MATH 4750 / Computational Statistics

Instructor: Mehdi Maadooliat

K-Means and Hierarchical Clustering

Slides are mostly borrowed from
Ryan Tibshirani Data Mining
Class Notes

Department of Mathematical and Statistical Sciences

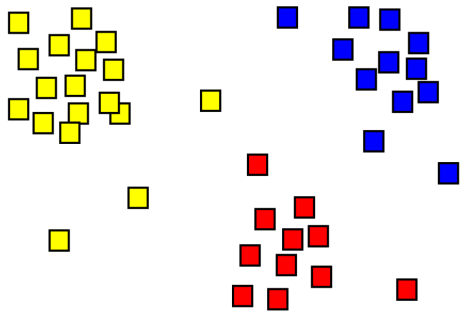
OUTLINE

- K-Means Clustering
  - Dissimilarity and within-cluster scatter
- Hierarchical clustering
  - Agglomerative vs Divisive
  - Single linkage
  - Complete linkage
  - Average linkage
  - Centroid linkage
  - MiniMax linkage
- Choosing the number of clusters
  - Within-cluster variation
  - Between-cluster variation
  - CH index

DON'T CONFUSE CLUSTERING AND CLASSIFICATION!

- In classfication, 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



DISSIMILARITY AND WITHIN-CLUSTER SCATTER

Given observations $X_1, \ldots X_n$, and dissimilarites $d(X_i, X_j)$. (E.g., think of $X_i \in \mathbb{R}^p$ and $d(X_i, X_j) = \|X_i - X_j\|_2^2$)

Let K be the number of clusters (fixed). A clustering of points $X_1, \ldots X_n$ 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. Also, let $d_{ij} = d(X_i, X_j)$

The within-cluster scatter is defined as

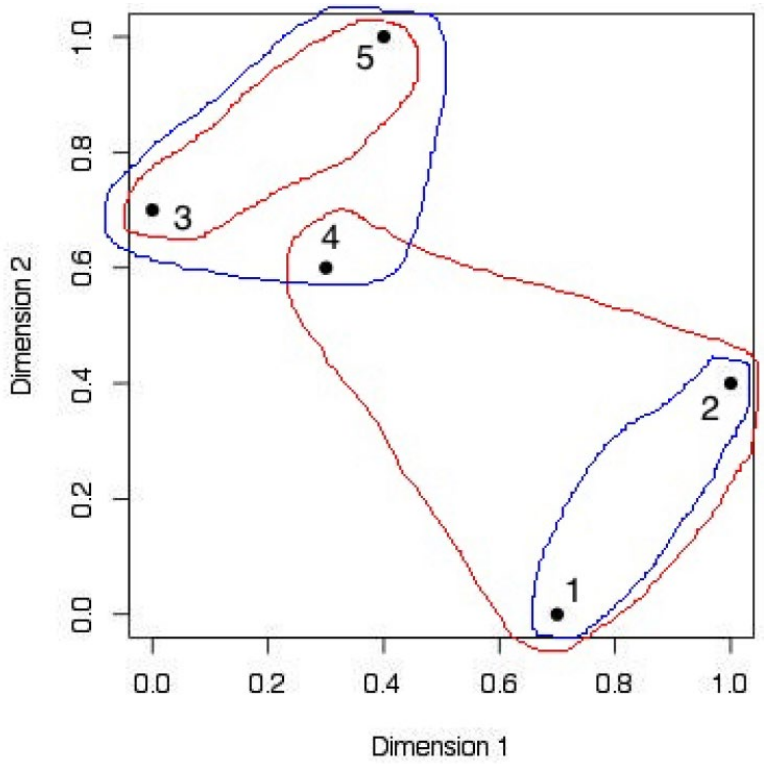
$$W = \frac{1}{2} \sum_{k=1}^{K} \frac{1}{n_k} \sum_{C(i)=k,\, C(j)=k} d_{ij}$$

Smaller W is better

Simple Example

Here $n = 5$ and $K = 2$,
$X_i \in \mathbb{R}^2$ and $d_{ij} = \|X_i - X_j\|_2^2$

