Statistical Machine Learning

Unsupervised learning

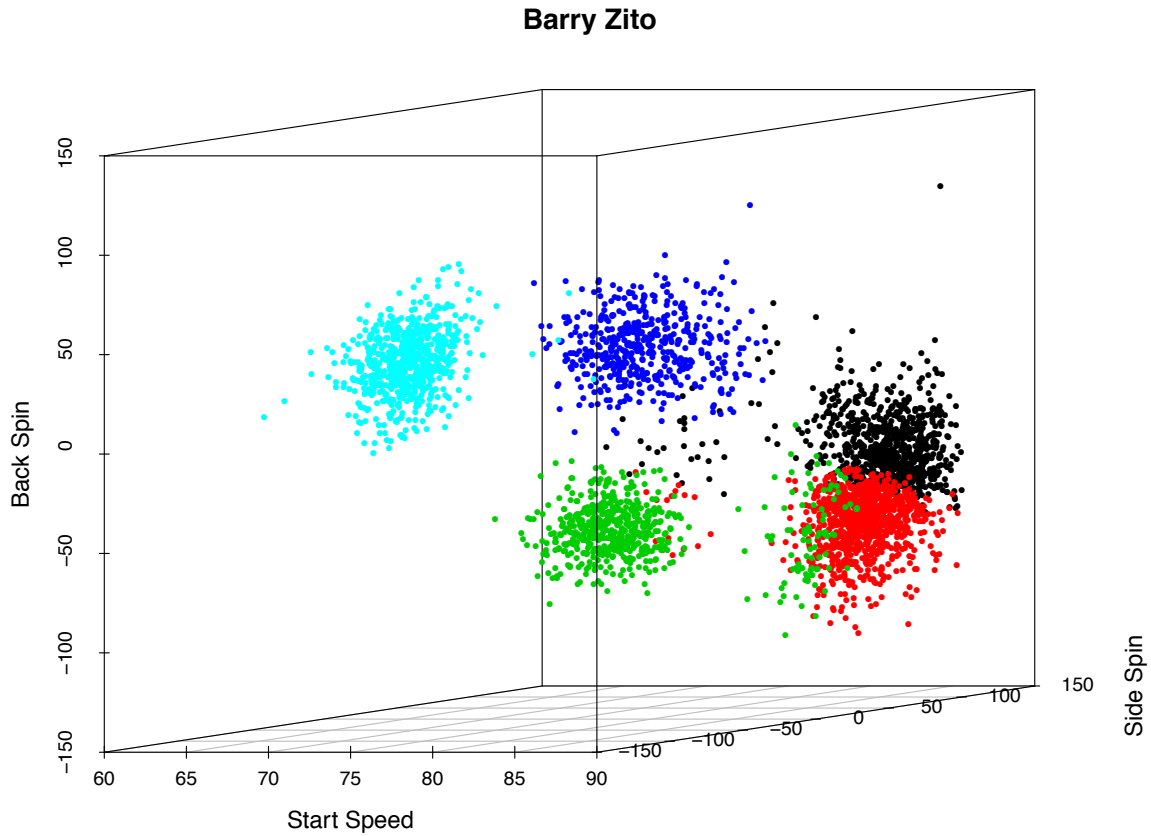
April, 12, 2019

Unsupervised learning

What is clustering? And why?

- Clustering: task of dividing up data into groups (clusters), so that points in any one group are more "similar" to each other than to points outside the group
- Why cluster? Two main uses
  - Summary: deriving a reduced representation of the full data set. E.g., vector quantization (we will see this)
  - Discovery: looking for new insights into the structure of the data.
  - Validation: Checking up on someone else's work/decisions, investigating the validity of pre-existing group assignments
  - Helping with prediction, i.e., in classification or regression

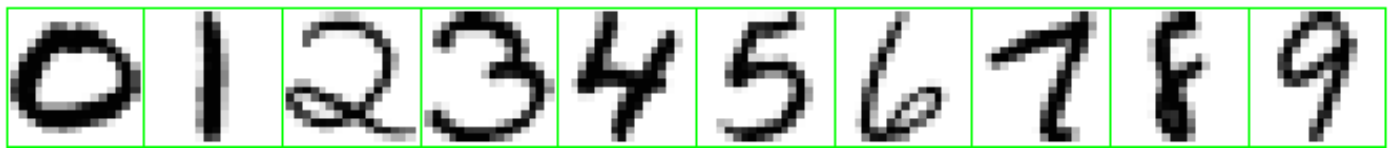
A neat example: Baseball pitches



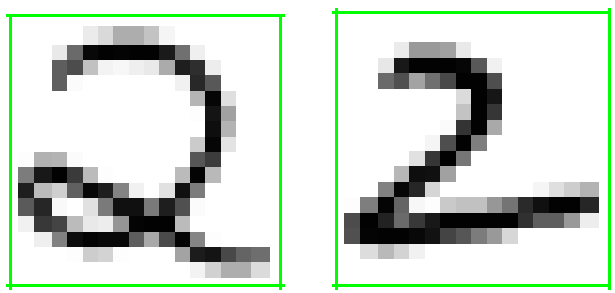
Inferred meaning of clusters:
black – fastball, red – sinker, green – changeup, blue – slider,
light blue – curveball

Don't confuse clustering and classification!

- In classification, we have data for which the groups are known, and we try to learn what differentiates these groups (i.e., classification function) to properly classify future data



- In clustering, we look at data for which groups are unknown and undefined, and try to learn the groups themselves, as well as what differentiates them



Two clustering methods

- In K-means clustering, we seek to partition the observations into a pre-specified number of clusters.
- In hierarchical clustering, we do not know in advance how many clusters we want; in fact, we end up with a tree-like visual representation of the observations, called a dendrogram, that allows us to view at once the clusterings obtained for each possible number of clusters, from 1 to n.

Dissimilarity and within-cluster scatter

- Given two observations $x_1, \ldots, x_n$, and dissimilarity measure $d(x_i, x_j)$.
- For example, $x_i \in \mathbb{R}^p$, $d(x_i, y_j) = \|x_i - x_j\|_2^2$
- Let K be the number of clusters (fixed). A clustering is a function C that assigns each observation x_i to a group $k \in \{1, \ldots, K\}$
- Notation: $C(i) = k$ means that x_i is assigned to group k, and n_k is the number of points in the group k.
- The with-cluster scatter is defined as

$$W = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{C(i)=k, C(j)=k} d(x_i, x_j)$$

- We want to minimize W by choosing different C

Finding the best group assignments

- Smaller W is better, so why don't we just directly find the clustering C that minimizes W?
- Problem: doing so requires trying all possible assignments of the n points into K groups. The number of possible assignments is

$$A(n, K) = \frac{1}{K!} \sum_{k=1}^{K} (-1)^{K-k} \binom{K}{k} k^n$$

- $A(10, 4) = 34,105$, and $A(25, 4) \approx 5 \times 10^{13}$
- Most problems we look at are going to have way more than $n = 25$ observations, and potentially more than $K = 4$ clusters too
- Consider an approximation approach

Rewriting the within-cluster scatter

- Focus on Euclidean space: now $x_i \in \mathbb{R}^p$ and dissimilarities are $d(x_i, x_j) = \|x_i - x_j\|_2^2$
- Fact:

$$\frac{1}{2}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{C(i)=k, C(j)=k}\|x_i - x_j\|_2^2 = \sum_{k=1}^{K}\sum_{C(i)=k}\|x_i - \bar{x}_k\|_2^2$$

  where $\bar{x}_k = \frac{1}{n_k}\sum_{C(i)=k} x_i$ is the mean of cluster k.
- The right-hand side above is called within-cluster variation
- Hence, equivalently we seek a clustering C that minimizes the within-cluster variation

Rewriting the minimization

- Recall: we want to choose C to

$$\min_{C} \sum_{k=1}^{K} \sum_{C(i)=k} \|x_i - \bar{x}_k\|_2^2$$

- Another fact:

$$\sum_{C(i)=k} \|x_i - c\|_2^2$$

  is minimized by taking $c = \bar{x}_k$.

