

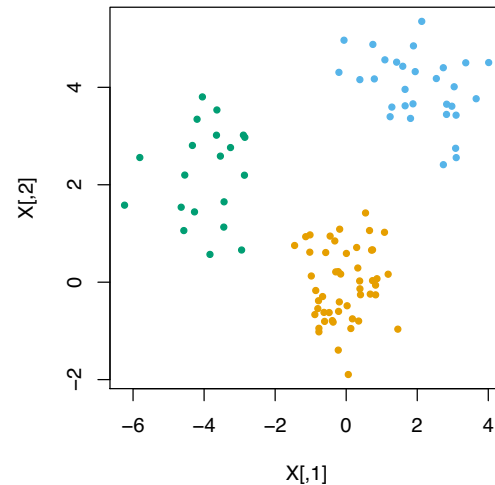
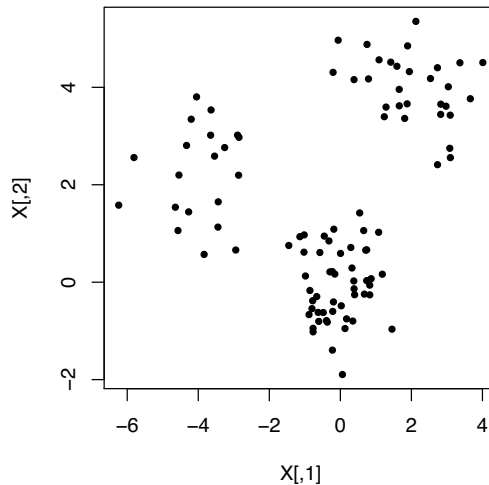
Unsupervised Statistical Learning: Hierarchical Clustering

Siva Balakrishnan
Data Mining: 36-462/36-662

March 26th, 2019

Recap: Clustering

- ▶ Divide n data points into K groups, where roughly points within group are more “similar” than points between groups.



- ▶ Lots of different uses, including data summarization, data compression, helping downstream supervised learning tasks and so on.

Recap: Clustering v/s classification

→ (pts, grps)

- ▶ In classification, we are given labeled data (i.e. grouped data) and want to learn a discriminator that works on new data.
- ▶ In clustering, we are just given the points, and want to learn the groups.
- ▶ Often the focus in classification is on *generalization*, i.e. we want to do well on new (test) data. In clustering, generalization is usually not the explicit goal – we just want to find interesting groups.

Recap: k-means

- ▶ One way to cluster data is sometimes called *objective based clustering*.
 1. We write down an intuitive way to measure the quality of a given clustering (our objective).
 2. We then try to derive an algorithm to find a good clustering (as measured by our objective).
- ▶ k -means is such a method.

Recap: k-means

- ▶ Given a set of points X_1, \dots, X_n and **dissimilarities** $d(X_i, X_j) = \|X_i - X_j\|_2^2$.
- ▶ Denote a clustering by a function C which maps data points to $\{1, \dots, K\}$.
- ▶ Three equivalent objectives:

1. **Within-cluster scatter:**

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

2. **Within-cluster variation:**

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

3. **Modified within-cluster variation:**

$$W(C) = \min_{c_1, \dots, c_K} \sum_{k=1}^K \sum_{i \in C_k} d(x_i, c_k)$$

- ▶ A good clustering has low value of these objectives.

Recap: Minimizing the objective – Lloyd's algorithm

- ▶ It is hard to minimize the k-means objective over all possible clusterings of the data. A popular heuristic is to use alternating minimization.

- ▶ We want to minimize

$$\sum_{k=1}^K \sum_{C(i)=k} \|X_i - c_k\|_2^2,$$

$\{c_1, \dots, c_k\}$ fixed.

over both clusterings C and $c_1, \dots, c_K \in \mathbb{R}^p$.

- ▶ Minimize it just over C (keep c_1, \dots, c_K fixed)?

$$C(i) = \arg \min_k d(X_i, c_k).$$

Minimize it just over c_1, \dots, c_K (keep C fixed)?

$$c_k = \frac{1}{n_k} \sum_{i \in C_k} X_i.$$

Recap: Lloyd's algorithm

We start with an initial guess for c_1, \dots, c_K (e.g., pick K points at random over the range of X_1, \dots, X_n), then repeat:

1. **Minimize over C** : for each $i = 1, \dots, n$, find the cluster center c_k closest to X_i , and let $C(i) = k$
2. **Minimize over c_1, \dots, c_K** : for each $k = 1, \dots, K$, let $c_k = \bar{X}_k$, the average of points in group k

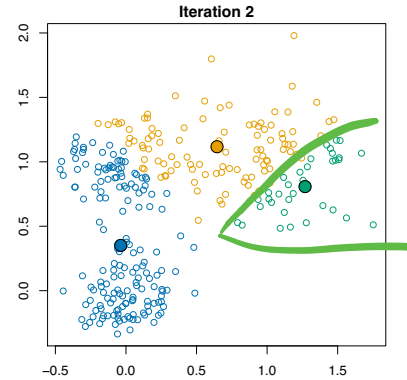
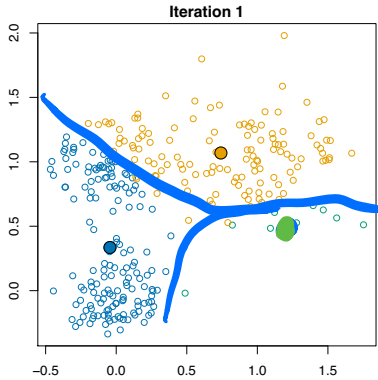
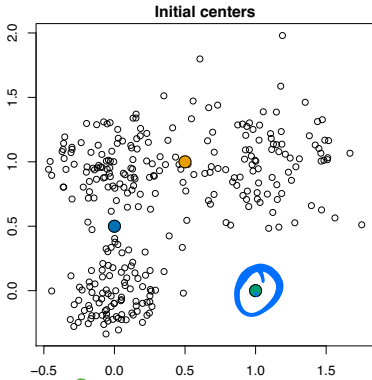
Stop when cluster assignments/within-cluster variation do not change.