	1	2	3	4	5
1	0	0.25	0.98	0.52	1.09
2	0.25	0	1.09	0.53	0.72
3	0.98	1.09	0	0.10	0.25
4	0.52	0.53	0.10	0	0.17
5	1.09	0.72	0.25	0.17	0



- Red clustering:
  $W_{\text{red}} = (0.25 + 0.53 + 0.52)/3 + 0.25/2 = 0.56$
- Blue clustering:
  $W_{\text{blue}} = 0.25/2 + (0.10 + 0.17 + 0.25)/3 = 0.30$

(Tip: `dist` function in R)

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

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

Note that $A(10, 4) = 34,105$, and $A(25, 4) \approx 5 \times 10^{13}$ ... huge

Most problems we look at are going to have way more than $n = 25$ observations, and potentially more than $K = 4$ clusters too (but $K = 4$ is not unrealistic)

So we'll have to settle for an approximation

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: within-cluster scatter can be rewritten as

$$\frac{1}{2}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{C(i)=k}\sum_{C(j)=k}\|X_i - X_j\|_2^2 = \sum_{k=1}^{K}\sum_{C(i)=k}\|X_i - \bar{X}_k\|_2^2$$

with $\bar{X}_k$ the average of points in group k, $\bar{X}_k = \frac{1}{n_k}\sum_{C(i)=k} X_i$.
The right-hand side above is called within-cluster variation

Hence, equivalently we seek a clustering C that minimizes the within-cluster variation (approximately so)

REWRITING THE MINIMIZATION

Remember: we want to choose C to minimize

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

Another fact : for any $Z_1, \ldots Z_m \in \mathbb{R}^p$, the quantity $\sum_{i=1}^{m} \|Z_i - c\|_2^2$ is minimized by taking $c = \bar{Z} = \frac{1}{m} \sum_{i=1}^{m} Z_i$, the average of the Z_i's

So our problem is the same as minimizing the enlarged criterion

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

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

K-MEANS ALGORITHM

The K-means clustering algorithm approximately minimizes the enlarged criterion by alternately minimizing over C and $c_1, \ldots c_K$

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 group k

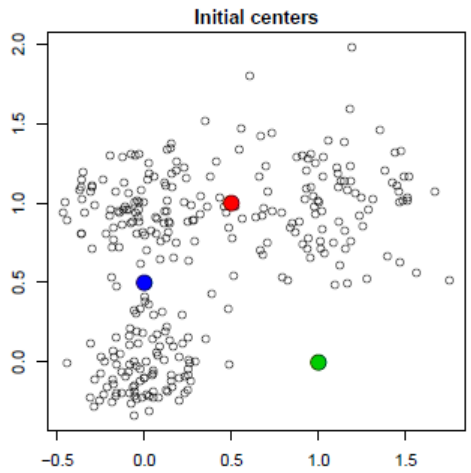
Stop when within-cluster variation doesn't change

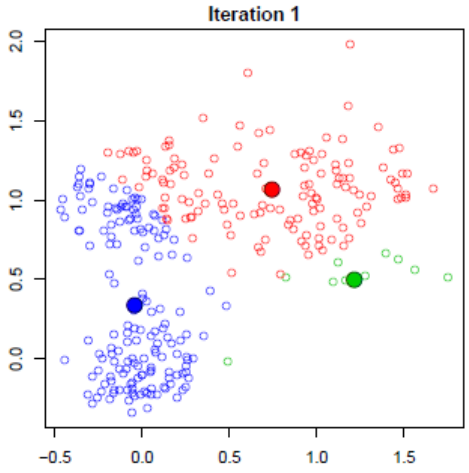
In words:

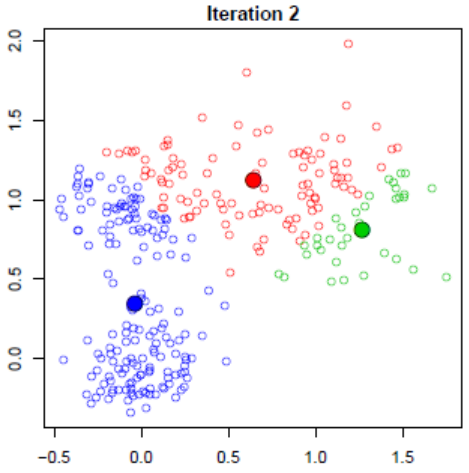
1. Cluster (label) each point based the closest center
2. Replace each center by the average of points in its cluster

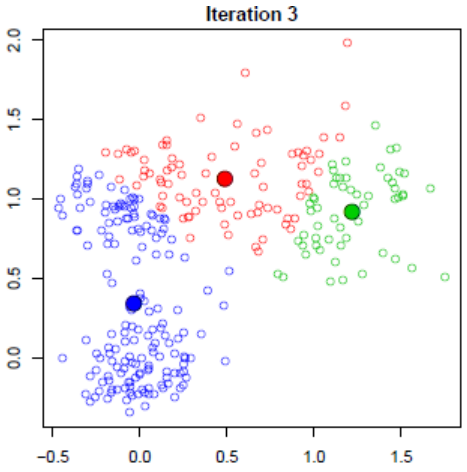
K-MEANS EXAMPLE

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

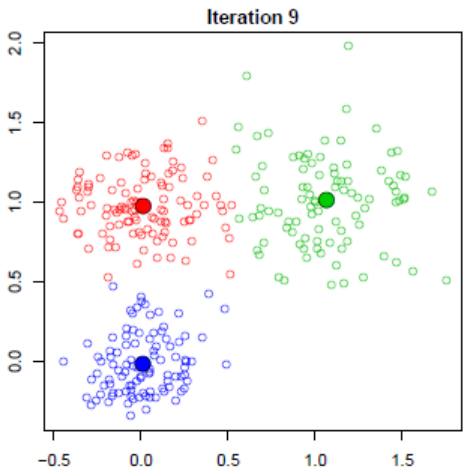








. . .