- The problem is the same as minimizing the enlarged criterion

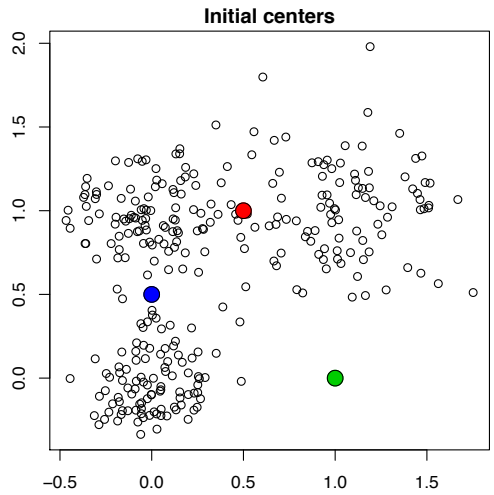
$$\min_{C, c_1, \ldots, c_K} \sum_{k=1}^{K} \sum_{C(i)=k} \|x_i - c_k\|_2^2$$

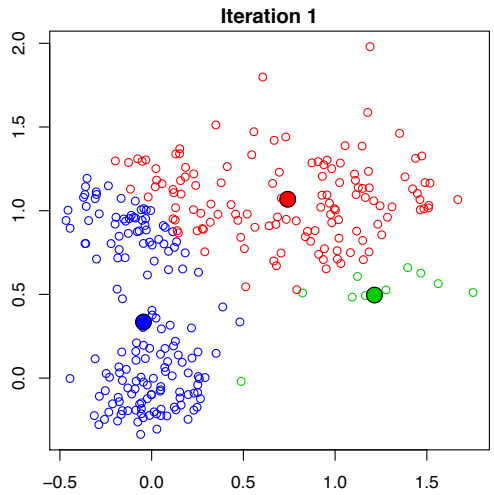
K-means algorithm

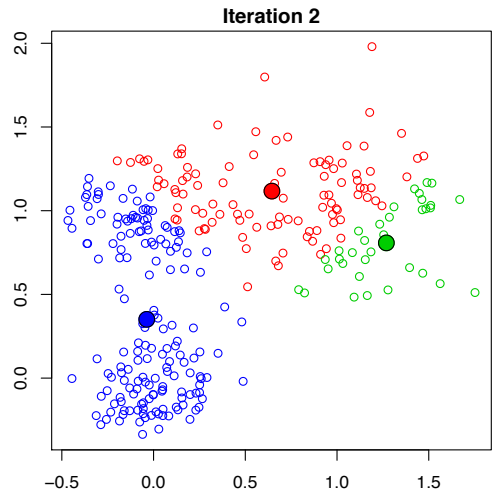
- The K-means clustering algorithm approximately minimizes the enlarged criterion by alternately minimizing over $C, c_1, \ldots, c_K$
- We start with an initial guess for $c_1, \ldots, c_k$ (e.g., pick K points at random over the range of $x_1, \ldots, x_n$), then repeat:
  1. Minimize over C: for each $i = 1, \ldots, n$, find the cluster center c_k closest to x_i, and let $C(i) = k$
  2. Minimize over $c_1, \ldots, c_K$: for each $k = 1, \ldots, K$, let $c_k = \bar{x}_k$, the average of points in cluster k
- Stop when within-cluster variation doesn't change
- In words:
  1. Cluster (label) each point based the closest center
  2. Replace each center by the average of points in its cluster

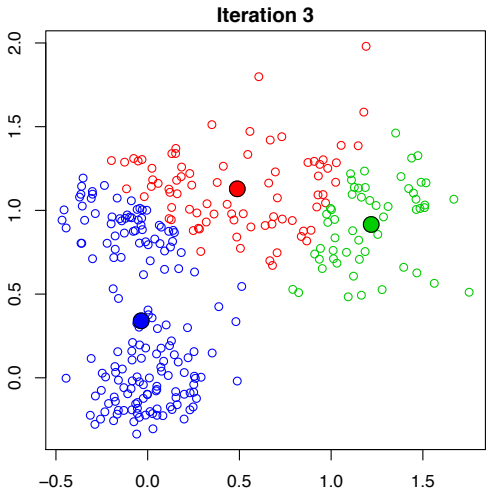
K-means example

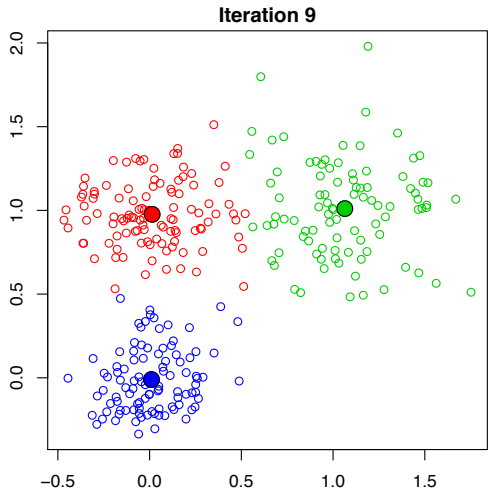
$n = 300,\ K = 3$





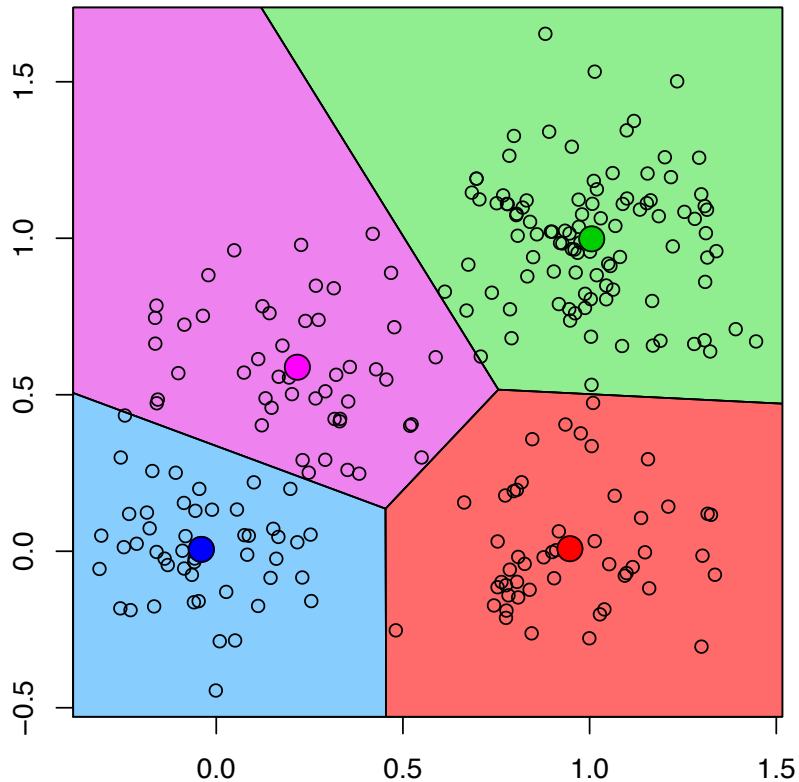






Voronoi tessellation

- Given cluster centers, we identify each point to its nearest center. This defines a Voronoi tessellation



- Given the centers $c_1, \ldots, c_K$, we define the Voronoi sets

$$V_k = \{x : \|x - c_k\|_2^2 \leq \|x - c_j\|_2^2, j = 1, \ldots, K\}$$

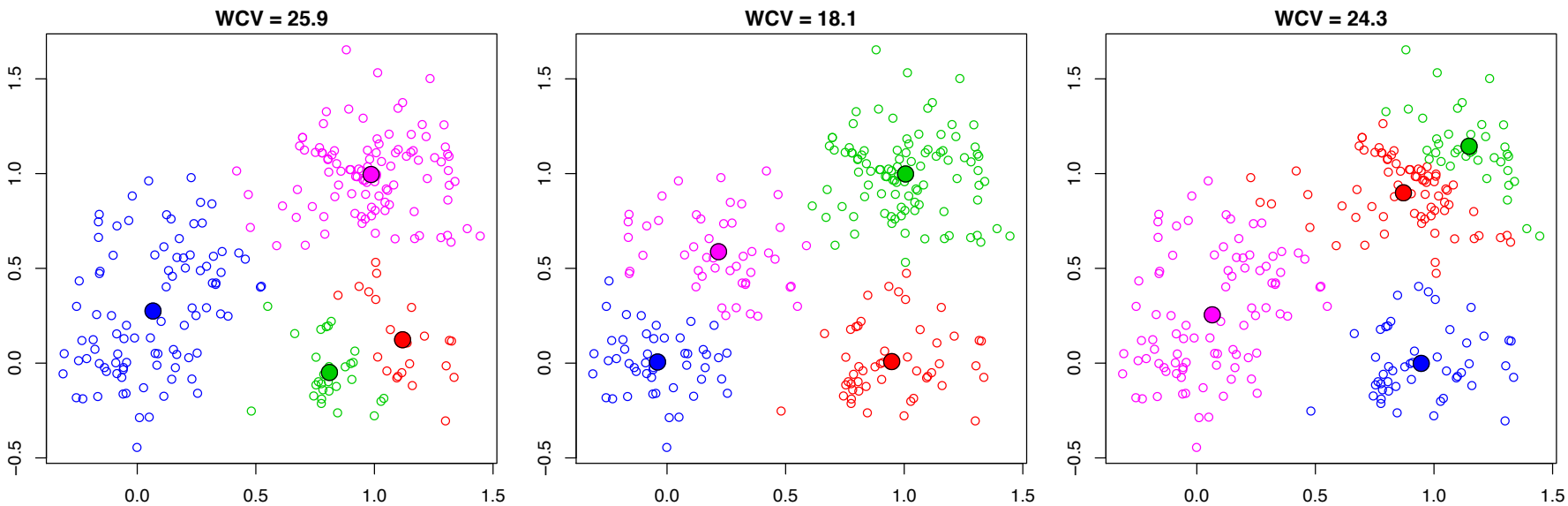
These are convex polyhedra.