In words:

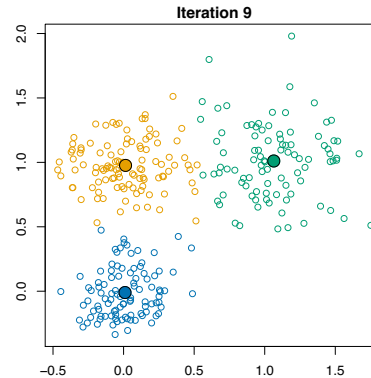
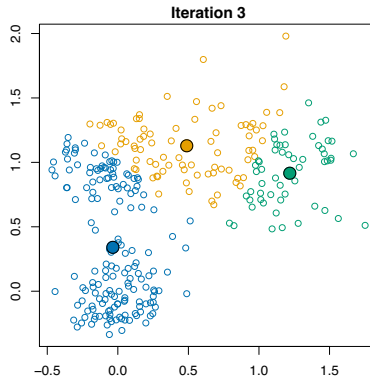
1. Cluster (label) each point based the closest center
2. Replace each center by the average of points in its cluster

Recap: K -means example

Here $X_i \in \mathbb{R}^2$, $n = 300$, and $K = 3$



K fixed.



Nice things: keeps decreasing $W(C)$, will eventually terminate
Some drawbacks: randomized, heuristic.

What are some things K -means lacks?

K -means is a famous, standard clustering algorithm. However, it lacks several potentially-desirable qualities:

- ▶ Ability to use other measures of dissimilarity
- ▶ “Interpretable” cluster centers
- ▶ Deterministic results
- ▶ Multi-level/scale view of clusters, Nested clusters.

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**?

Sometimes we prefer methods that return a **representative** item for the cluster, rather than an average. For example: a “typical” asset or company that is similar to all the other members of the cluster. This makes it easier to think about what the cluster means.

Suppose we were clustering documents. What does an “average” document mean? A typical document is more useful.

K-medoids algorithm

K-medoids clustering addresses the first two concerns. It make each center **one of the cluster points**. It also allows other dimilarities to be substituted.

Initial guess for centers c_1, \dots, c_K (e.g., randomly select K of the points X_1, \dots, X_n), then repeat:

I want c_1, \dots, c_K to be actual data pts.

1. **Minimize over C** : for each $i = 1, \dots, n$, find the cluster center c_k closest to X_i , and let $C(i) = k$
2. **Minimize over c_1, \dots, c_K** : for each $k = 1, \dots, K$, let $c_k = X_k^*$, the **medoid** of points in cluster k , i.e., the point X_i in cluster k that minimizes $\sum_{C(j)=k} \|X_j - X_i\|_2^2$

Stop when within-cluster variation doesn't change

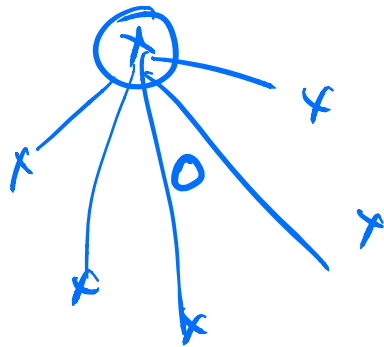
In words:

1. Cluster (label) each point based on the closest center
2. Replace each center by the medoid of points in its cluster

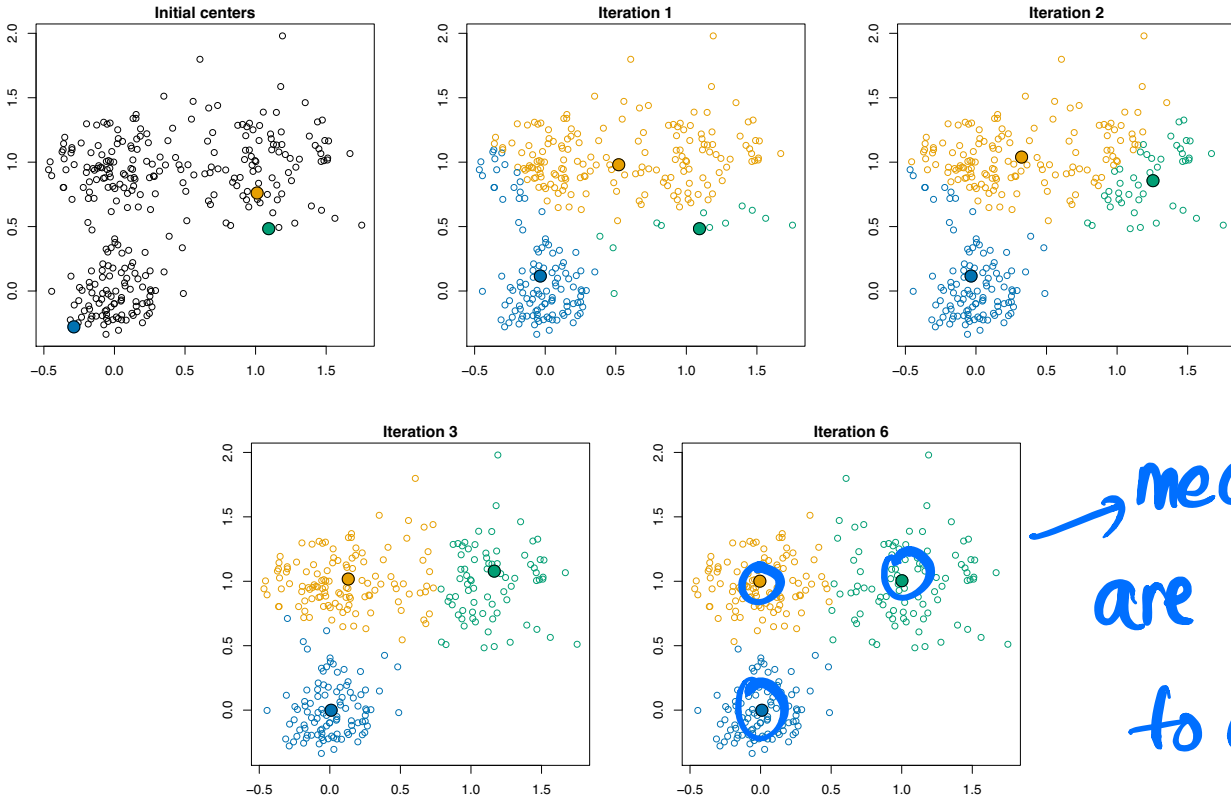
~~A~~ $X_1 \dots X_k$

mean: $\frac{1}{k} \sum_{i=1}^k X_i = \arg \min_c \sum \|X_i - c\|_2^2$

medoid: $X^* \in \{X_1, \dots, X_k\} \arg \min_i \sum \|X_i - X^*\|_2^2$



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 that we discussed (each iteration decreases the criterion; the algorithm always converges; different starts gives different final answers; it does not achieve the global minimum)

K -medoids returns centers that are actual data points.