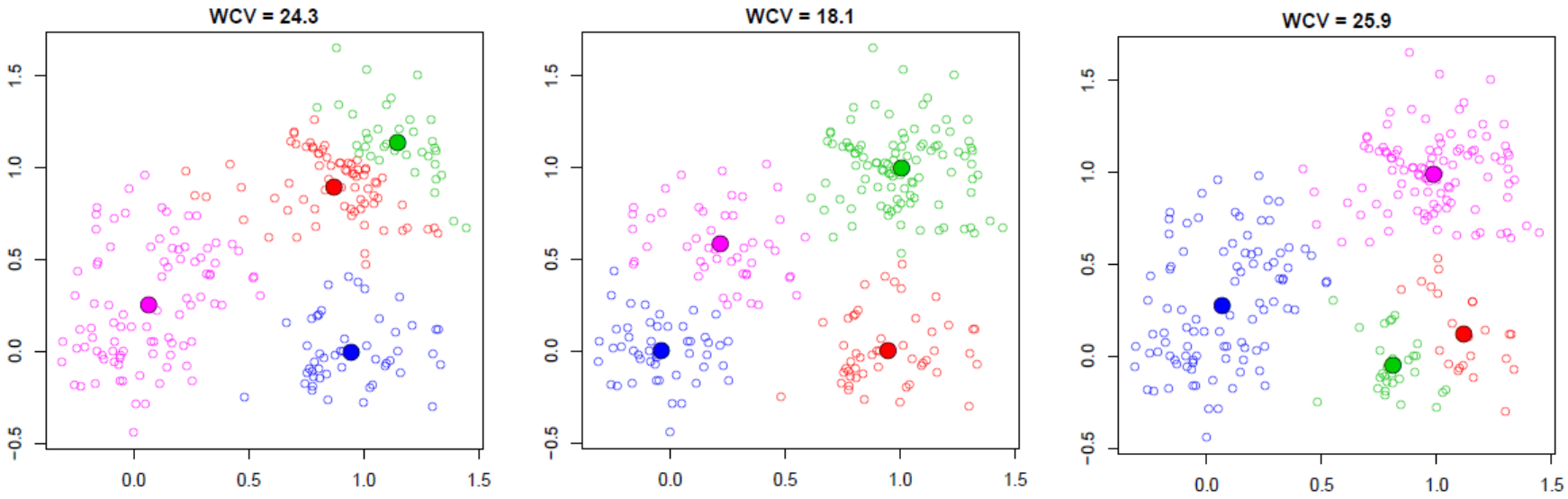


PROPERTIES OF K-MEANS

- Within-cluster variation decreases with each iteration of the algorithm. i.e., if W_t is the within-cluster variation at iteration t, then $W_{t+1} \leq W_t$
- The algorithm always converges, no matter the initial cluster centers. In fact, it takes $\leq K^n$ iterations (why?)
- The final clustering depends on the initial cluster centers. Sometimes, different initial centers lead to very different final outputs. So we typically run K-means multiple times (e.g., 10 times), randomly initializing cluster centers for each run, then choose among from collection of centers based on which one gives the smallest within-cluster variation
- The algorithm is not guaranteed to deliver the clustering that globally minimizes within-cluster variation (recall: this would require looking through all possible assignments!)

K-MEANS EXAMPLE, MULTIPLE RUNS

Here $X_i \in \mathbb{R}^2$, $n = 250$, and $K = 4$, the points are not as well-separated



These are results of result of running the K-means algorithm with different initial centers (chosen randomly over the range of the X_i's). We choose the second collection of centers because it yields the smallest within-cluster variation

From K-means to Hierarchical Clustering

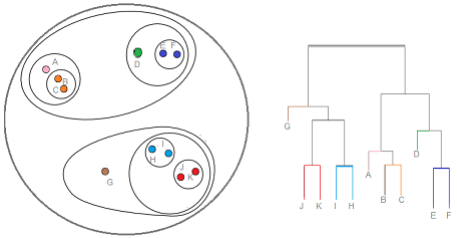
Recall two properties of K-means (K-medoids) clustering:

1. It fits exactly K clusters (as specified)
2. 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

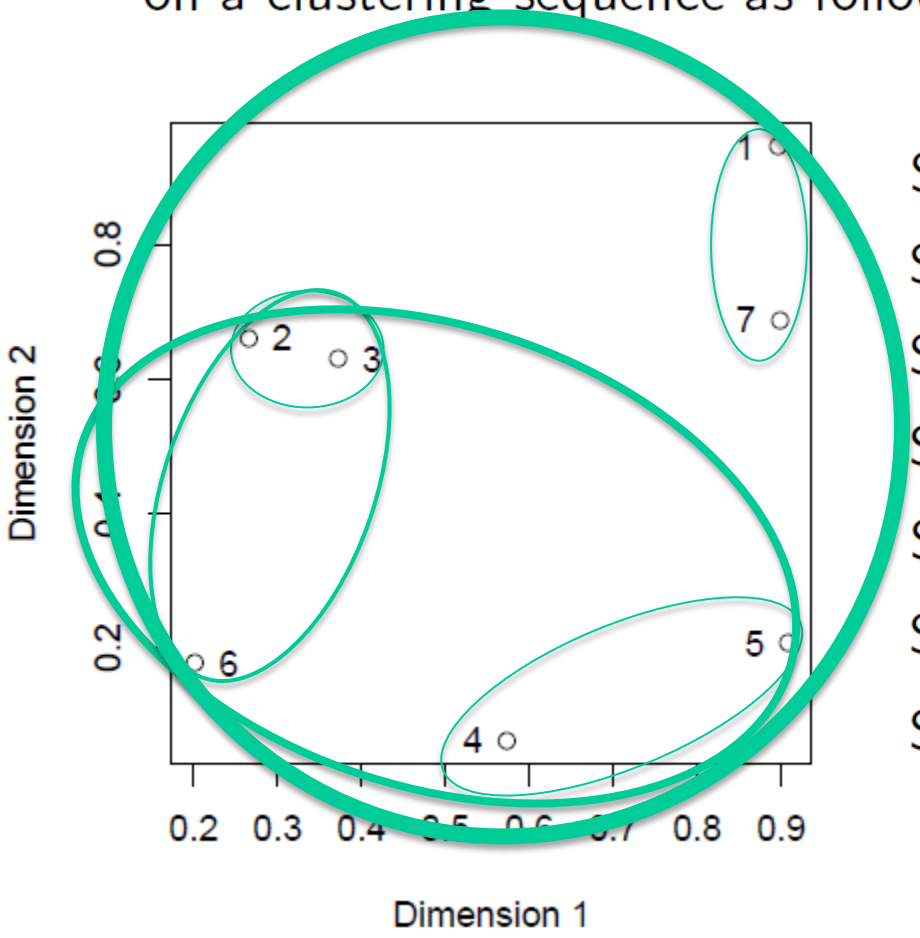
Divisive (i.e., top-down):