Properties of K-means

- Within-cluster variation decreases with each iteration of the algorithm.
- The algorithm always converges, no matter the initial cluster centers. In fact, it take less than K^n iterations.
- The final clustering depends on the initial cluster centers.
  - different initial centers lead to very different results
  - run K-means multiple times, randomly initializing cluster centers for each run
  - choose the one gives the smallest within-cluster variation
- The algorithm is not guaranteed to deliver the clustering that globally minimizes within-cluster variation

K-means example, multiple runs

$n = 250$,and $K = 4$, the points are not as well-separated



We choose the second collection of centers because it yields the smallest within-cluster variation

Vector quantization

- Left: original image; middle: using 23.9% of the storage; right: using 6.25% of the storage

- K-means is often called "Lloyd's algorithm" in computer science and engineering, and is used in vector quantization for compression
- Basic idea: run K-means clustering on pixels in an image, and keep only the clusters and labels. Smaller K means more compression

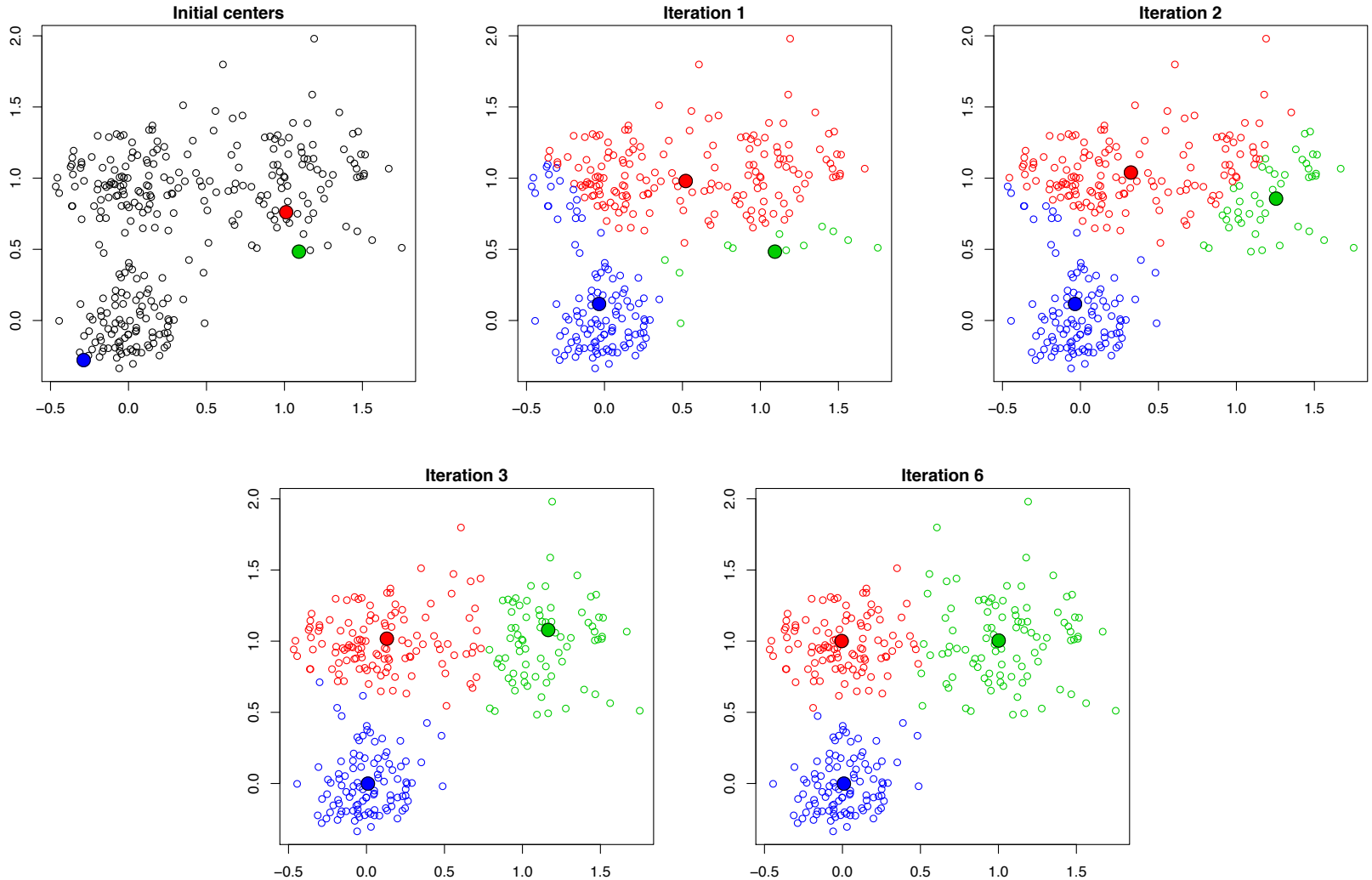
In K-means, cluster centers are averages

- A cluster center is representative for all points in a cluster, also called a prototype
- In K-means, we simply take a cluster center to be the average of points in the cluster. Great for computational purposes — but how does it lend to interpretation?
- This would be fine if we were clustering, e.g., houses in Pittsburgh based on features like price, square footage, number of bedrooms, distance to nearest bus stop, etc.
- Not so if we were clustering images.

K-MEDOIDS ALGORITHM

- In some applications we want each center to be one of the points itself.
- Medoid of a cluster is the point x_i in cluster k that minimizes $\sum_{C(j)=k} \|x_j - x_i\|_2^2$.
- K-medoids: similar to the K-means algorithm, except when fitting the centers $c_1, \ldots, c_K$, we restrict our attention to the points themselves
  1. Cluster (label) each point based on the closest center
  2. Replace each center by the medoid of points in its cluster

K-MEDOIDS EXAMPLE



Note: only 3 points had different labels under K-means

Properties of K-medoids

- The K-medoids algorithm shares the properties of K-means
  - each iteration decreases the criterion
  - the algorithm always converges
  - different starts gives different final answers
  - it does not achieve the global minimum
- K-medoids generally returns a higher value of within-cluster variation.
- K-medoids is computationally harder than K-means
- K-medoids has the (potentially important) property that the centers are located among the data points themselves

How to choose K?

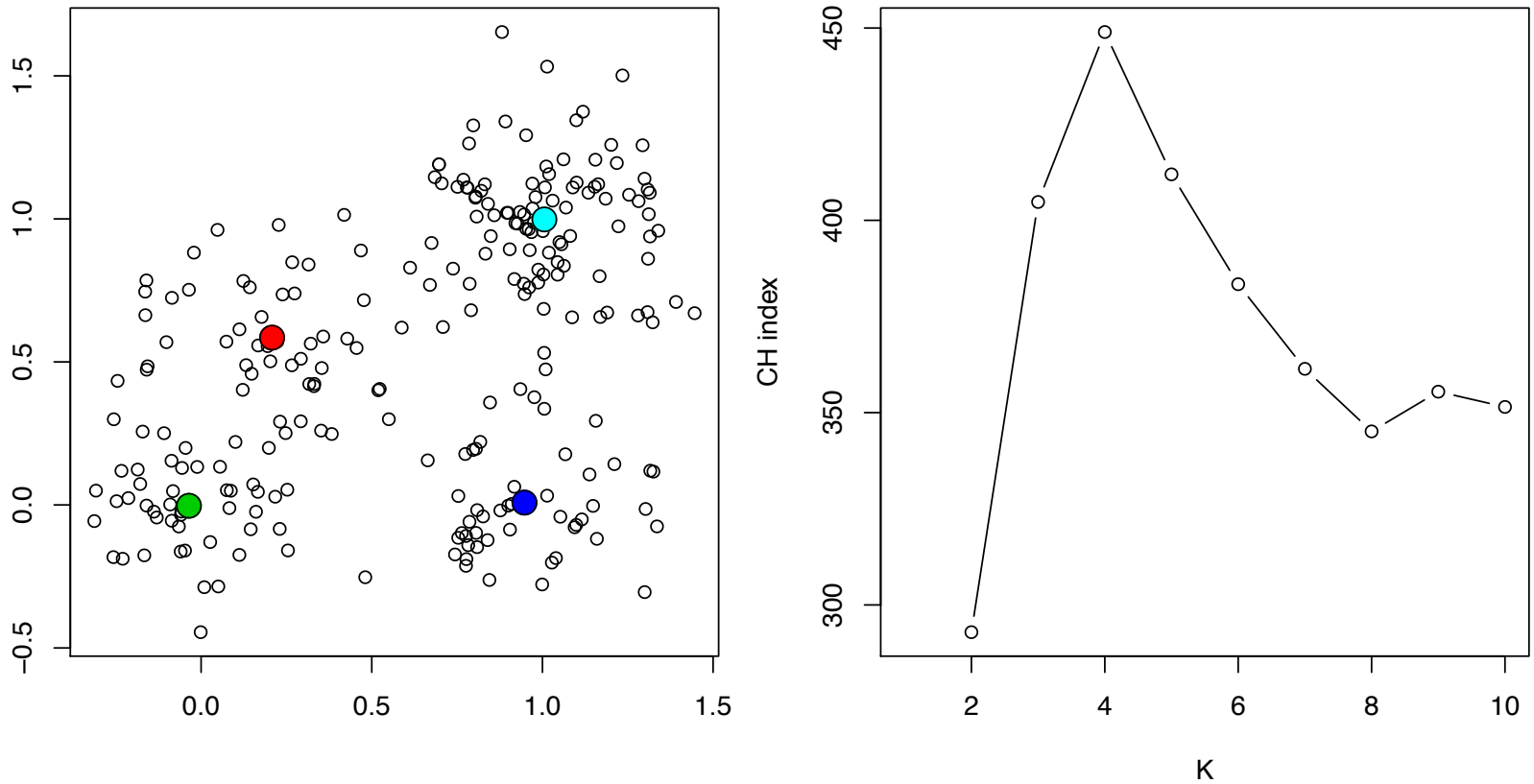
- Determining the number of clusters is a hard task for humans to perform
- It is also hard to explain what it is we're looking for
- It might mean a big difference scientifically if we were convinced that there were $K = 2$ subtypes of breast cancer vs. $K = 3$ subtypes
- One of the goals of data mining/statistical learning is automatic inference, e.g., choosing K.
- CH Index

$$\mathrm{CH}(K) = \frac{B(K)/(K-1)}{W(K)/(n-K)}$$

  where $B(K)$ is the between cluster variation $\sum_{k=1}^{K} n_k \|\bar{x}_k - \bar{x}\|_2^2$, $W(K)$ is the within cluster variation and $\bar{x}$ is the overall average.
- choose the value of K with the largest CH index

Example: CH index

- $n = 250, p = 2, K = 2, \ldots, 10$.



