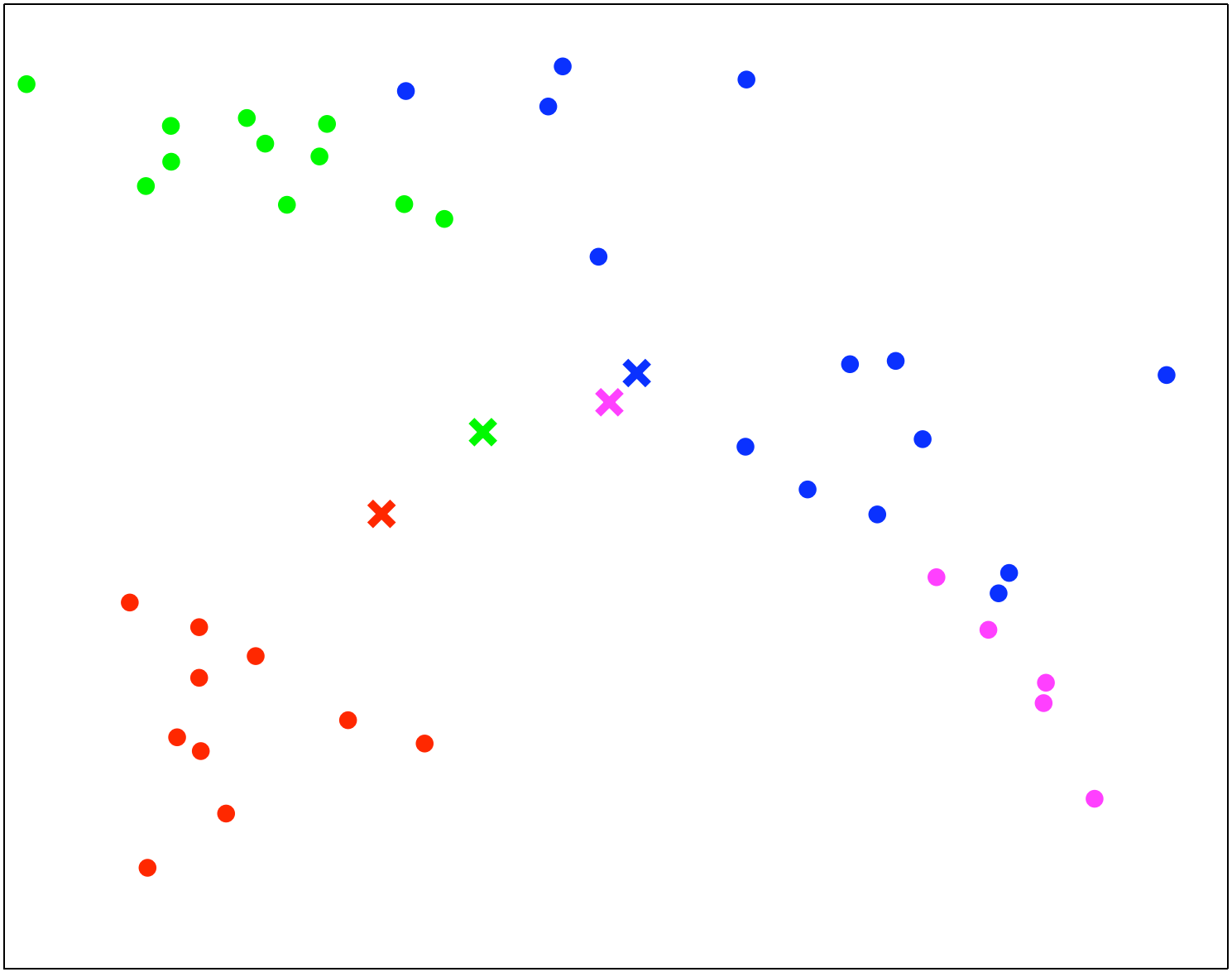
Example: assign into 4 clusters randomly