K -medoids generally returns a **higher value** of $\sum_{k=1}^K \sum_{C(i)=k} \|X_i - c_k\|_2^2$ than does K -means (why?).

 K -medoids is **computationally harder** than K -means (because of step 2: computing the medoid is harder than computing the average)

From K -means to hierarchical clustering

Recall two properties of K -means (K -medoids) clustering:

1. Final clustering depends on the initial centers (does not find global max)
2. Fits exactly K clusters (for fixed K)

Suppose we cluster with $K = 3$, and then decide we want a more refined clustering with $K = 4$. The clusterings may have no relation. This makes tuning the level of clustering a bit odd.

Hierarchical clustering will produce a result that:

- ▶ Is deterministic, based on the distances $d_{ij} = d(x_i, x_j)$. *No random starts.*
- ▶ Describes the resulting clusterings across all levels of clustering (numbers of clusters K)
- ▶ Gives nested clusters as K changes!

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):

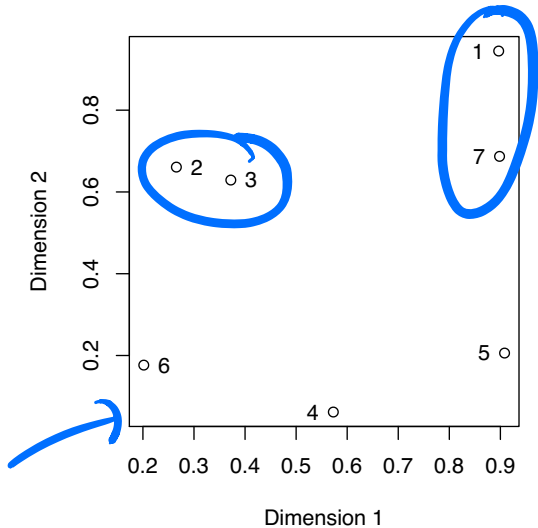
- ▶ Start with all points in one cluster
- ▶ Until all points are in their own cluster, repeatedly: split the group into two resulting in the biggest dissimilarity

We will focus on agglomerative strategies.

Recursive partitioning

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: $\{1, 7\}, \{2, 3\}, \{4\}, \{5\}, \{6\};$

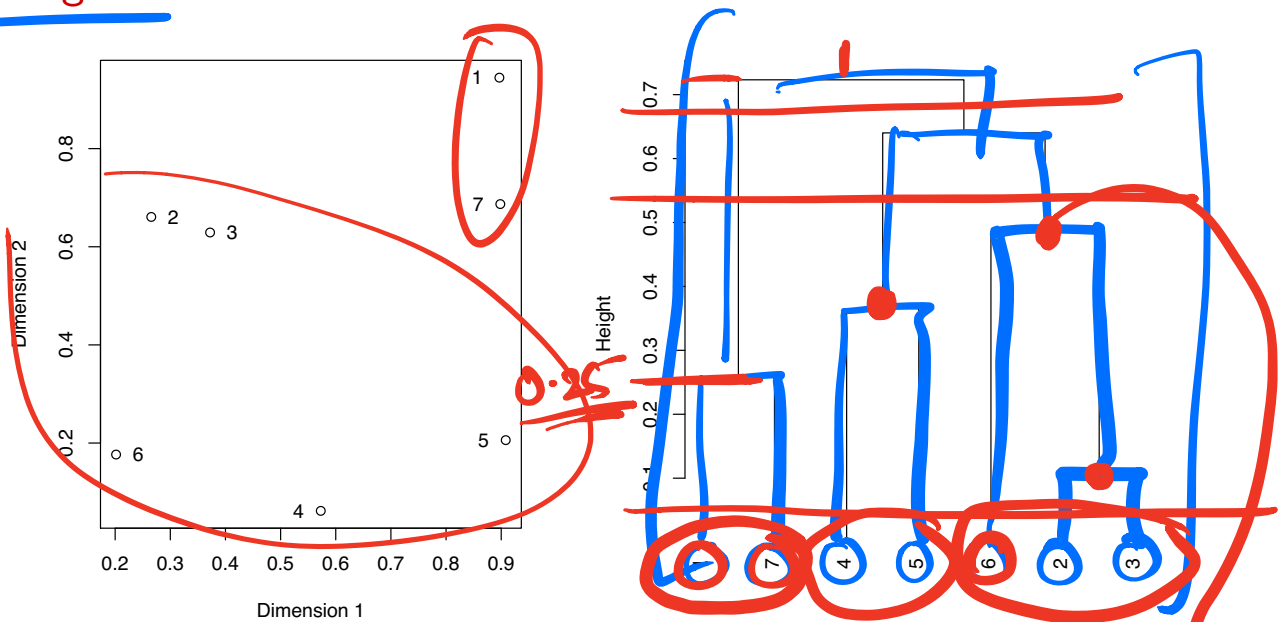
Step 4: $\{1, 7\}, \{2, 3\}, \{4, 5\}, \{6\};$

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

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

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

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



Note that **cutting the dendrogram** horizontally partitions the data points into clusters.

A dendrogram, however, contains more information than just the sequence of clusters. The height at which a pair of clusters is merged tells us how “similar” the clusters are (we’ll have lots more to say about this).

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 dissimilarity between its two daughter nodes.

How do we obtain a dendrogram?