- We would choose $K = 4$ clusters, which seems reasonable
- General problem: the CH index is not defined for $K = 1$. We could never choose just one cluster (the null model)!

From K-means to hierarchical clustering

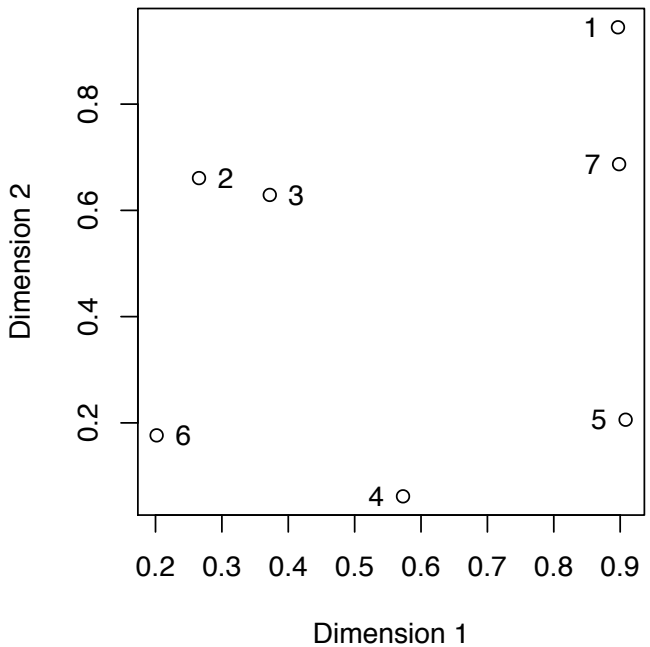
- Recall two properties of K-means (K-medoids) clustering:
  - It fits exactly K clusters (as specified)
  - Final clustering assignment depends on the chosen initial cluster centers
- Given pairwise dissimilarites d_{ij} between data points, hierarchical clustering produces a consistent result, without the need to choose initial starting positions (number of clusters)
- The catch: we need to choose a way to measure the dissimilarity between groups, called the linkage
- Given the linkage, hierarchical clustering produces a sequence of clustering assignments. At one end, all points are in their own cluster, at the other end, all points are in one cluster

Agglomerative vs divisive

- Two types of hierarchical clustering algorithms
- Agglomerative (i.e., bottom-up):
  - Start with all points in their own group
  - Until there is only one cluster, repeatedly: merge the two groups that have the smallest dissimilarity
- Divisive (i.e., top-down):
  - Start with all points in one cluster
  - Until all points are in their own cluster, repeatedly: split the group into two resulting in the biggest dissimilarity
- We will focus on agglomerative strategies today.

Simple example

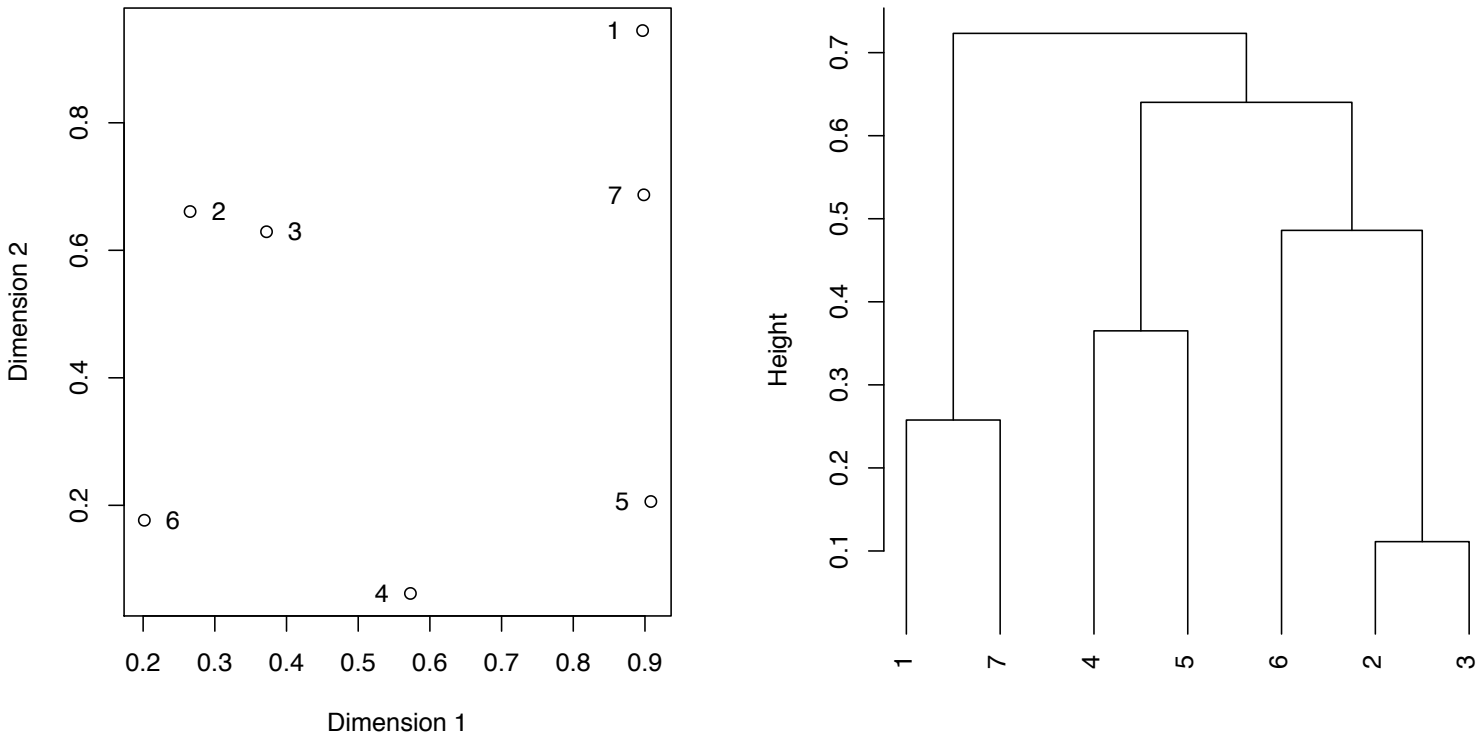
- Given these data points, an agglomerative algorithm might decide on a clustering sequence as follows:



Step 1: {1}, {2}, {3}, {4}, {5}, {6}, {7};
Step 2: {1}, {2, 3}, {4}, {5}, {6}, {7};
Step 3: {1, 7}, {2, 3}, {4}, {5}, {6};
Step 4: {1, 7}, {2, 3}, {4, 5}, {6};
Step 5: {1, 7}, {2, 3, 6}, {4, 5};
Step 6: {1, 7}, {2, 3, 4, 5, 6};
Step 7: {1,2,3,4,5,6,7}.

Dendrogram

- We can also represent the sequence of clustering assignments as a dendrogram:



What's a dendrogram?