- 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

Agglomerative strategies are simpler, we'll focus on them. Divisive methods are still important, but we won't be able to cover them in lecture

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

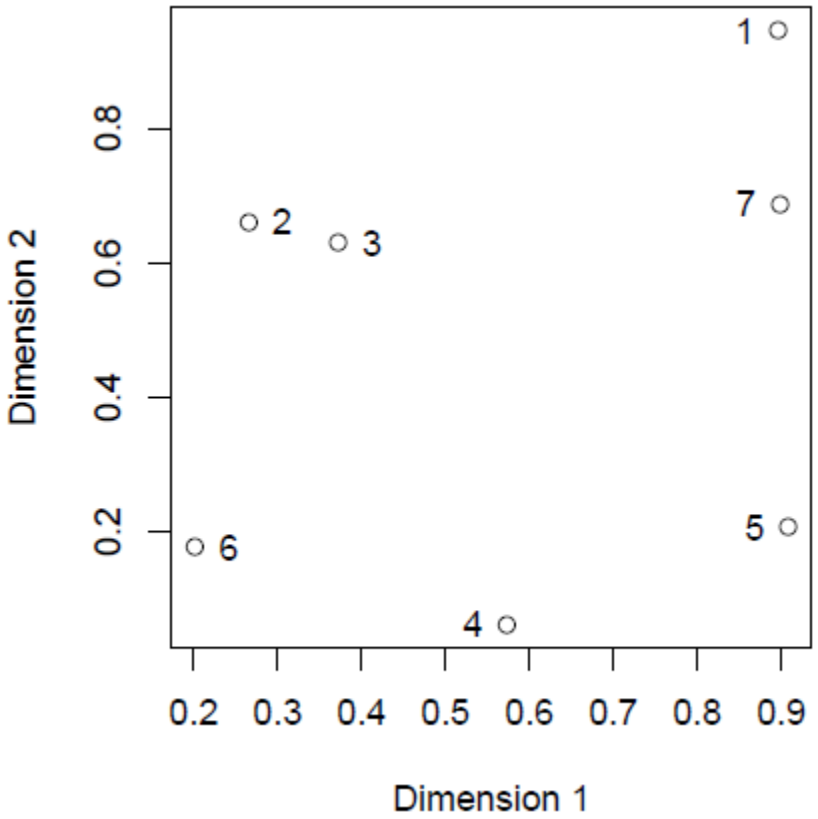
Step 5: $\{1, 7\}, \{2, 3, 6\}, \{4, 5\}$;