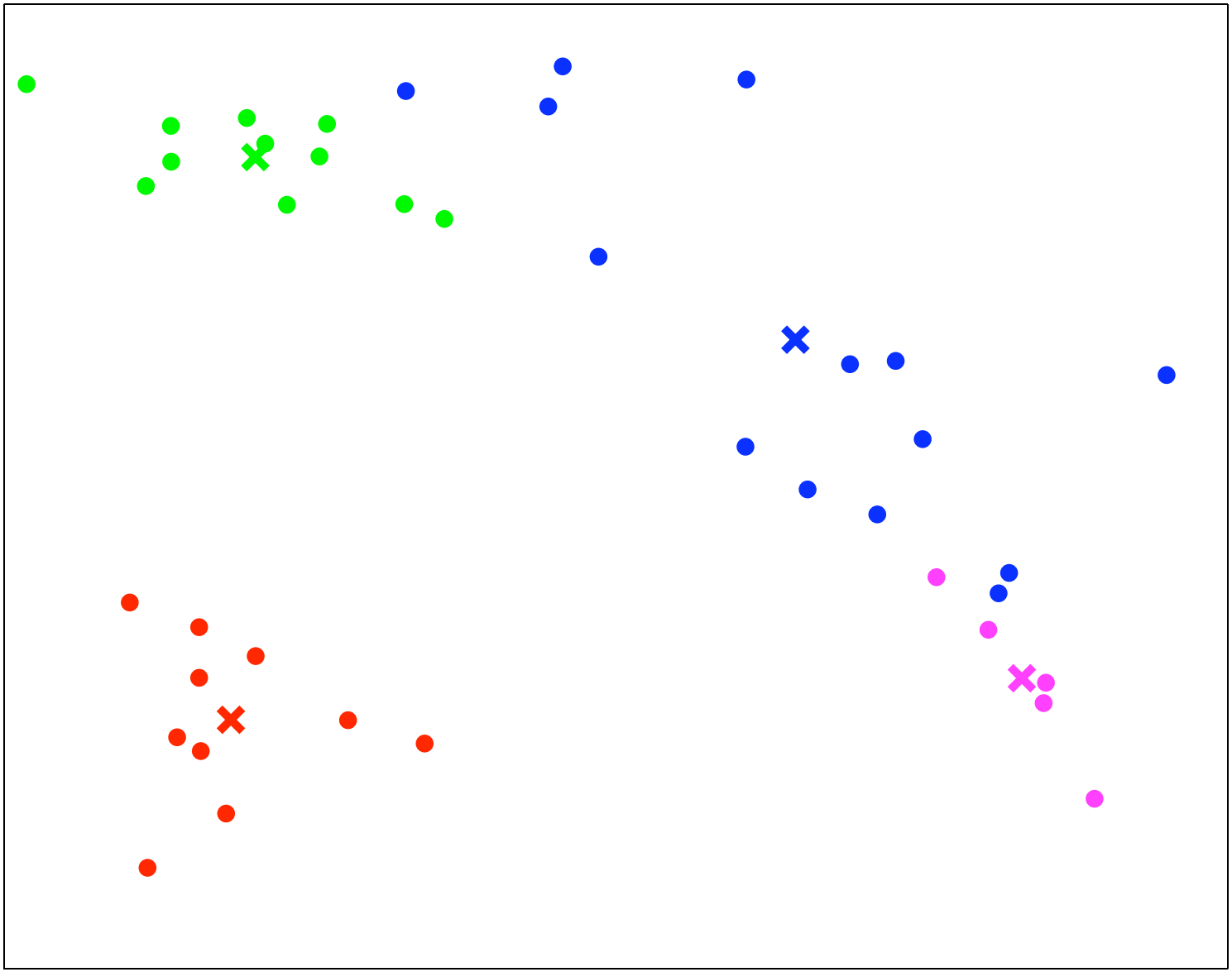
Example: compute centroids



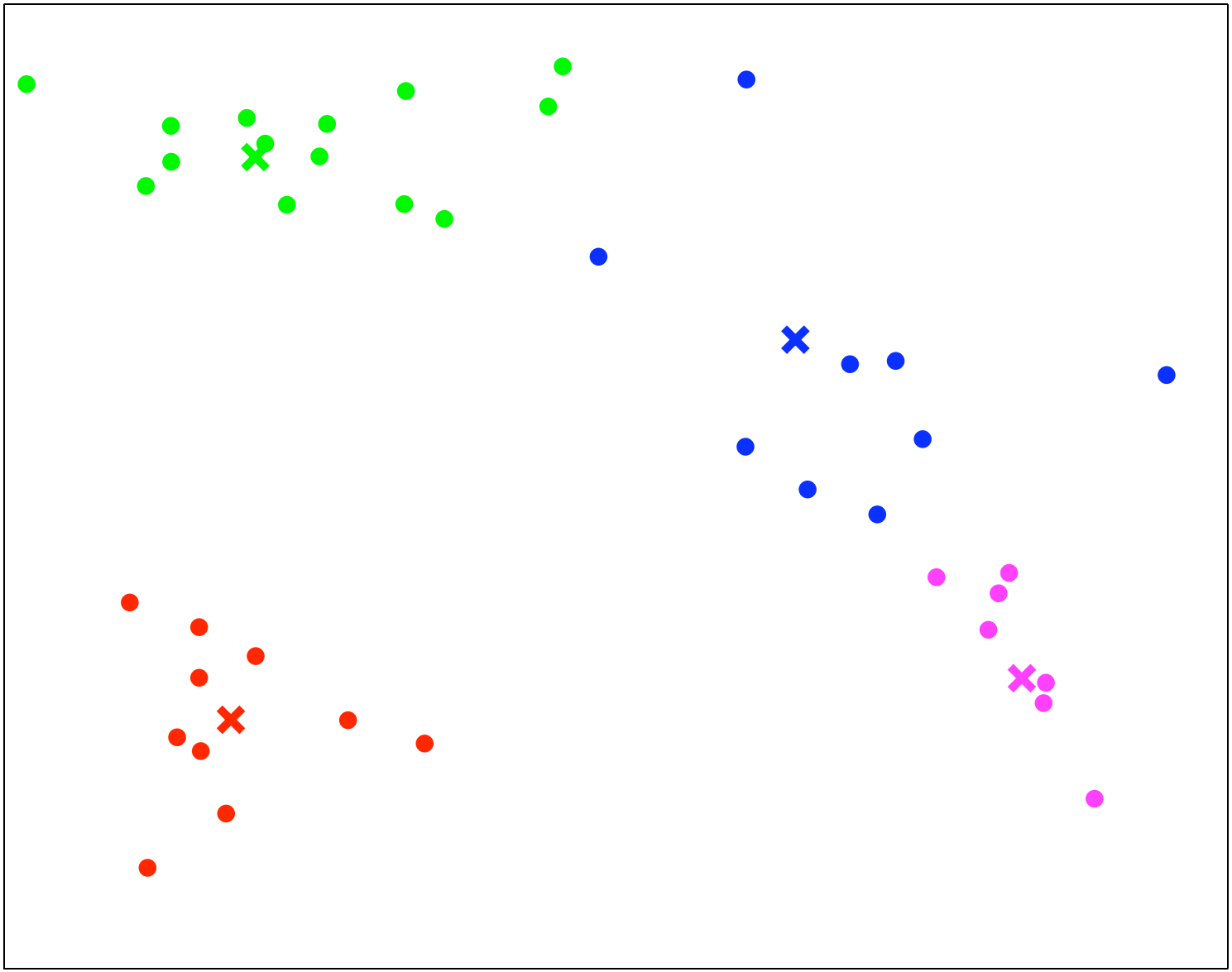
Example: reassign clusters



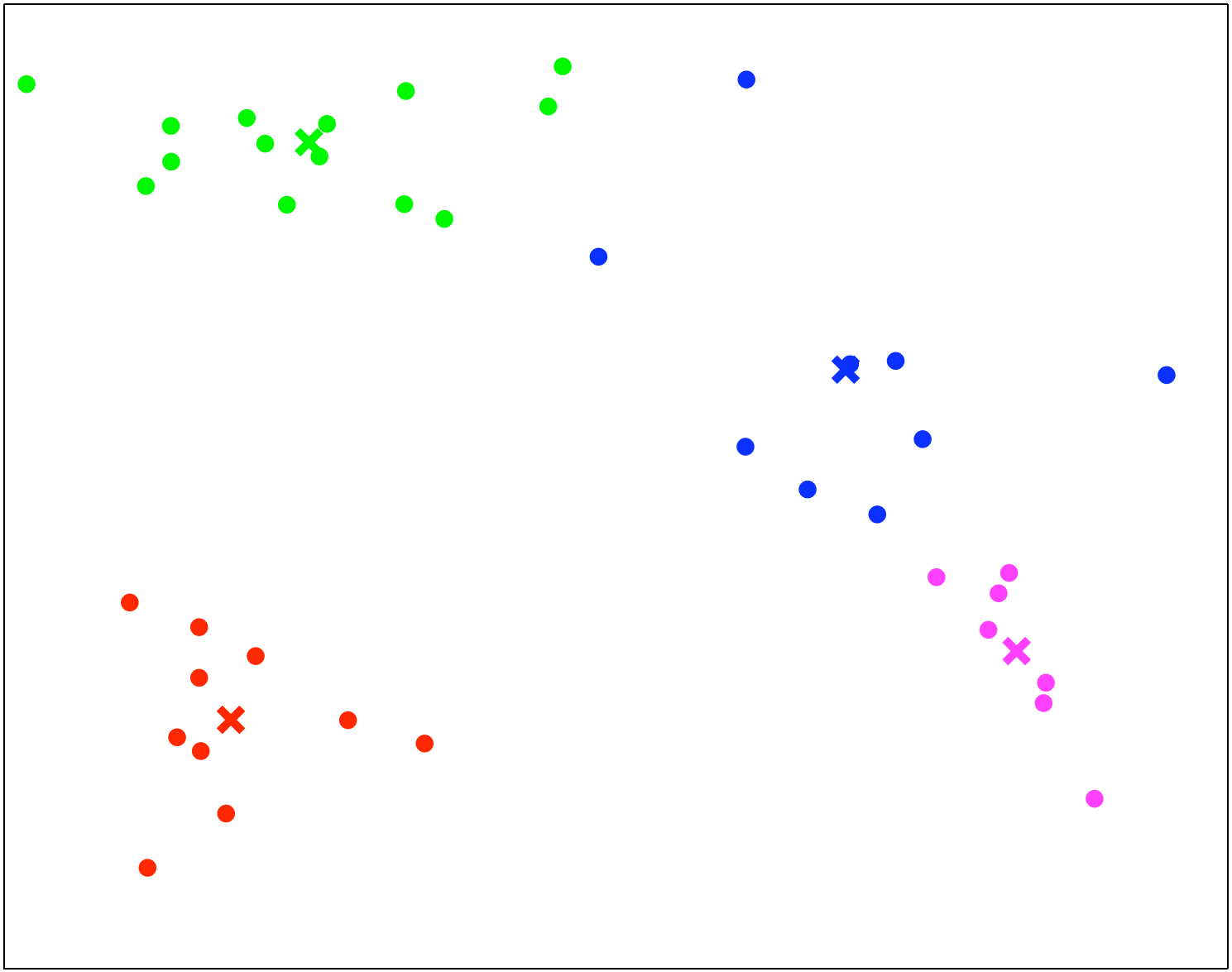
Example: recompute centroids



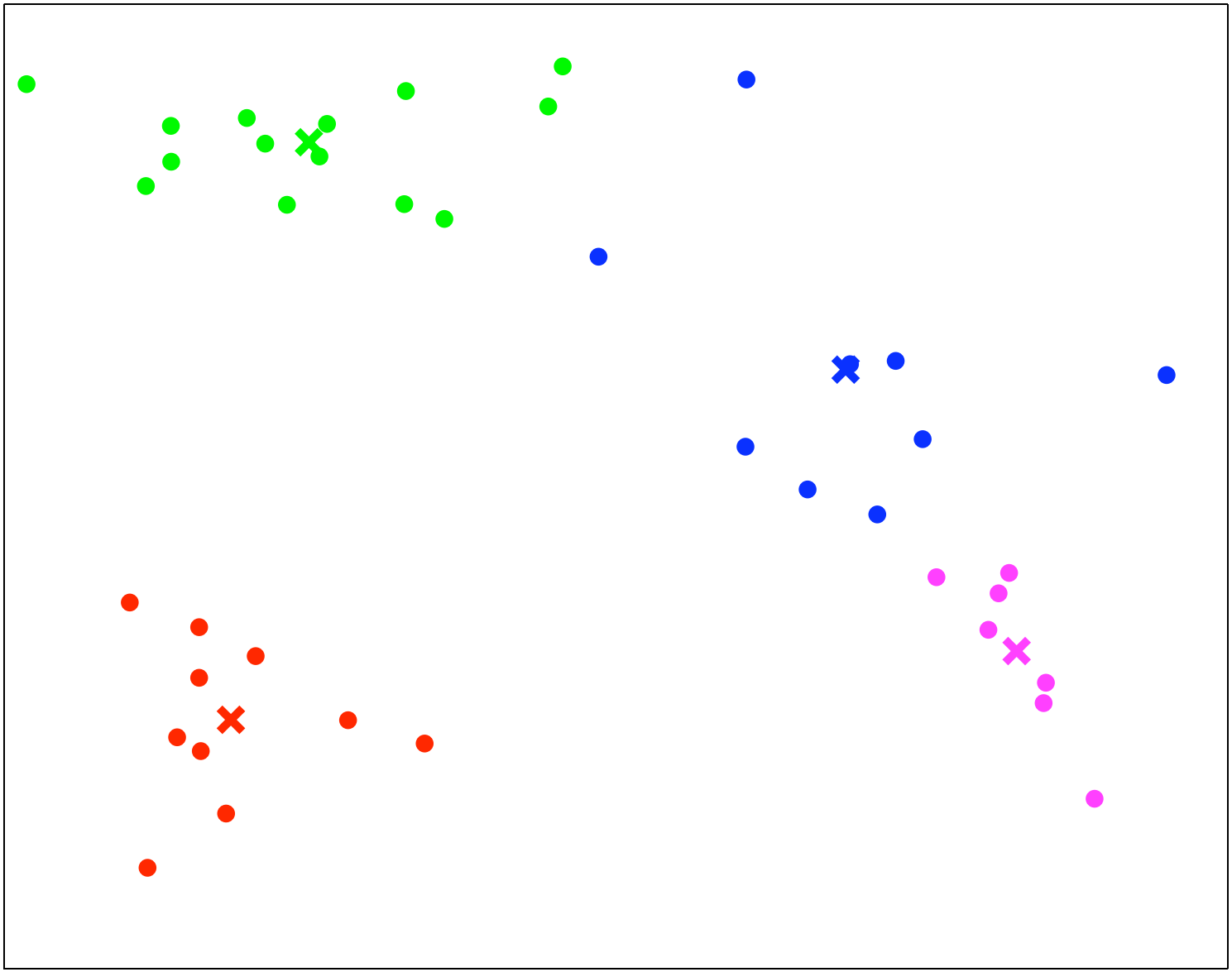
Example: reassign clusters



Example: recompute centroids



Example: reassign clusters – done!



Assessing the quality of the clustering

- If the clustering is used as a pre-processing step for supervised learning, measure the performance of the supervised learner
- Measure the “tightness" of the clusters: points in the same cluster should be close together, points in different clusters should be far apart