- ▶ We go from the bottom-up. We just need to (recursively) decide which pair of groups to *merge*.
- ▶ Given points X_1, \dots, X_n , and **dissimilarities** d_{ij} between each pair X_i and X_j . (Think of $X_i \in \mathbb{R}^p$ and $d_{ij} = \|X_i - X_j\|_2$; note: this is the Euclidean distance, not squared distance).
- ▶ Initially, when every point is in its own group this is easy: we just merge the two closest points, i.e. the pair for which the dissimilarity is smallest.
- ▶ We can imagine continuing to do this, but what is the problem?

↳ we don't know how far groups are.

Linkages

At any level, clustering assignments can be expressed by sets $G = \{i_1, i_2, \dots, i_r\}$, 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, $G = \{1, \dots, n\}$

Linkage: function $d(G, H)$ that takes two groups G, H and returns a dissimilarity score between them *How far are G, H.*

Agglomerative clustering, given the linkage:

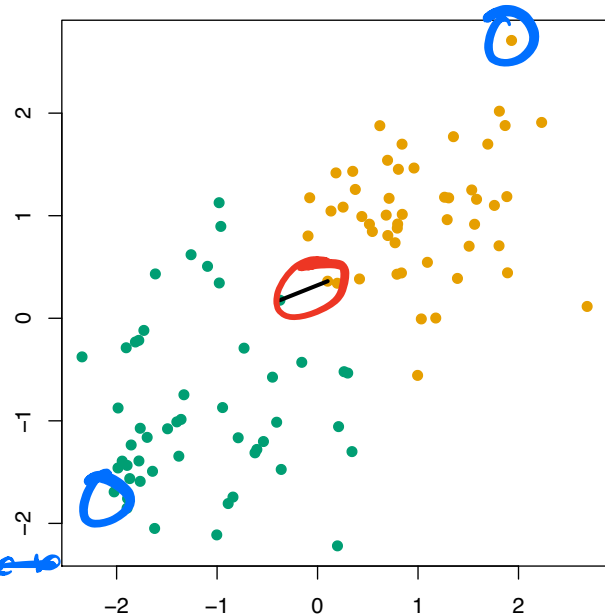
- ▶ Start with all points in their own group
- ▶ Until there is only one cluster, repeatedly: merge the two groups G, H such that $d(G, H)$ is smallest

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:

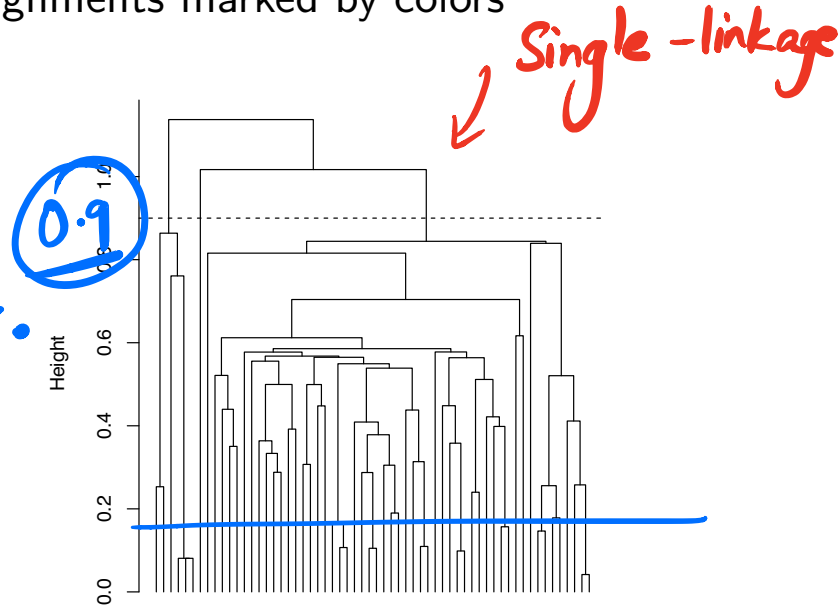
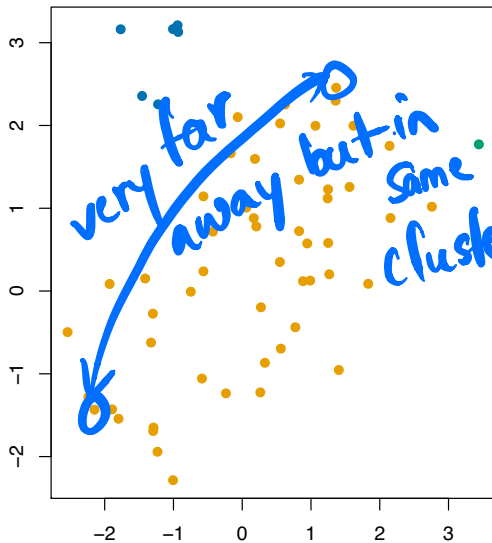
$$d_{\text{single}}(G, H) = \min_{i \in G, j \in H} d_{ij}$$

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

Here $n = 60$, $X_i \in \mathbb{R}^2$, $d_{ij} = \|X_i - X_j\|_2$. Cutting the tree at $h = 0.9$ gives the clustering assignments marked by colors



except
singleton
clusters

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

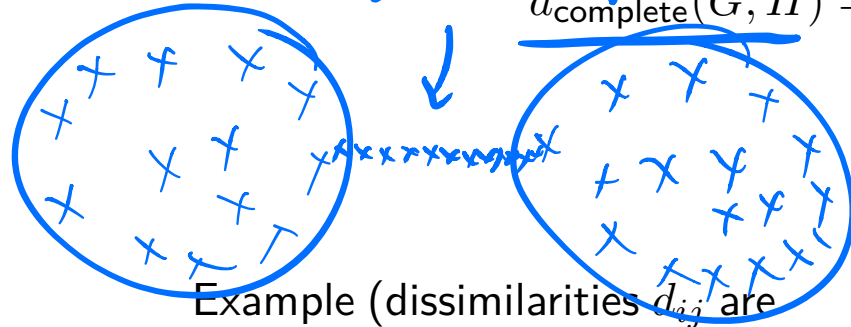
$\forall X_i \exists X_j$ within
dist 0.9.