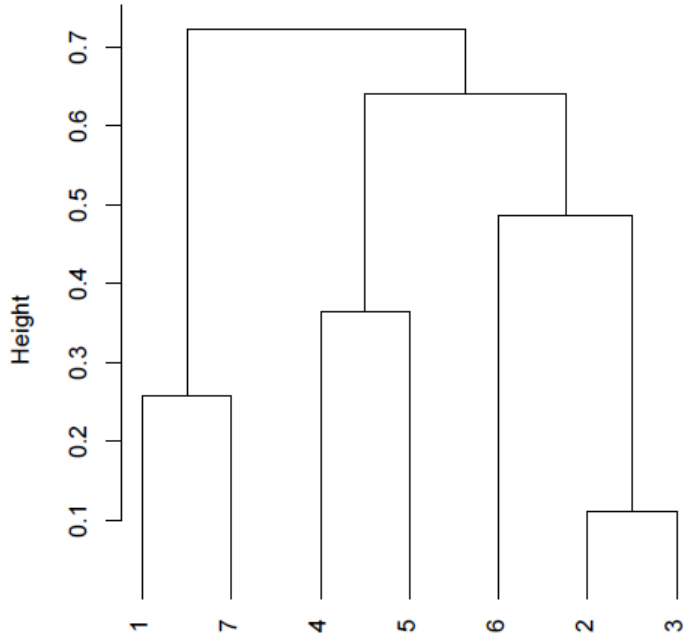
Step 6: $\{1, 7\}, \{2, 3, 4, 5, 6\}$;

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

Simple Example Cont…

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





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

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

Remember: 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 points $X_1, \ldots X_n$, 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 distance, not squared distance)

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:

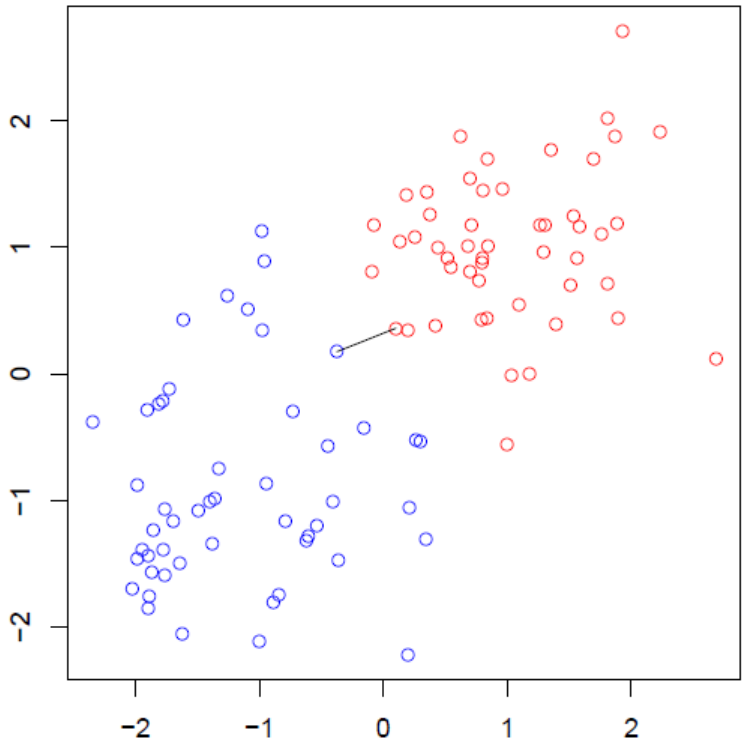
- Start with all points in their own group
- Until there is only one cluster, repeatedly: merge the two groups G, H such that $d(G, H)$ is smallest

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:

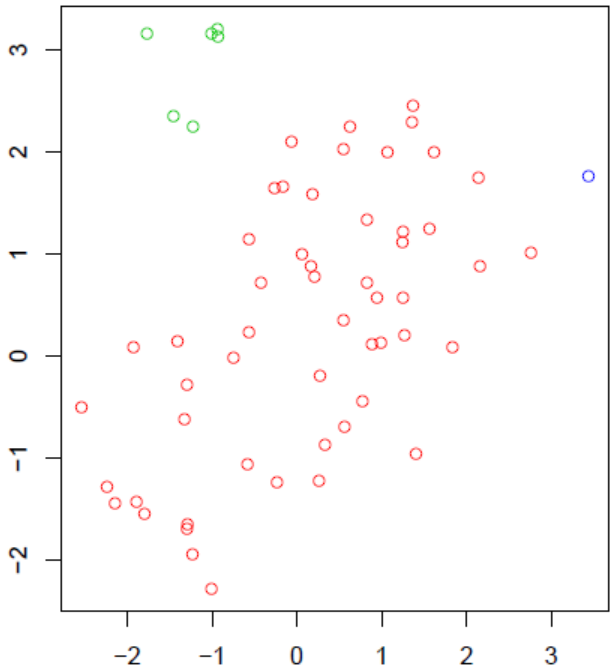
$$d_{\text{single}}(G, H) = \min_{i \in G,\, j \in H} d_{ij}$$