- Tightness can be measured by the minimum distance, maximum distance or average distance between points
- Problem: these measures usually favour large numbers of clusters, so penalizing the number of clusters is necessary

Typical applications of clustering

- Pre-processing step for supervised learning
- Data inspection/experimental data analysis
- Discretizing real-valued variables in non-uniform buckets.
- Data compression

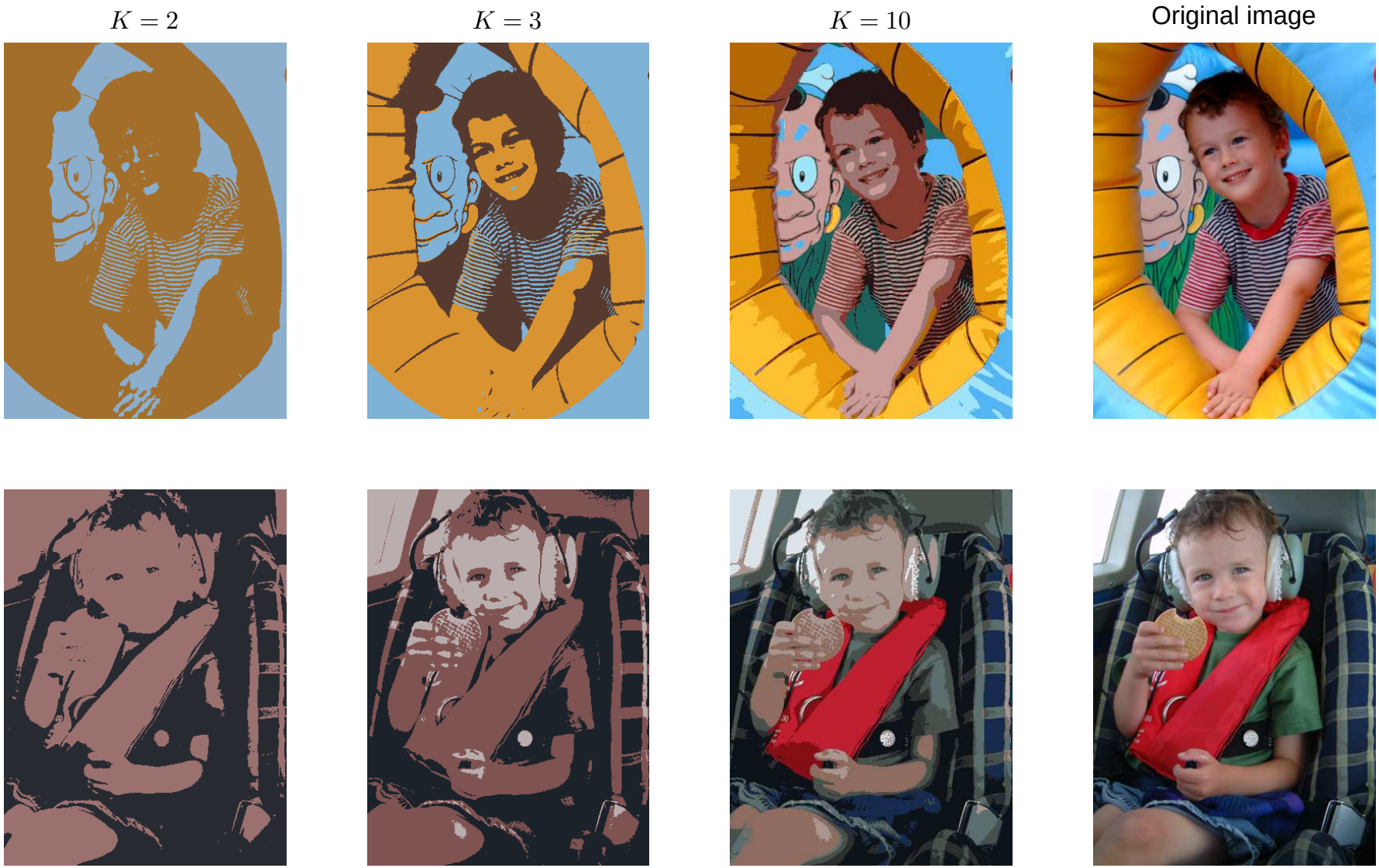
Example application: Color quantization

- Suppose you have an image stored with 24 bits per pixel (3 bytes)
- 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)