Complete linkage

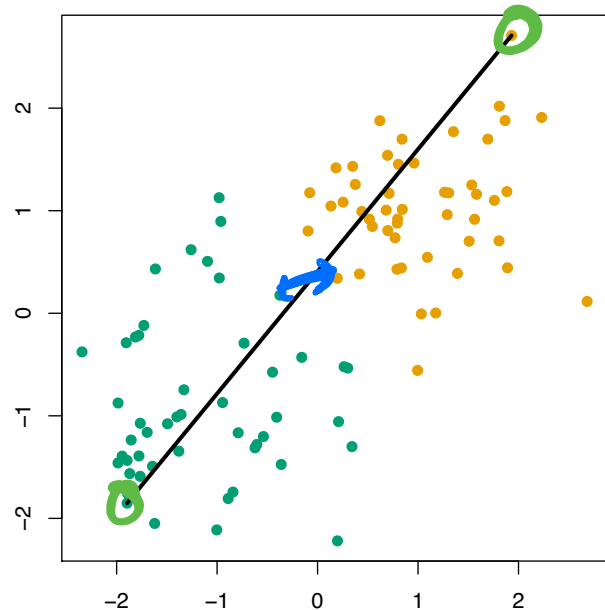
In **complete linkage** (i.e., furthest-neighbor linkage), dissimilarity between G, H is the largest dissimilarity between two points in opposite groups:

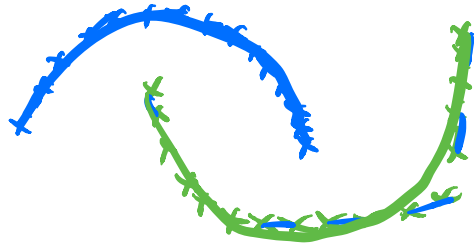
bad example for single-linkage

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



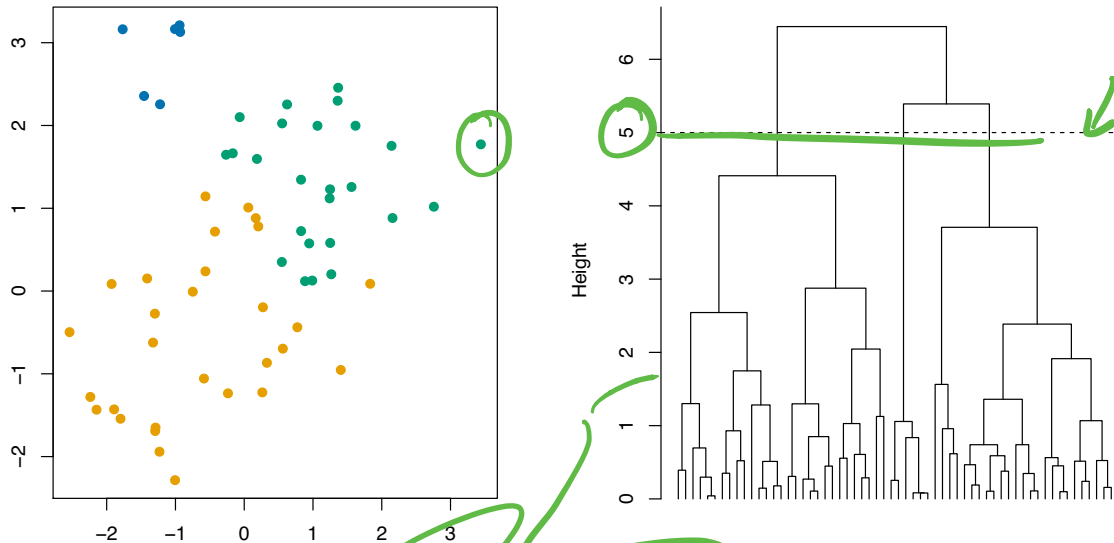
Example (dissimilarities d_{ij} are distances, groups are marked by colors): complete linkage score $d_{\text{complete}}(G, H)$ is the distance of the **furthest pair**





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

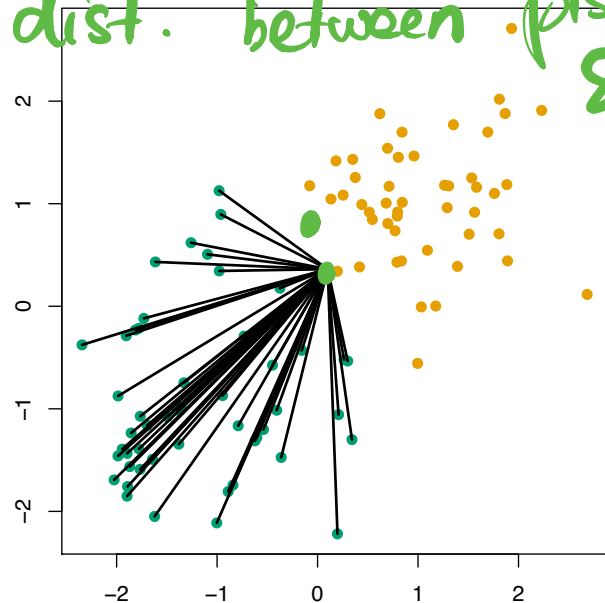
In **average linkage**, the dissimilarity between G, H is the average dissimilarity over all points in opposite groups:

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

Expected/average dist. between pts in G & H .