- Dendrogram: convenient graphic to display a hierarchical sequence of clustering assignments. This is simply a tree where:
  - Each node represents a group
  - Each leaf node is a singleton (i.e., a group containing a single data point)
  - Root node is the group containing the whole data set
  - Each internal node has two daughter nodes (children), representing the the groups that were merged to form it
- the choice of linkage determines how we measure dissimilarity between groups of points
- If we fix the leaf nodes at height zero, then each internal node is drawn at a height proportional to the dissmilarity between its two daughter nodes

Linkages

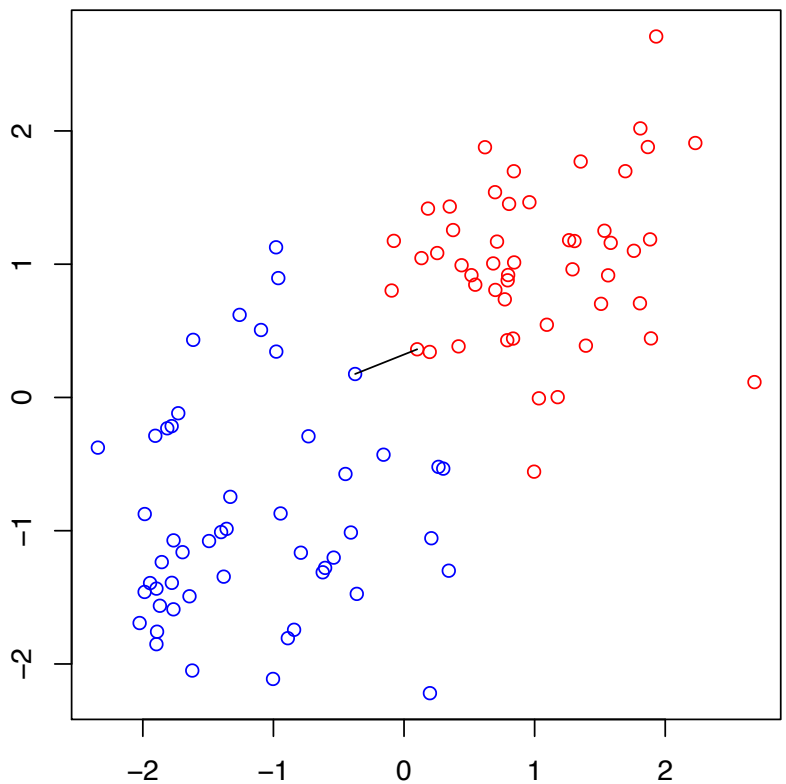
- Given n points, $x_1, \ldots, x_n$ and dissimilarities d_{ij} between x_i and x_j. (E.g. $x_i \in \mathbb{R}^p$, $d_{ij} = \|x_i - x_j\|_2$)
- At any level, clustering assignments can be expressed by sets $G = \{i_1, i_2, \ldots, i_r\}$, giving indices of points in this group.
  - Let n_G be the size of G (here $n_G = r$).
  - Bottom level: each group looks like $G = \{i\}$
  - top level: only one group, $G = \{1, \ldots, n\}$
- Linkage: function $d(G, H)$ that takes two groups G, H and returns a dissimilarity score between them
- Agglomerative clustering, given the linkage:
  - Start with all points in their own group
  - Until there is only one cluster, repeatedly: merge the two groups G, H such that $d(G, H)$ is smallest

Single linkage

- In single linkage (i.e., nearest-neighbor linkage), the dissimilarity between G, H is the smallest dissimilarity between two points in opposite groups:

$$d_{\text{single}}(G, H) = \min_{i \in G, j \in H} d_{ij}$$

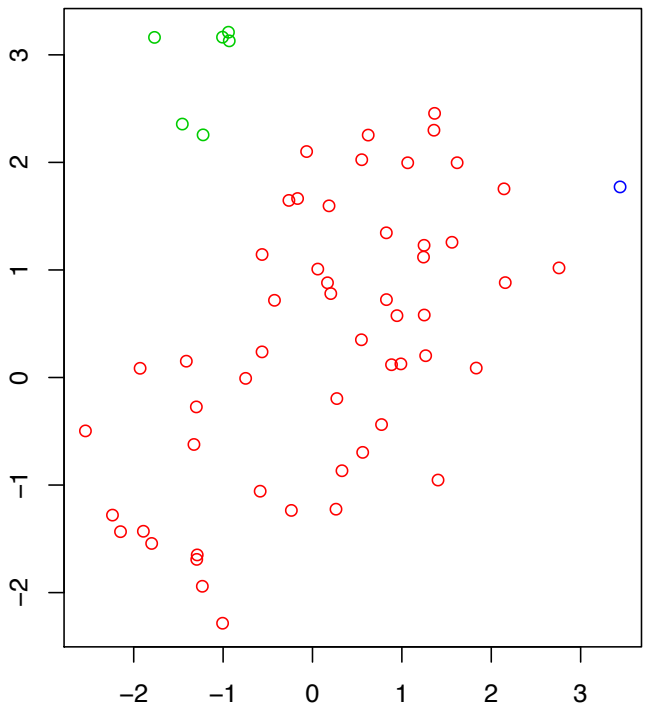
-

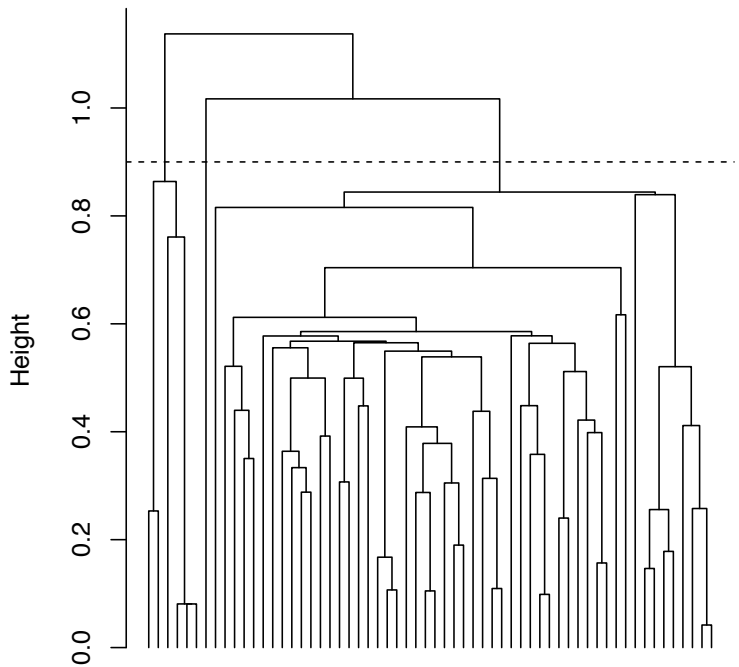


Example (dissimilarities d_{ij} are distances, groups are marked by colors): single linkage score $d_{\text{single}}(G, H)$ is the distance of the closest pair

Single linkage example

- $n = 60$, d_{ij}: euclidean distance
- Cutting the tree at $h = 0.9$ gives the clustering assignments marked by colors





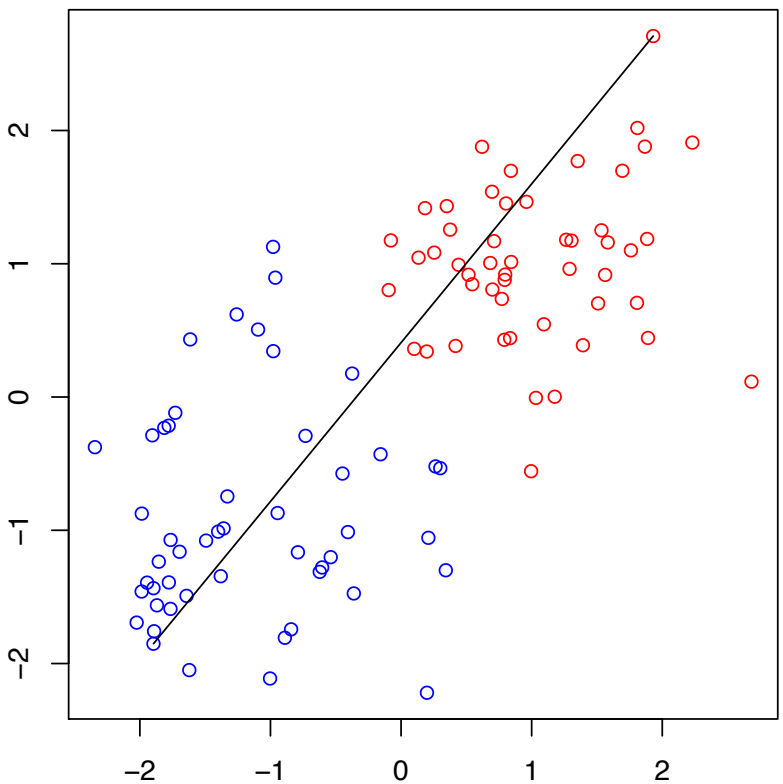
- Cut interpretation: for each point x_i, there is another point x_j in its cluster with $d_{ij} \leq 0.9$