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 a data point can be measured as Euclidian distance between the true value and its encoding
- An optimal K-means clustering minimizes the squared reconstruction error among all possible codings of the same type

Questions

- Will K-means 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 $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and a clustering C, consider the sum of the squared Euclidian distance between each vector and the center of its cluster:

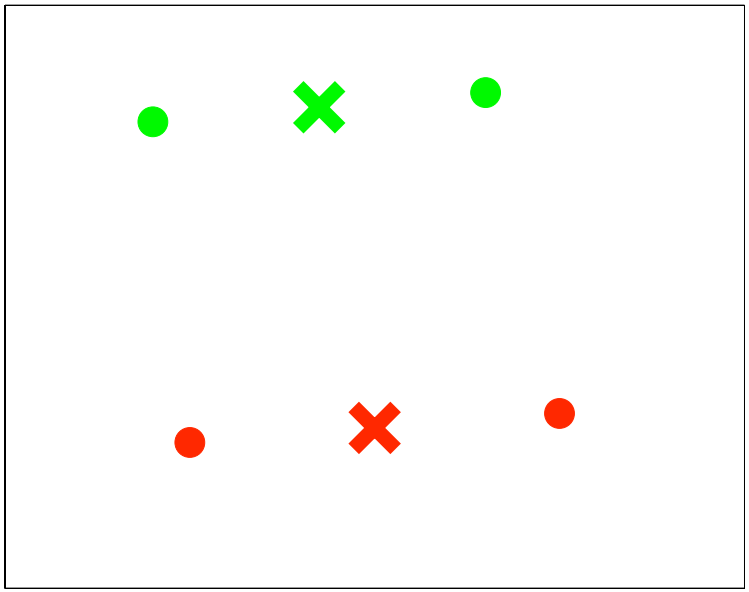
$$J = \sum_{i=1}^{n} \|\mathbf{x}_i - \mu_{C(i)}\|^2 \ ,$$

  where $\mu_{C(i)}$ denotes the centroid of the cluster containing $\mathbf{x}_i$.
- There are finitely many possible clusterings: at most K^n.
- 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) because the sample mean minimizes sum of squared differences.
- Thus, the algorithm must terminate.

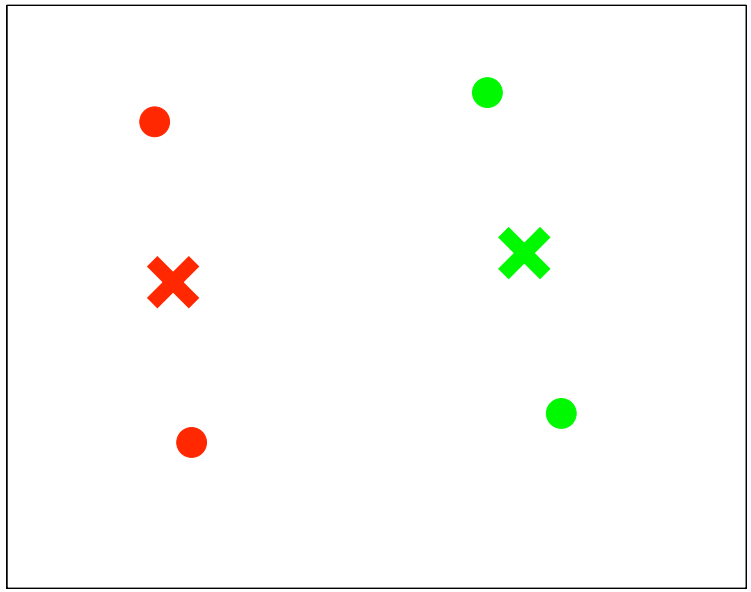
Does K-means always find the same answer?

- 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



$J = 0.22870$



$J = 0.3088$

Finding good initial configurations

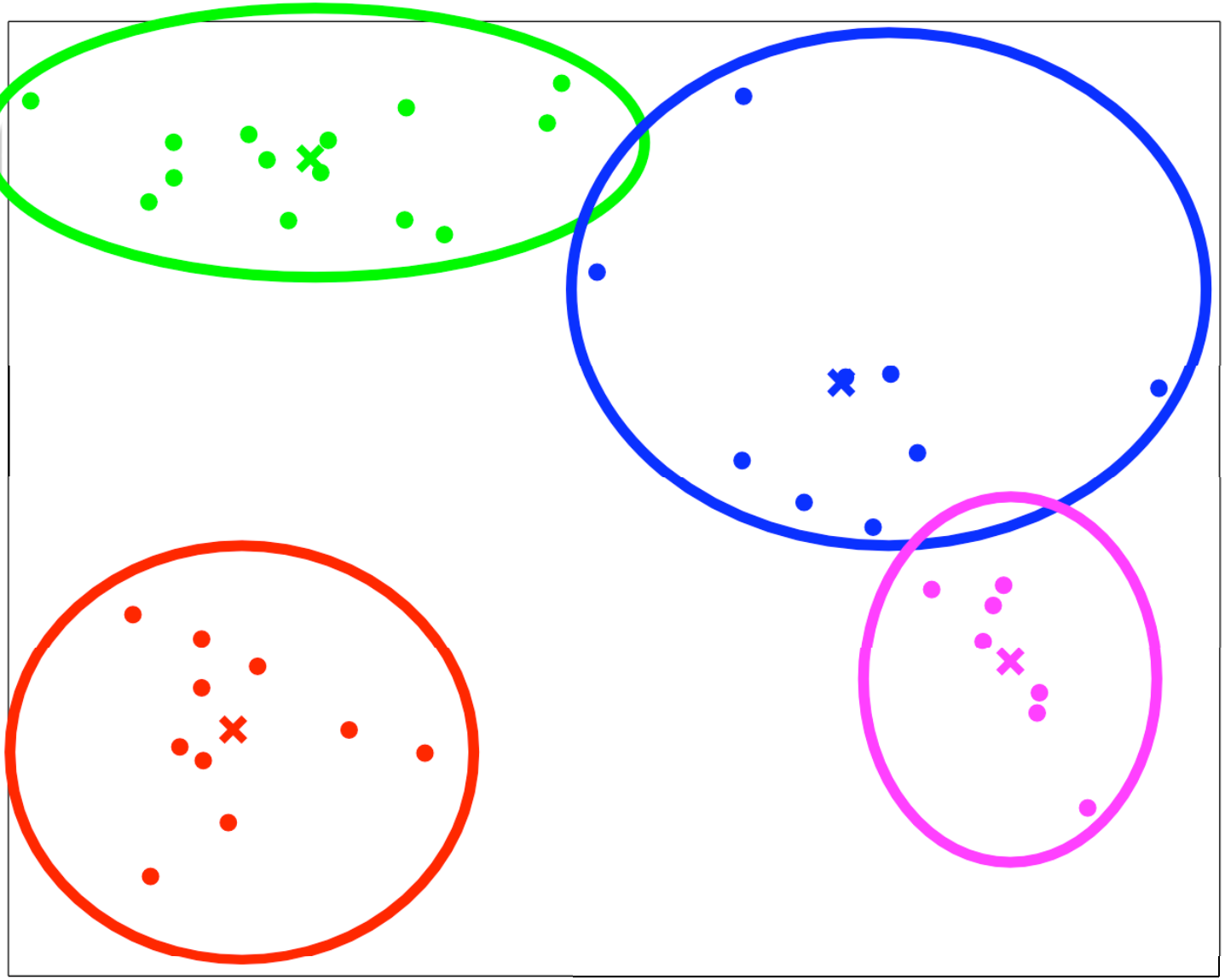
- The initial configuration can influence the final clustering
- Assigning each item to random cluster in $\{1, \ldots, K\}$ is unbiased... but typically results in cluster centroids near the centroid of all the data in the first round.
- A different heuristic tries to spread the initial centroids around 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 randomized 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. Our own Shai Ben-David has done significant work.
  `http://www.cs.uwaterloo.ca/~shai/`
- 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)