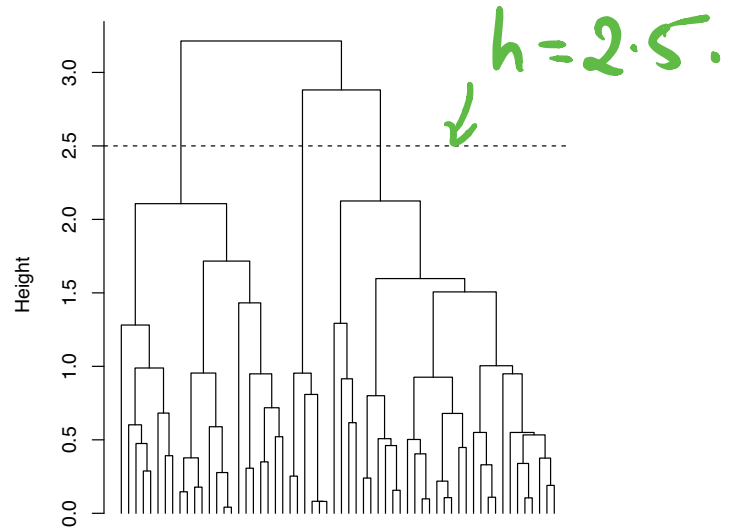
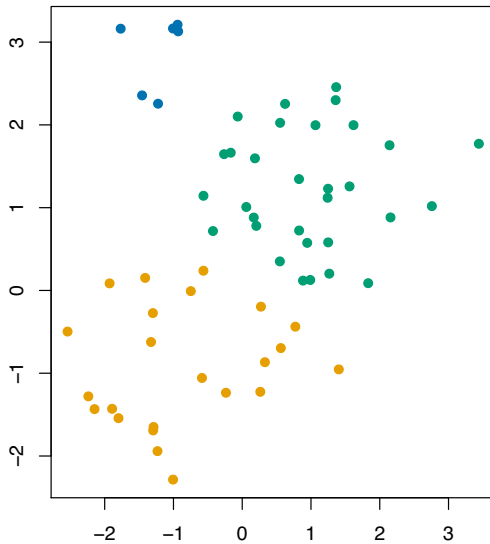
Example (dissimilarities d_{ij} are distances, groups are marked by colors): average linkage score $d_{\text{average}}(G, H)$ is the **average distance** across all pairs

(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 clustering assignments marked by the colors



→ **Cut interpretation:** there really isn't a good one!

Common properties



Single, complete, average linkage share the following properties:

- ▶ These linkages operate on **dissimilarities** d_{ij} , and don't need the points X_1, \dots, 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: dissimilarity scores between merged clusters only **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

term, frequency vector:
'the', 'it', 'obama', 'caket'
[1000, 999, 1, 2000, ---]

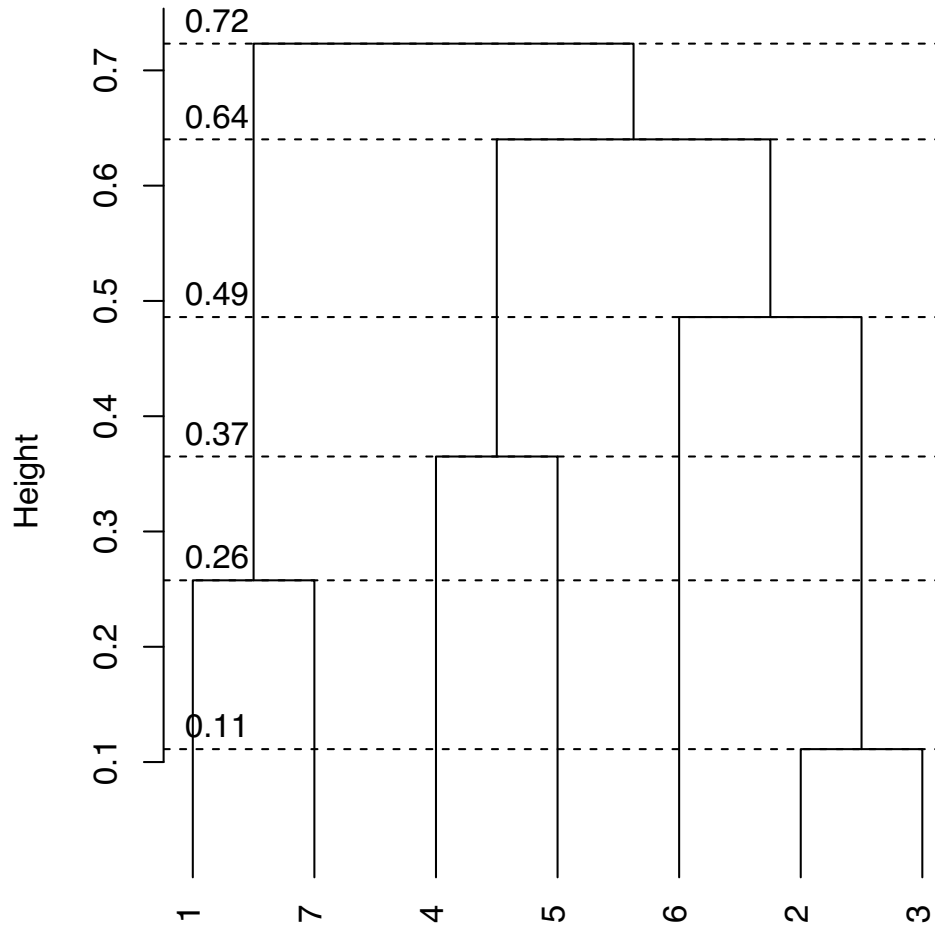
x_i, x_j

$\|x_i - x_j\|_2$ → not very good.

↳ other measures of dist./
similarity.

"cosine-similarity".