Complete linkage

- In complete linkage (i.e., furthest-neighbor linkage), dissimilarity between G, H is the largest dissimilarity between two points in opposite groups:

$$d_{\text{complete}}(G, H) = \max_{i \in G, j \in H} d_{ij}$$

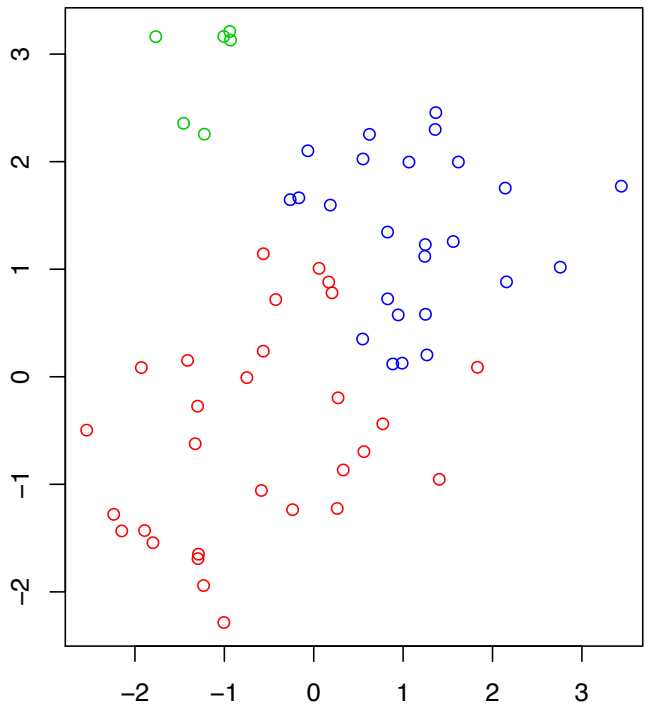
-

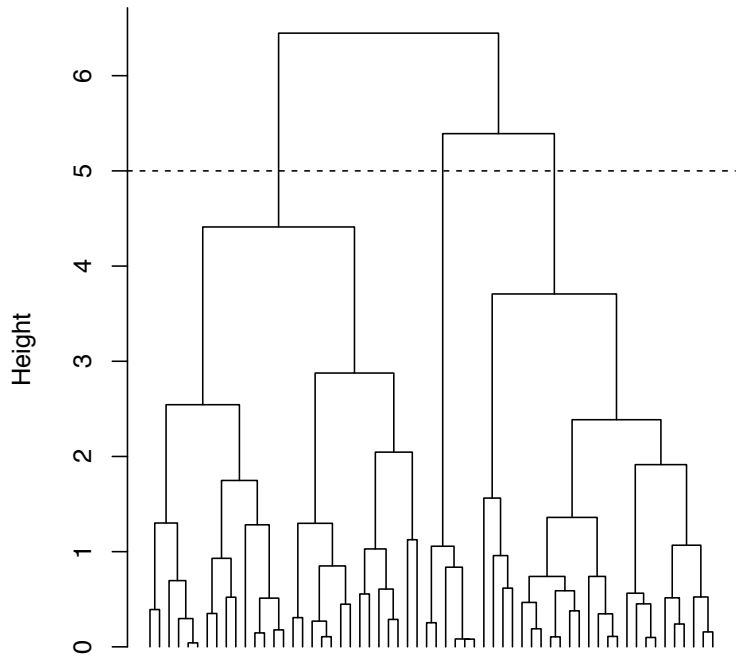


Example (dissimilarities d_{ij} are distances, groups are marked by colors): complete linkage score $d_{\text{complete}}(G, H)$ is the distance of the furthest pair

Complete linkage example

- Same data as before. Cutting the tree at $h = 5$ gives the clustering assignments marked by colors
-



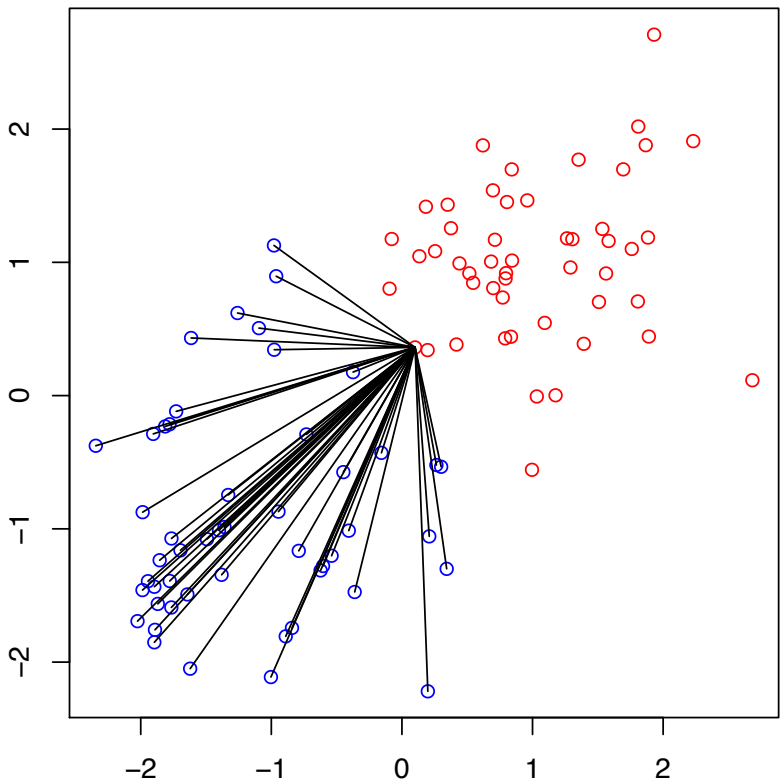


- Cut interpretation: for each point x_i, every other point x_j in its cluster satisfies $d_{ij} \leq 5$

Average linkage

In average linkage, the dissimilarity between G, H is the average dissimilarity over all points in opposite groups:

$$d_{\text{average}}(G, H) = \frac{1}{n_G n_H} \sum_{i \in G, j \in H} d_{ij}$$

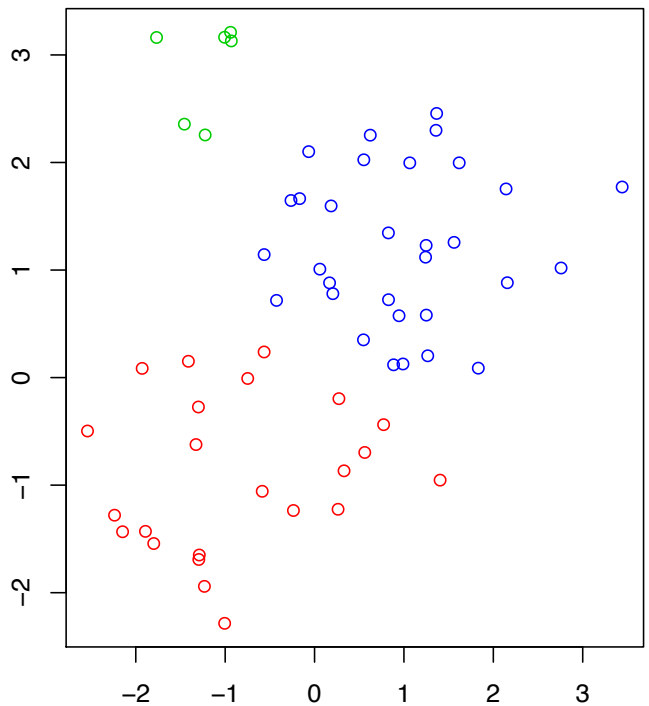


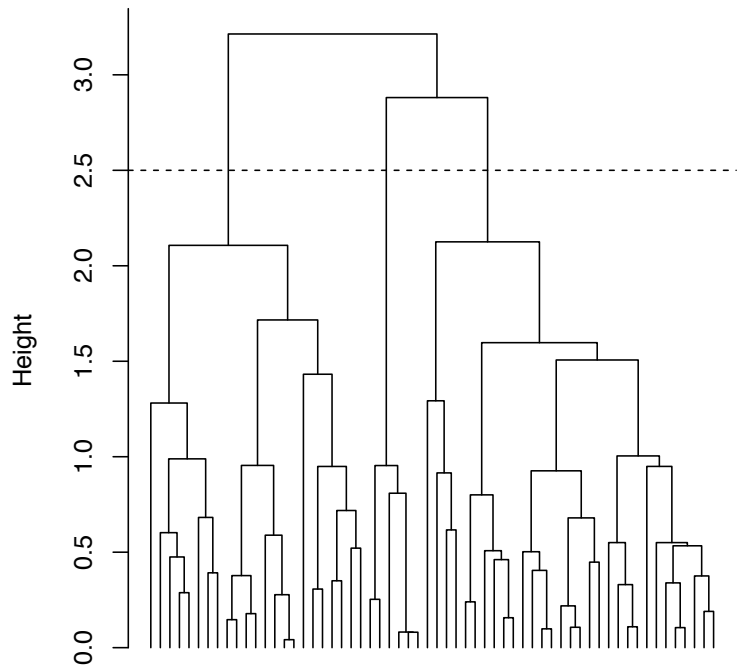
Example (dissimilarities d_{ij} are distances, groups are marked by colors): average linkage score $d_{\text{average}}(G, H)$ is the average distance across all pairs

(Plot here only shows distances between the blue points and one red point)

Average linkage example

- Same data as before. Cutting the tree at $h = 1.5$ gives the clustering assignments marked by colors
-