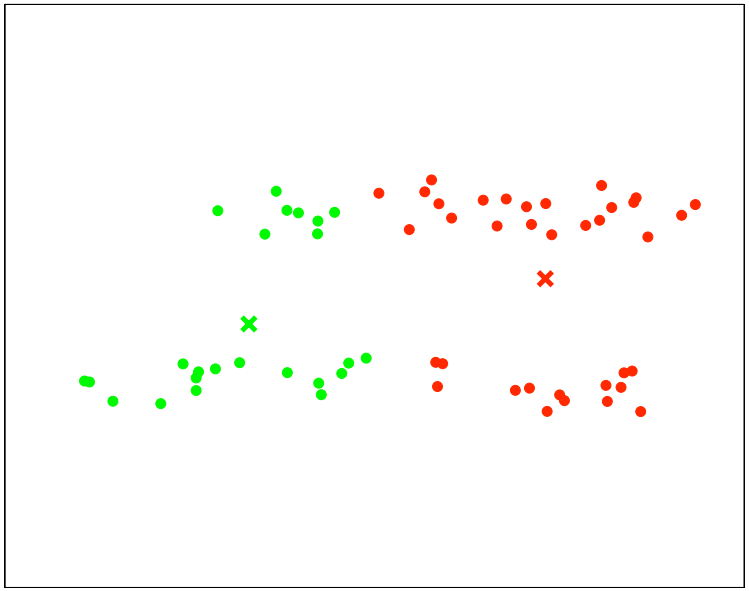
Why the sum of squared Euclidean distances?

Subjective reason: It produces nice, round clusters.



Why *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

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
     1. Find the two clusters in W that are most “similar".
     2. Remove them from W.
     3. 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: $2n - 1$, where n is the number of data objects.
- Why?
  - The working list W starts with n singleton clusters
  - Each iteration removes two clusters from W and adds one new cluster
  - The algorithm stops when W has one cluster, which is after $n - 1$ iterations

How do we measure dissimilarity between clusters?

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

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

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

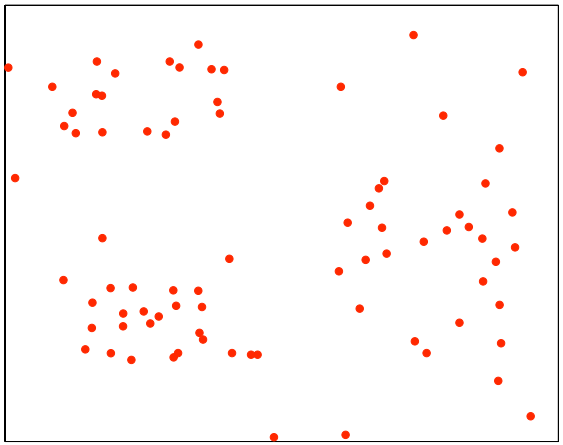
$$\max_{x \in C, x' \in C'} d(\mathbf{x}, \mathbf{x}')$$

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

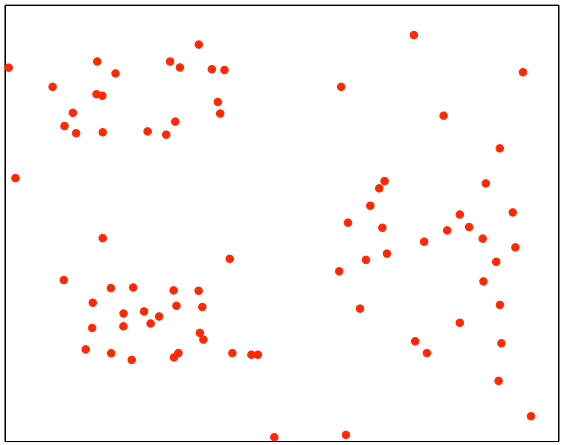
$$\frac{1}{|C||C'|} \sum_{x \in C, x' \in C'} d(\mathbf{x}, \mathbf{x}')$$