Example of a dendrogram with no inversions



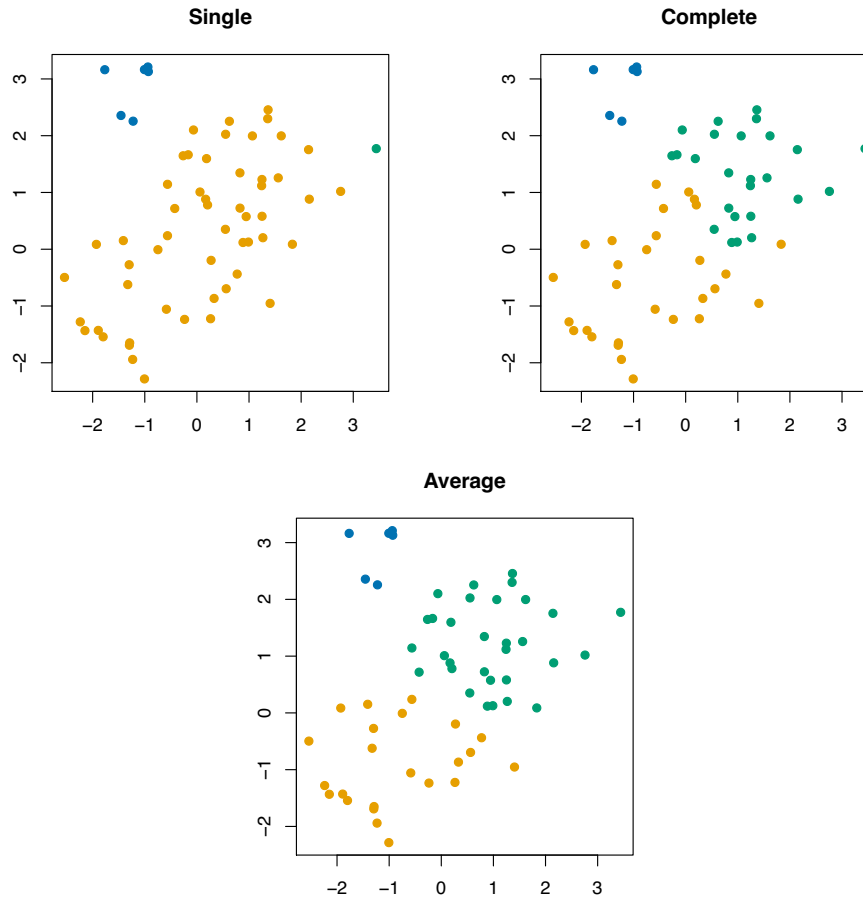
Shortcomings of single, complete linkage

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

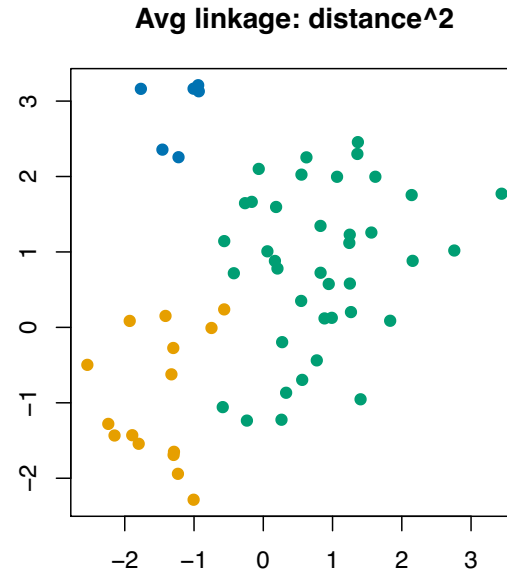
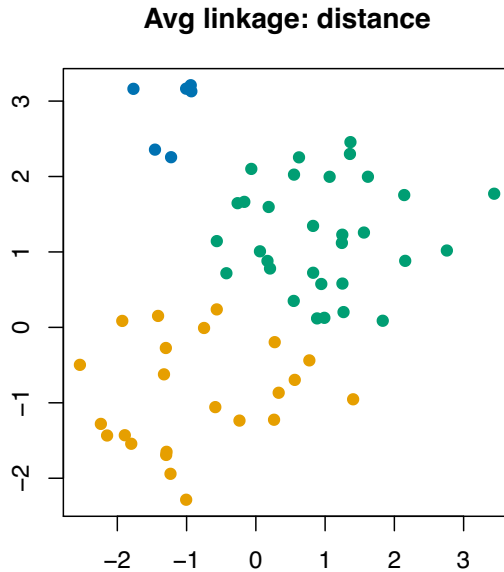
Average linkage isn't perfect, it has its own problems:

- ▶ 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} . I.e., if h is such that $h(x) \leq h(y)$ whenever $x \leq y$, and we used dissimilarities $h(d_{ij})$ instead of d_{ij} , then we could get different answers

Depending on the context, second problem may be important or unimportant. E.g., it could be very clear what dissimilarities should be used, or not

Note: results of single, complete linkage clustering are **unchanged** under monotone transformations

Example of a change with monotone increasing transformation



Recap: hierarchical agglomerative clustering

Hierarchical agglomerative clustering: start with all data points in their own groups, and repeatedly merge groups, based on linkage function. Stop when points are in one group (this is agglomerative; there is also divisive)

This produces a sequence of clustering assignments, visualized by a **dendrogram** (i.e., a tree). Each node in the tree represents a group, and its height is proportional to the dissimilarity of its daughters

Three most common linkage functions: **single, complete, average linkage**. Single linkage measures the least dissimilar pair between groups, complete linkage measures the most dissimilar pair, average linkage measures the average dissimilarity over all pairs

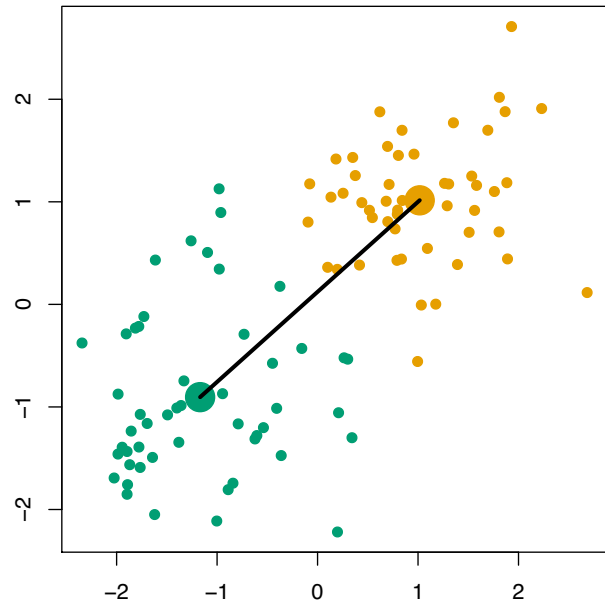
Each linkage has its strengths and weaknesses

Centroid linkage

Centroid linkage¹ is commonly used. Assume that $X_i \in \mathbb{R}^p$, and $d_{ij} = \|X_i - X_j\|_2$. Let \bar{X}_G, \bar{X}_H denote group averages for G, H . Then:

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

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)