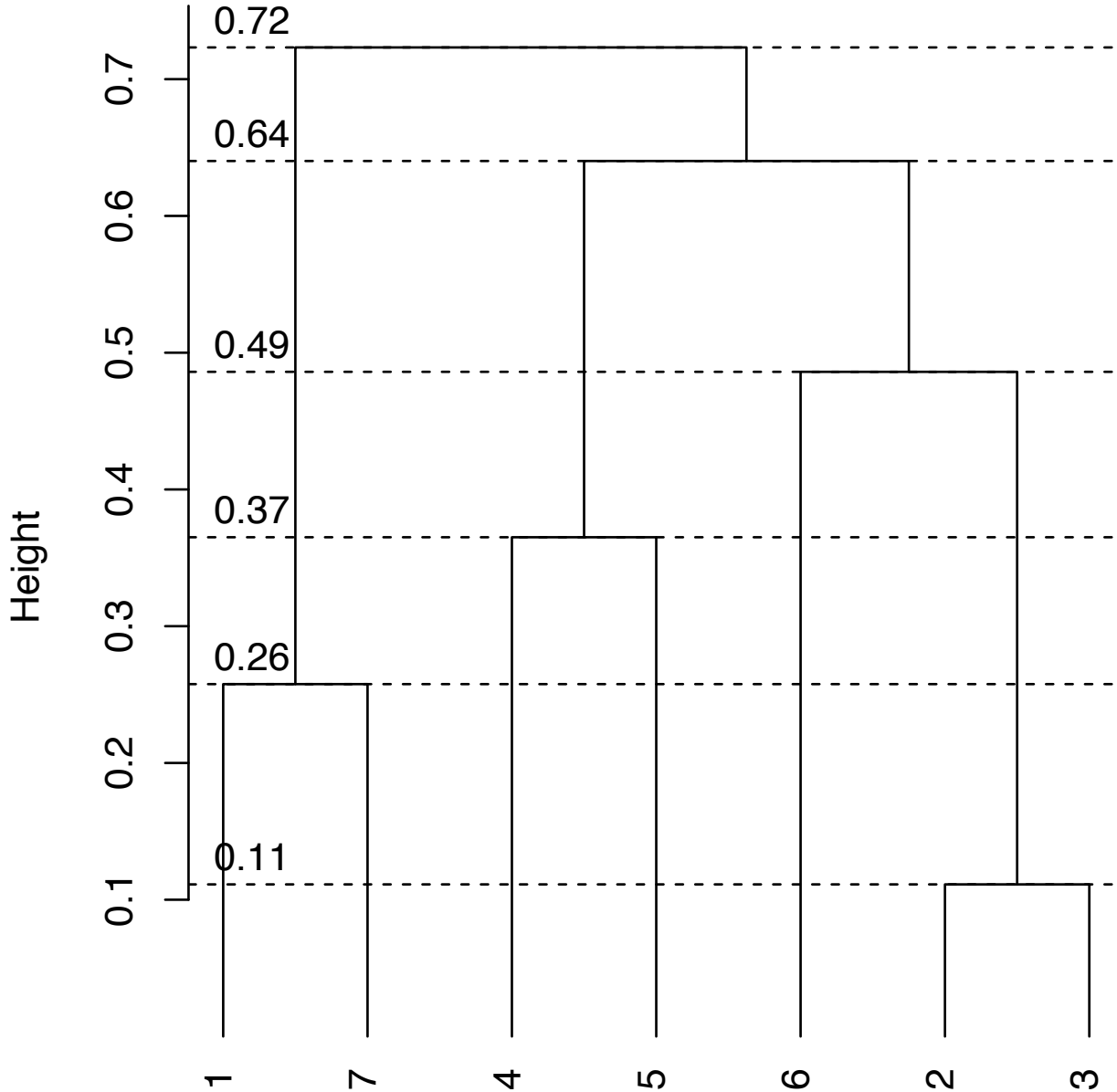


- Cut interpretation: difficult to interpret

Common properties

- Single, complete, average linkage share the following properties: These linkages operate on dissimilarities d_{ij}
  - don't need the points $x_1, \ldots, x_n$ to be in Euclidean space
  - Running agglomerative clustering with any of these linkages produces a dendrogram with no inversions
- Second property, in words: disimilarity scores between merged clusters always increases as we run the algorithm
- Means that we can draw a proper dendrogram, where the height of a parent is always higher than height of its daughters

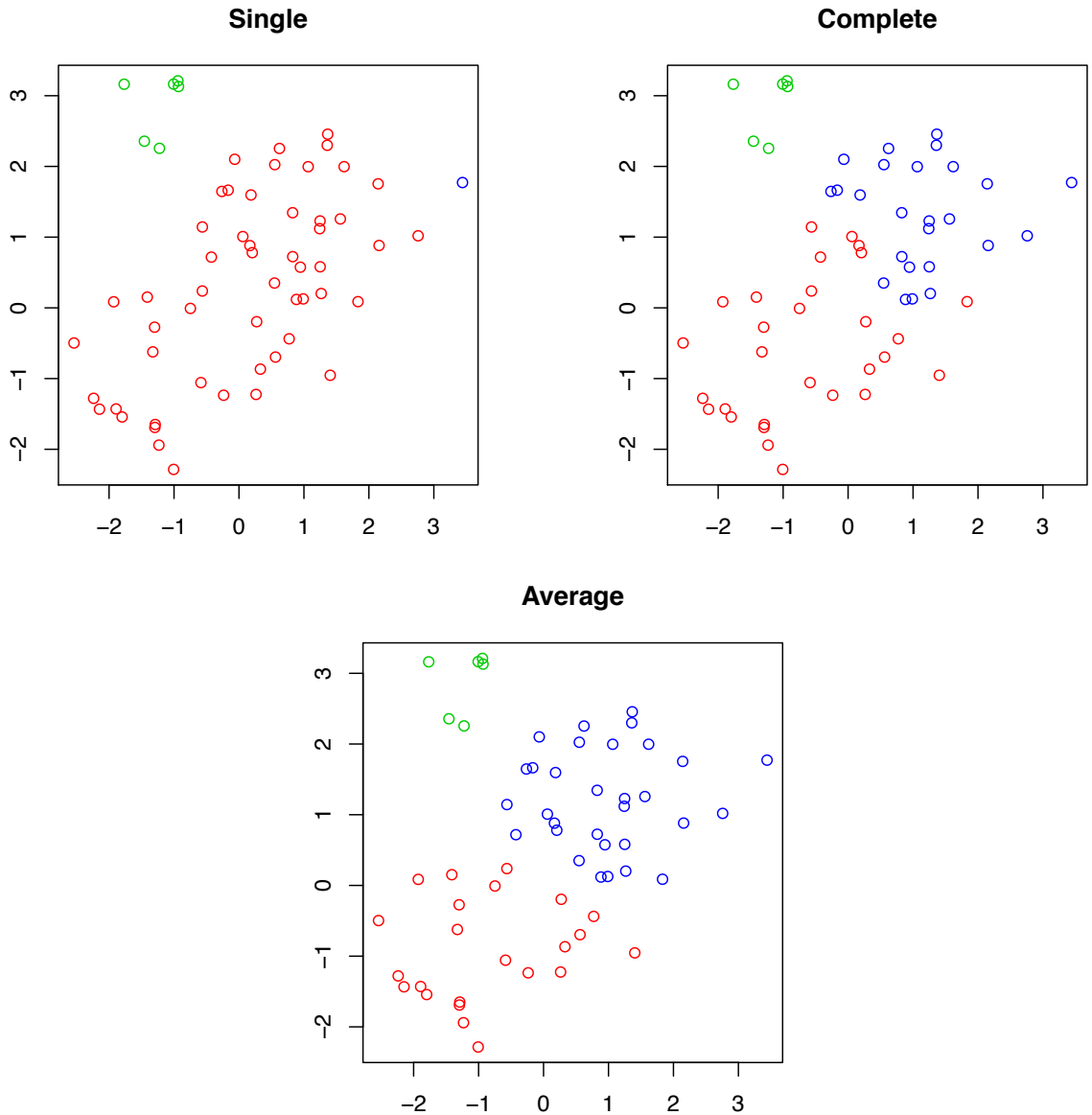
Example of a dendrogram with no inversions



Shortcomings of single, complete linkage

- Single and complete linkage can have some practical problems:
  - Single linkage suffers from chaining. In order to merge two groups, only need one pair of points to be close, irrespective of all others. Therefore clusters can be too spread out, and not compact enough
  - Complete linkage avoids chaining, but suffers from crowding. Because its score is based on the worst-case dissimilarity between pairs, a point can be closer to points in other clusters than to points in its own cluster. Clusters are compact, but not far enough apart
- Average linkage tries to strike a balance. It uses average pairwise dissimilarity, so clusters tend to be relatively compact and relatively far apart