Example 1: Data

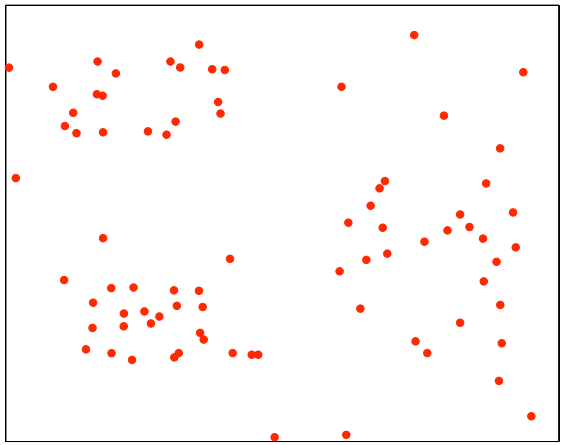
Single



Average

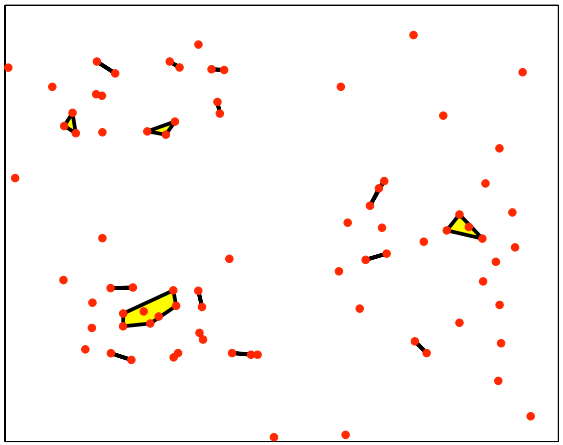


Complete

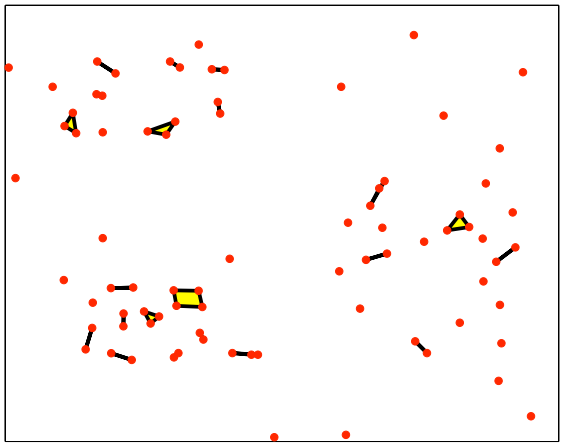


Example 1: Iteration 30

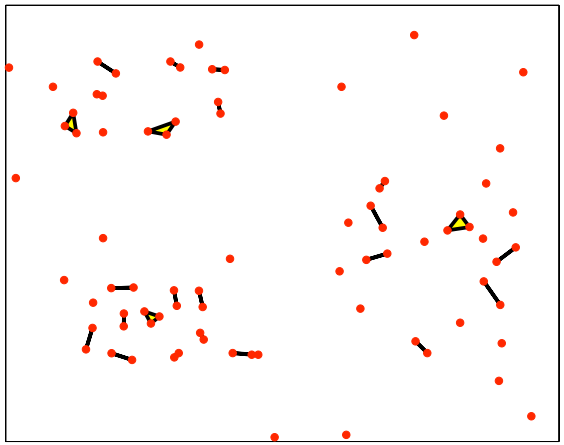
Single



Average

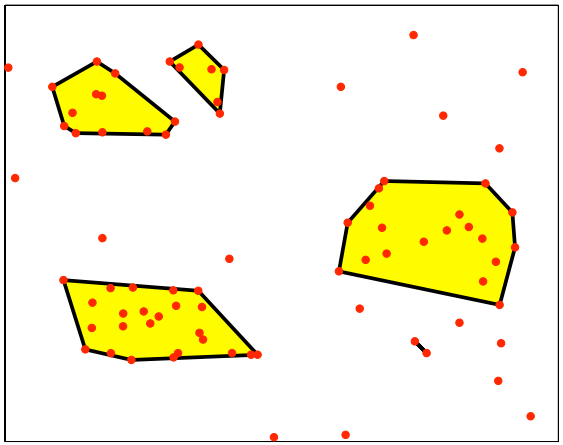


Complete

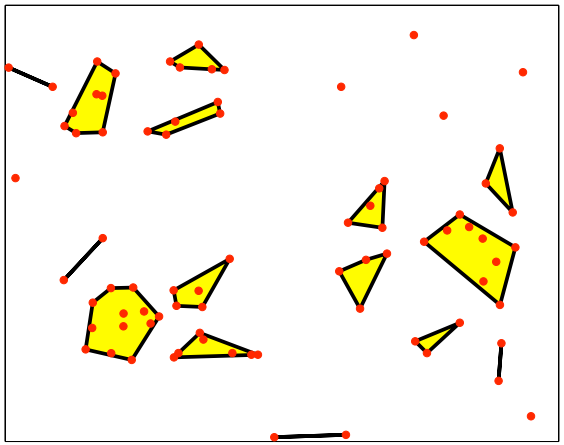


Example 1: Iteration 60

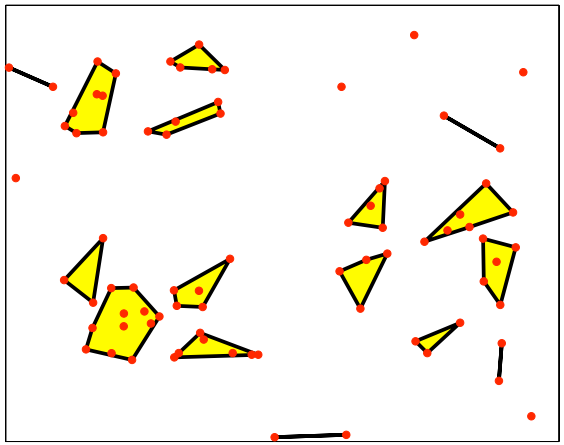
Single



Average



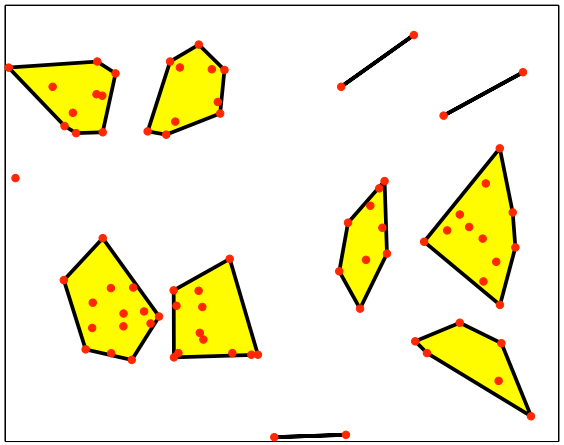
Complete



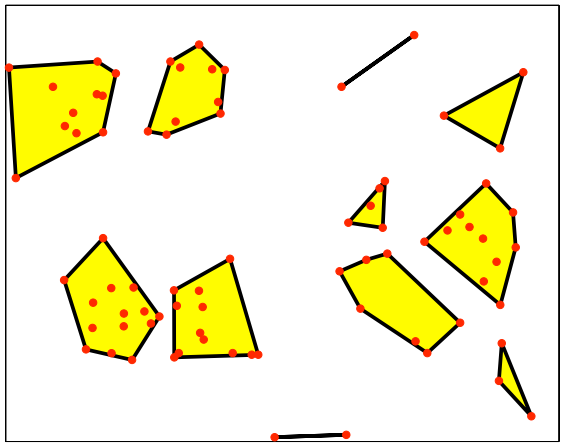
Example 1: Iteration 70

Single

Average

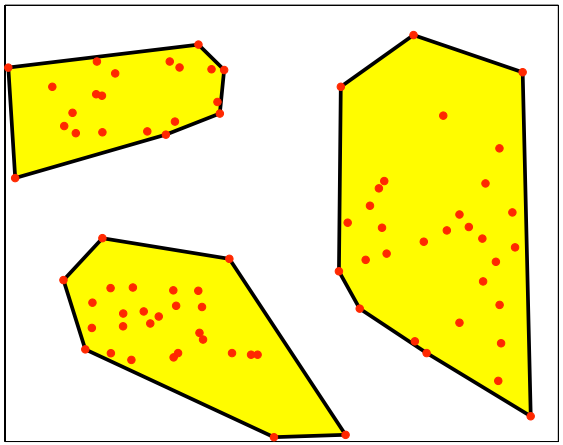


Complete

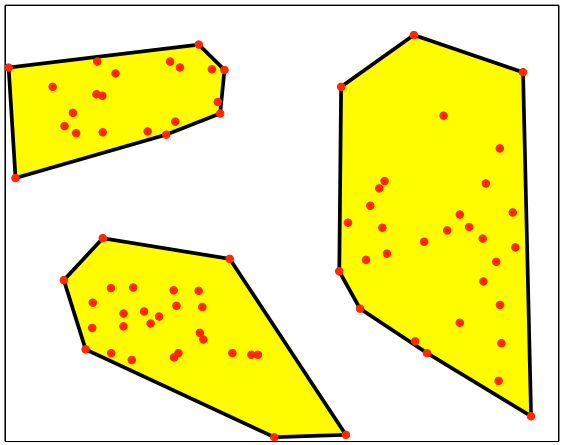


Example 1: Iteration 78