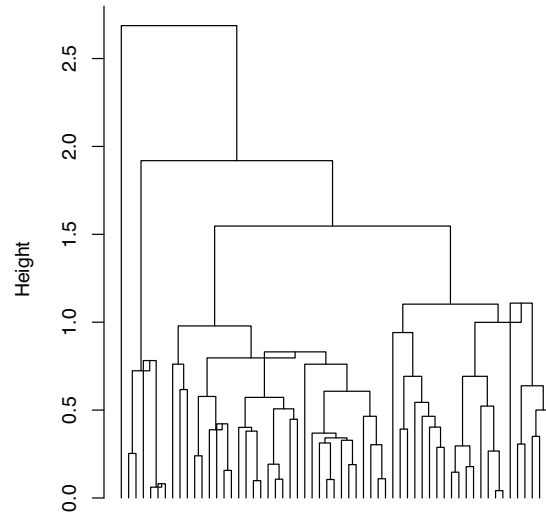
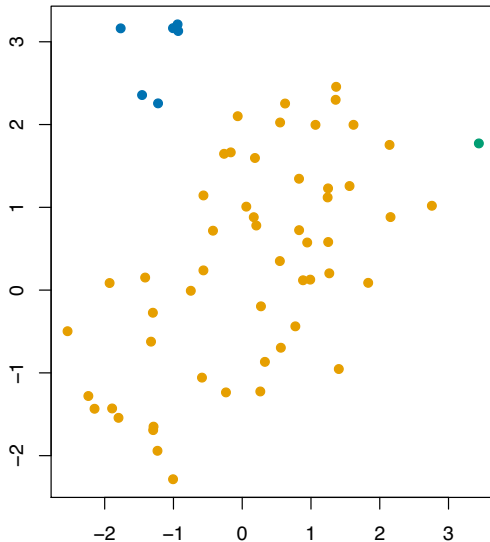


¹Eisen et al. (1998), "Cluster Analysis and Display of Genome-Wide Expression Patterns"

Centroid linkage is the standard in biology

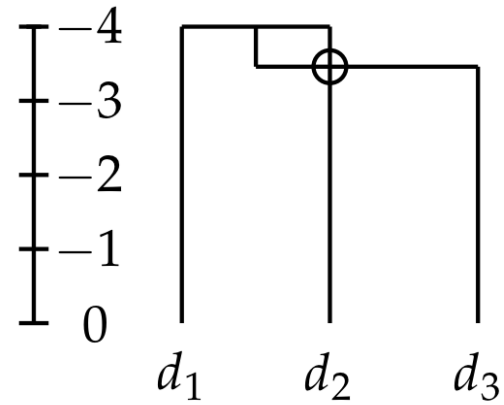
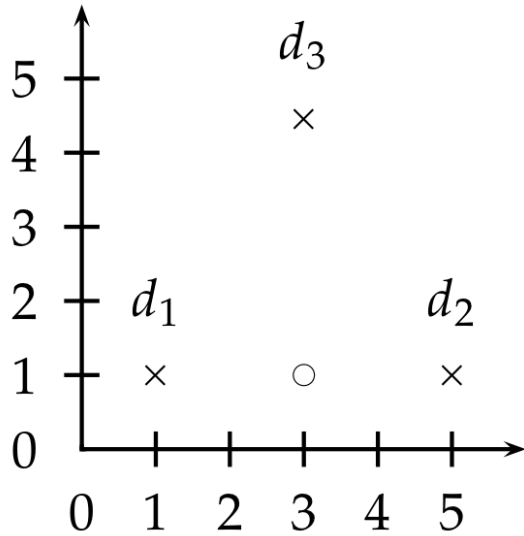
Centroid linkage is **simple**: easy to understand, and easy to implement. Maybe for these reasons, it has become the standard for hierarchical clustering in biology

Here $n = 60$, $X_i \in \mathbb{R}^2$, $d_{ij} = \|X_i - X_j\|_2$. Cutting the tree at some heights wouldn't make sense ... because the dendrogram has **inversions**! But we can, e.g., still look at output with 3 clusters



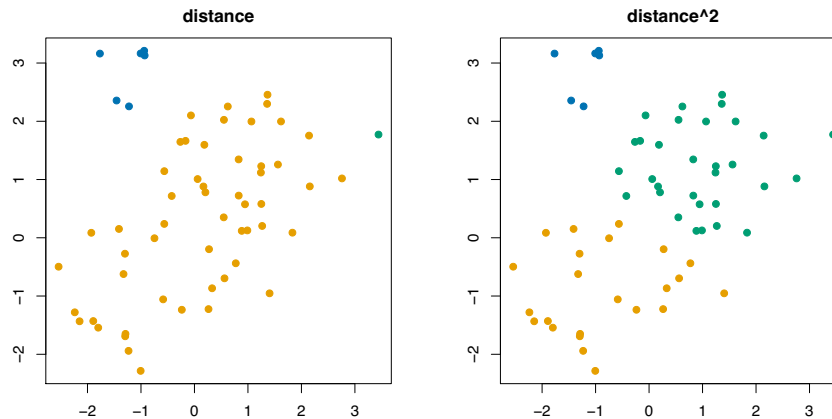
Cut interpretation: there isn't one, even with no inversions

Centroid Linkage Inversions



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 $d_{ij} = \|X_i - X_j\|_2$. E.g., changing to $d_{ij} = \|X_i - X_j\|_2^2$ would give a different clustering



Linkages summary

Linkage	No inversions?	Unchanged with monotone transformation?	Cut interpretation?	Notes
Single	✓	✓	✓	chaining
Complete	✓	✓	✓	crowding
Average	✓	×	×	
Centroid	×	×	×	simple

Note: this doesn't tell us what "best linkage" is.

Remember that choosing a linkage can be very **situation dependent**.

More Practical Considerations

- ▶ As usual lots of choices – what dissimilarity, what linkage, if K -means then how many clusters?
- ▶ Are clusters *statistically significant*? Imagine clustering noise, would still obtain a partition of data, so how do we know the clusters we found are “real”.
- ▶ In supervised learning, there was often a simple answer – try them all out and select using performance on a validation set. What do we do here?
- ▶ Generally difficult to answer, often try different things and see what groups are *persistent/stable*, i.e. are prominent across different methods and choices.