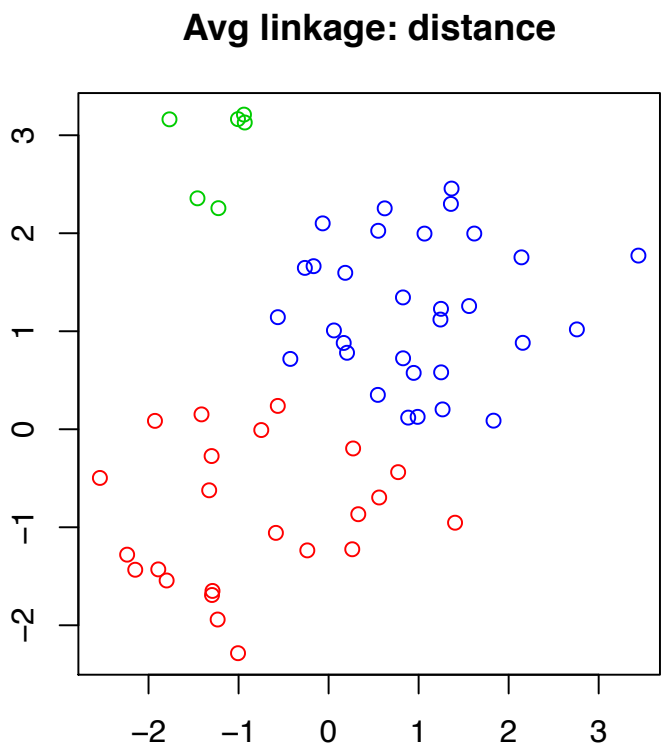
Example of chaining and crowding

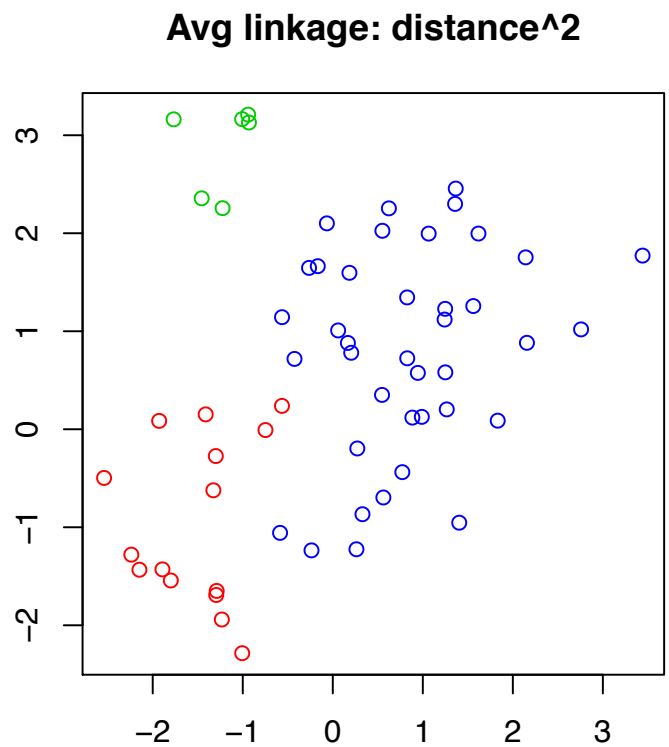


Shortcomings of average linkage

- It is not clear what properties the resulting clusters have when we cut an average linkage tree at given height h.
- Single and complete linkage trees each had simple interpretations
- Results of average linkage clustering can change with a monotone increasing transformation of dissimilarities d_{ij}.
- Depending on the context, it may be important or unimportant. It could be very clear/unclear what dissimilarities should be used or not

Example of a change with monotone increasing transformation

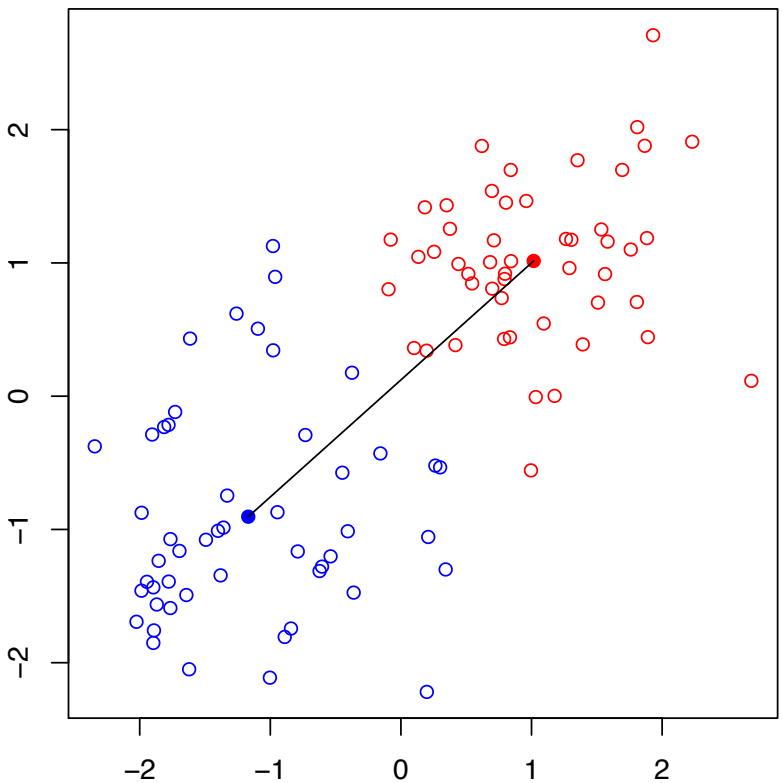




Centroid linkage

In Centroid linkage, the dissimilarity between G, H is the distance between the group averages of G and H

$$d_{\text{centroid}}(G, H) = \|\bar{x}_G - \bar{x}_H\|_2$$

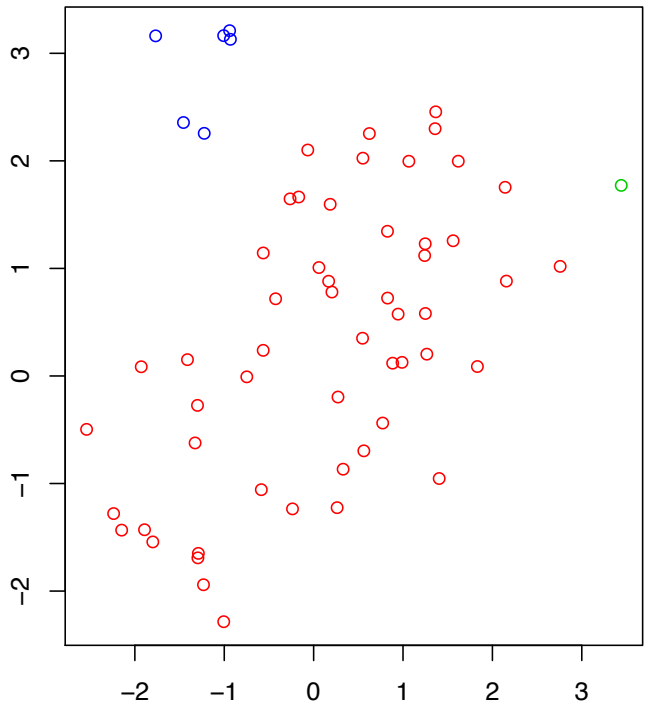


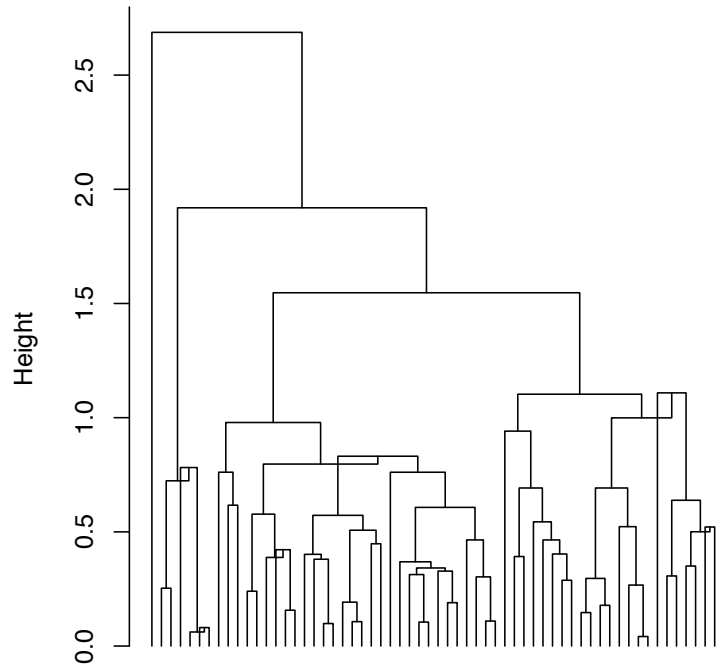
Example (dissimilarities d_{ij} are distances, groups are marked by colors): centroid linkage score $d_{\text{centroid}}(G, H)$ is the distance between the group centroids (i.e., group averages)

Centroid linkage is simple: easy to understand, and easy to implement. It has become the standard for hierarchical clustering in biology

Centroid linkage example

- Same data as before. Cutting the tree at some heights wouldn't make sense because the dendrogram has inversions.
-





- Cut interpretation: no!

Shortcomings of centroid linkage

- Can produce dendrograms with inversions, which really messes up the visualization
- Even if were we lucky enough to have no inversions, still no interpretation for the clusters resulting from cutting the tree
- Answers change with a monotone transformation of the dissimilarity measure

Ward's linkage

- Ward's linkage says that the distance between two clusters, G and H, is how much the sum of squares will increase when we merge them

$$d^2_{\text{ward}}(G,H) = \sum_{i \in G \cup H} \|x_i - \bar{x}_{G\cup H}\|_2^2 - \sum_{i \in G} \|x_i - \bar{x}_G\|_2^2 - \sum_{i \in H} \|x_i - \bar{x}_H\|_2^2$$
$$= \frac{n_G n_H}{n_G + n_H} \|\bar{x}_G - \bar{x}_H\|_2^2$$

- Similar to centroid linkage: easy to understand, easy to implement, also have nice statistical interpretation.
- But, it has no inversions!