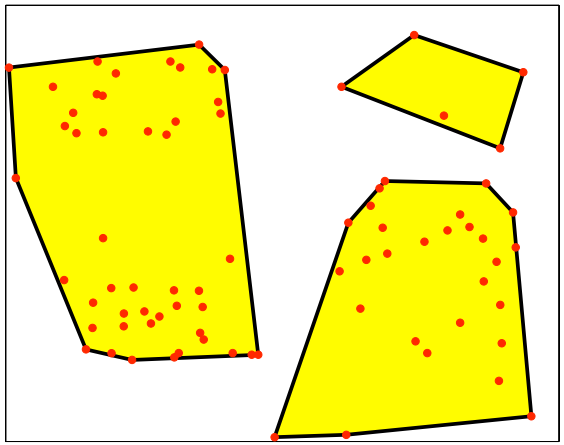
Single



Average

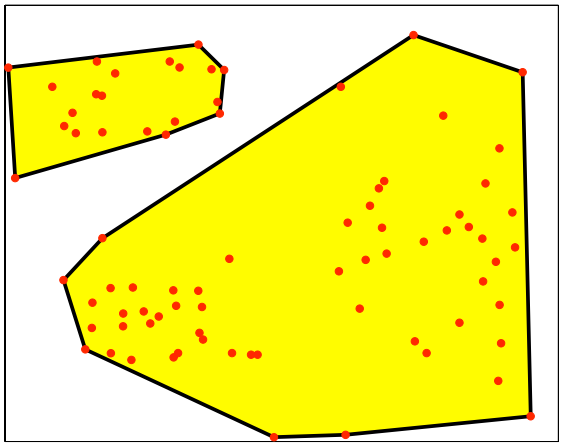


Complete



Example 1: Iteration 79

Single

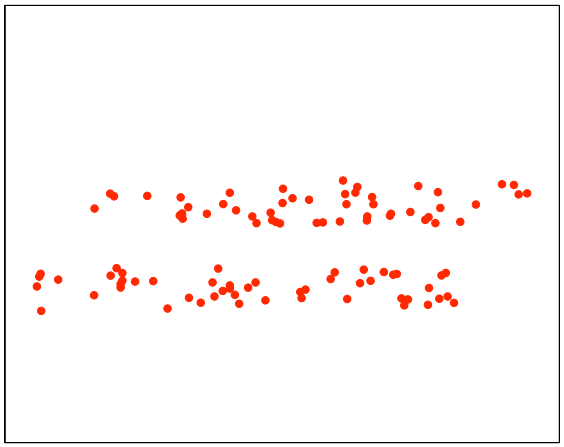


Average

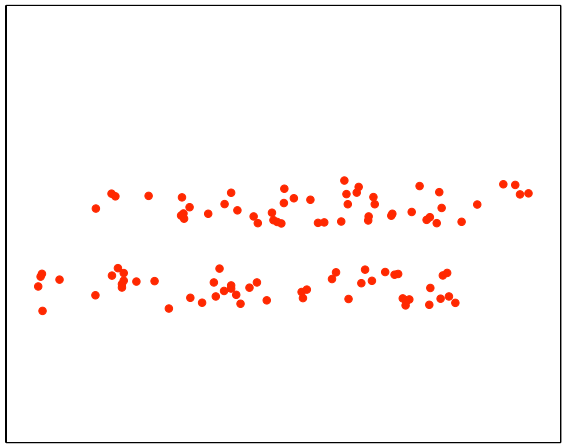
Complete

Example 2: Data

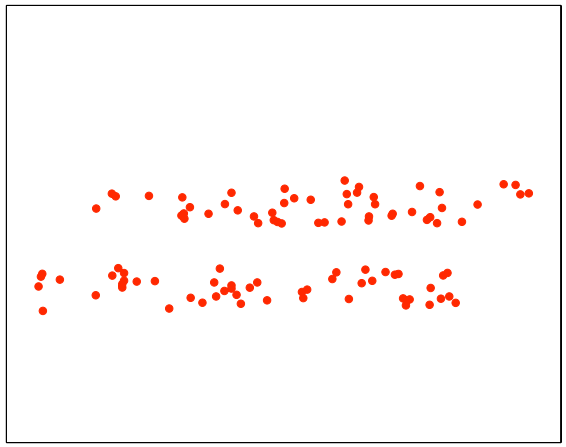
Single



Average

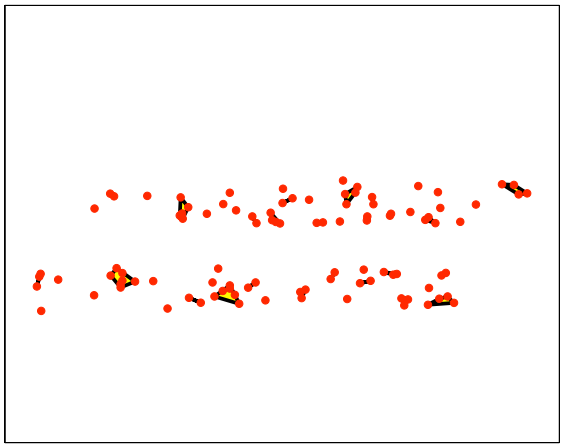


Complete

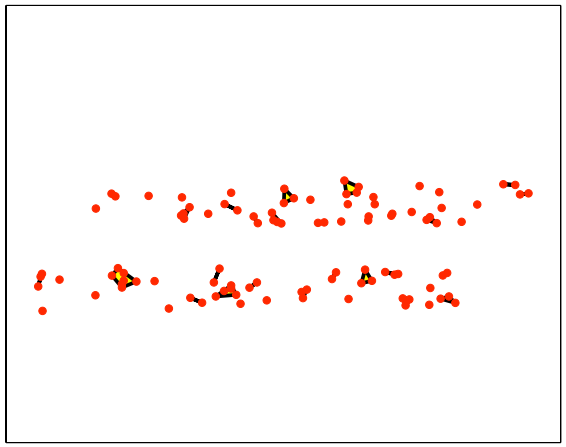


Example 2: Iteration 50

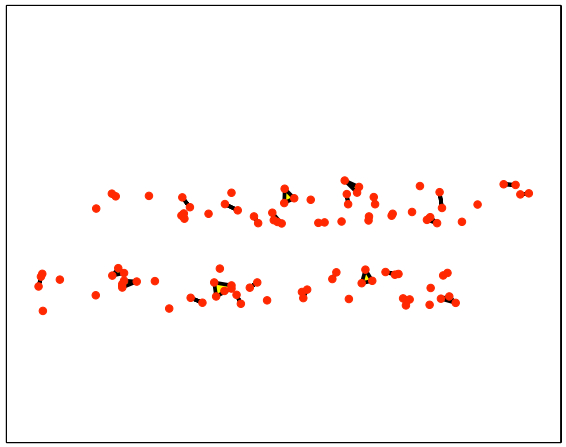
Single



Average

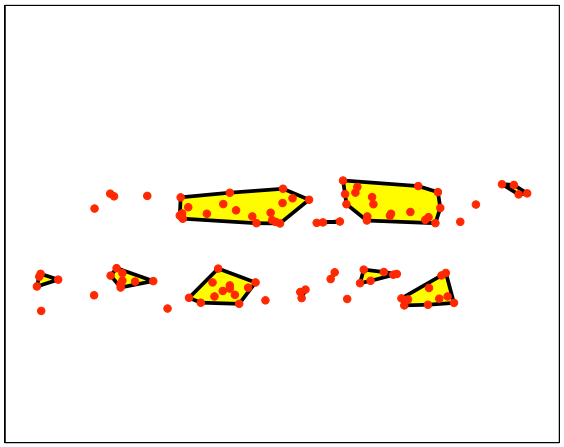


Complete

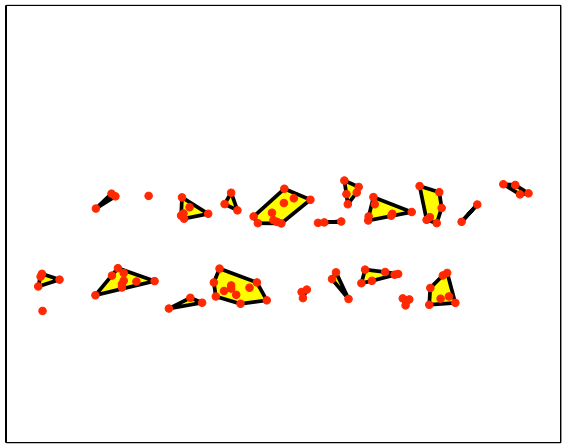


Example 2: Iteration 80

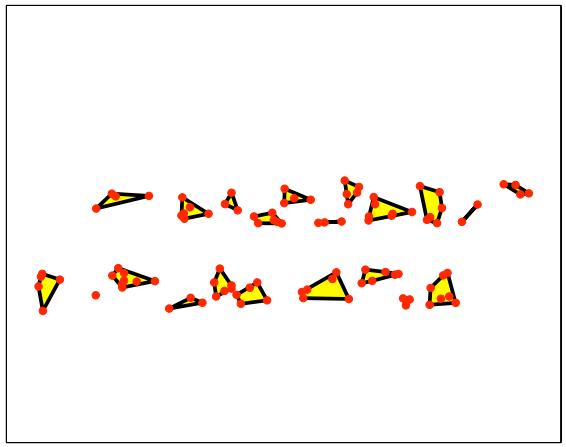
Single



Average

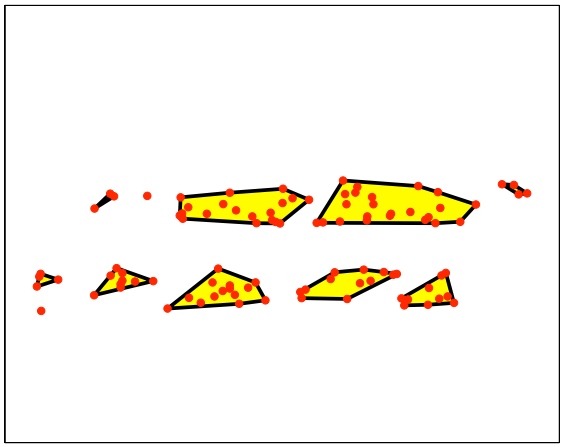


Complete

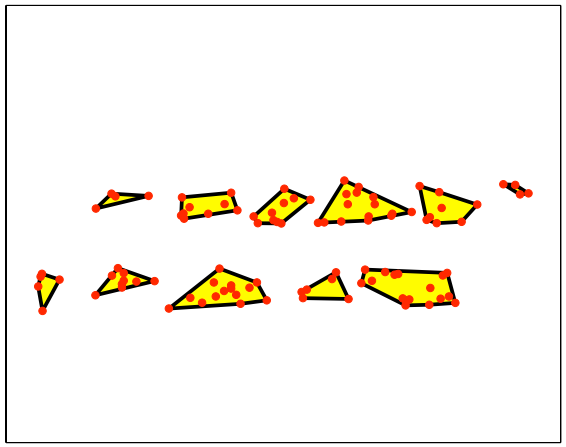


Example 2: Iteration 90

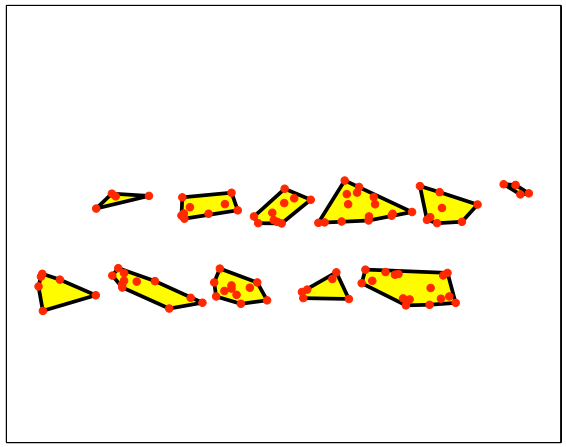
Single



Average

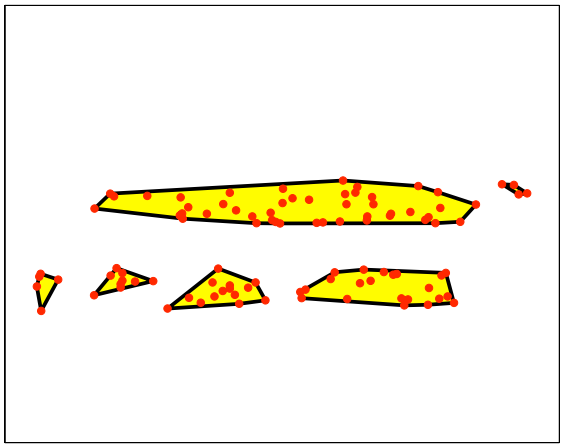


Complete

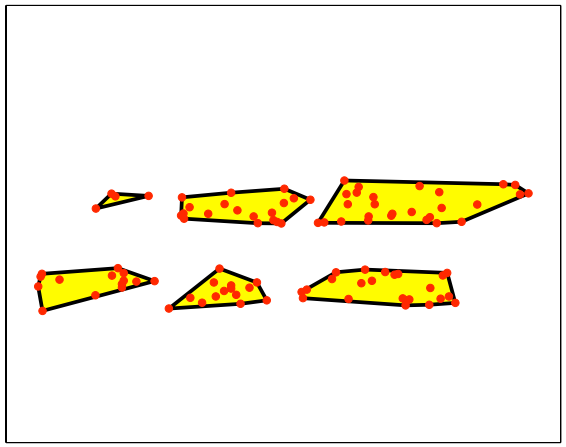


Example 2: Iteration 95

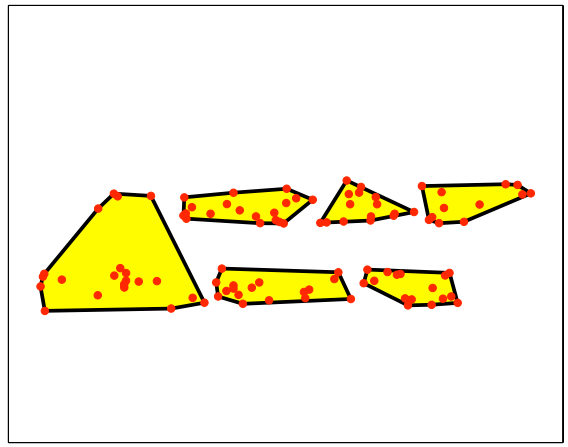
Single



Average

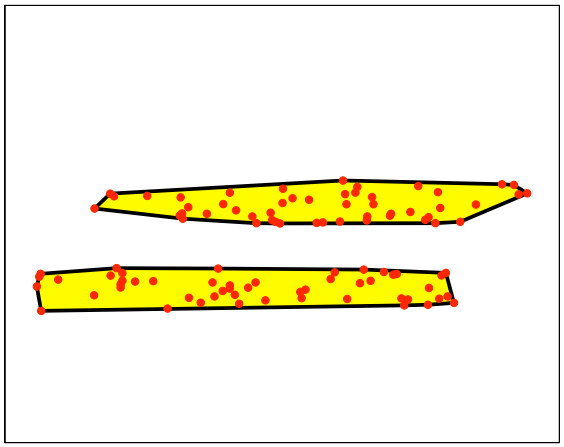


Complete

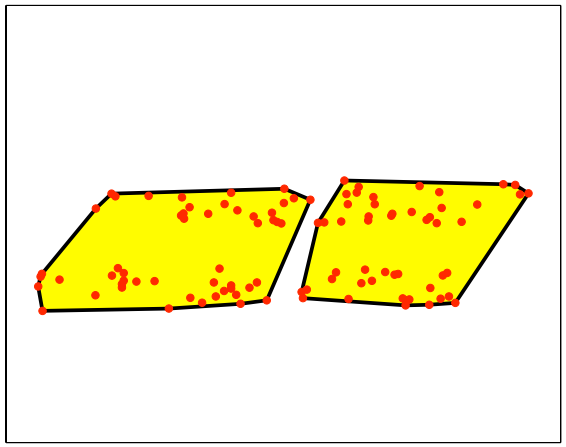


Example 2: Iteration 99

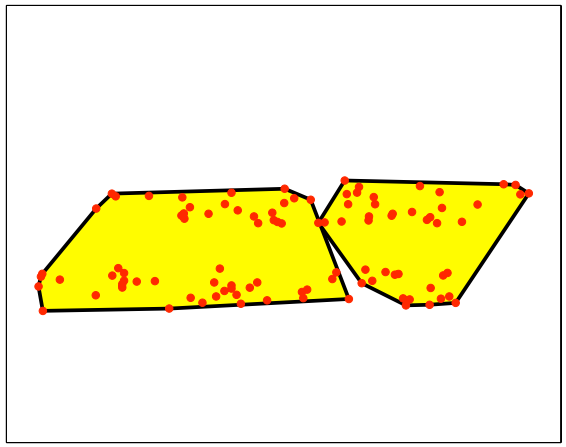
Single



Average



Complete



Intuitions about cluster similarity

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

Summary of clustering

- 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.