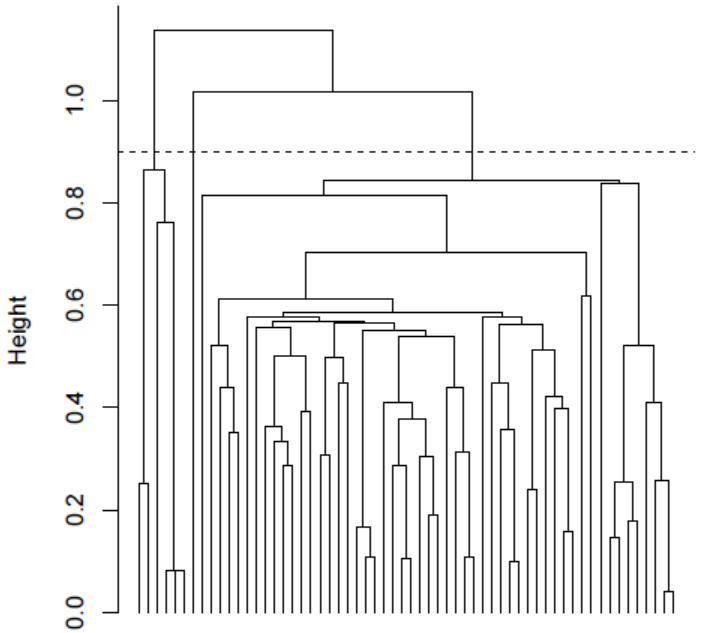
Example (dissimilarities d_{ij} are distances, groups are marked by colors): single linkage score $d_{\text{single}}(G, H)$ is the distance of the closest pair



Single Linkage Example

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





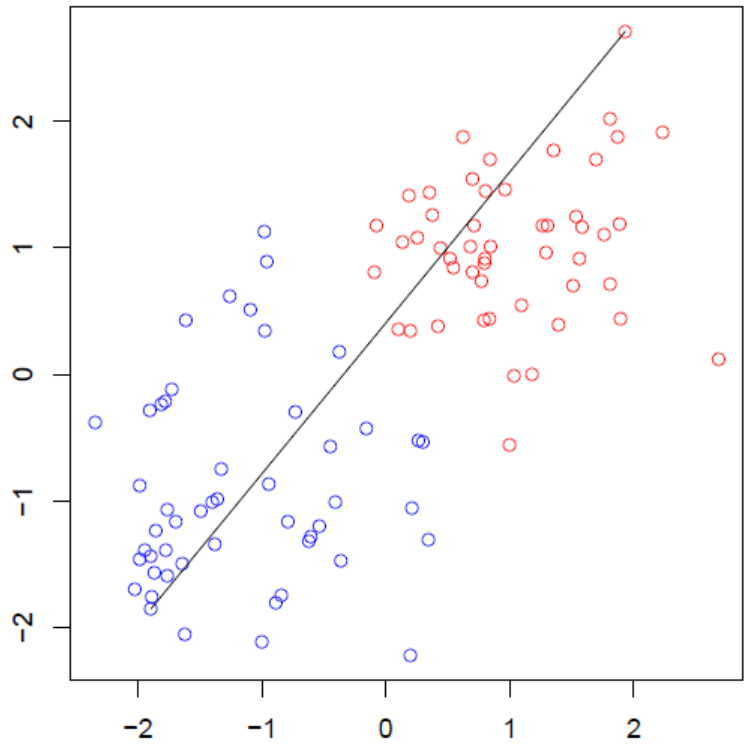
Cut interpretation: for each point X_i, there is another point X_j in its cluster with $d_{ij} \leq 0.9$

Complete Linkage

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

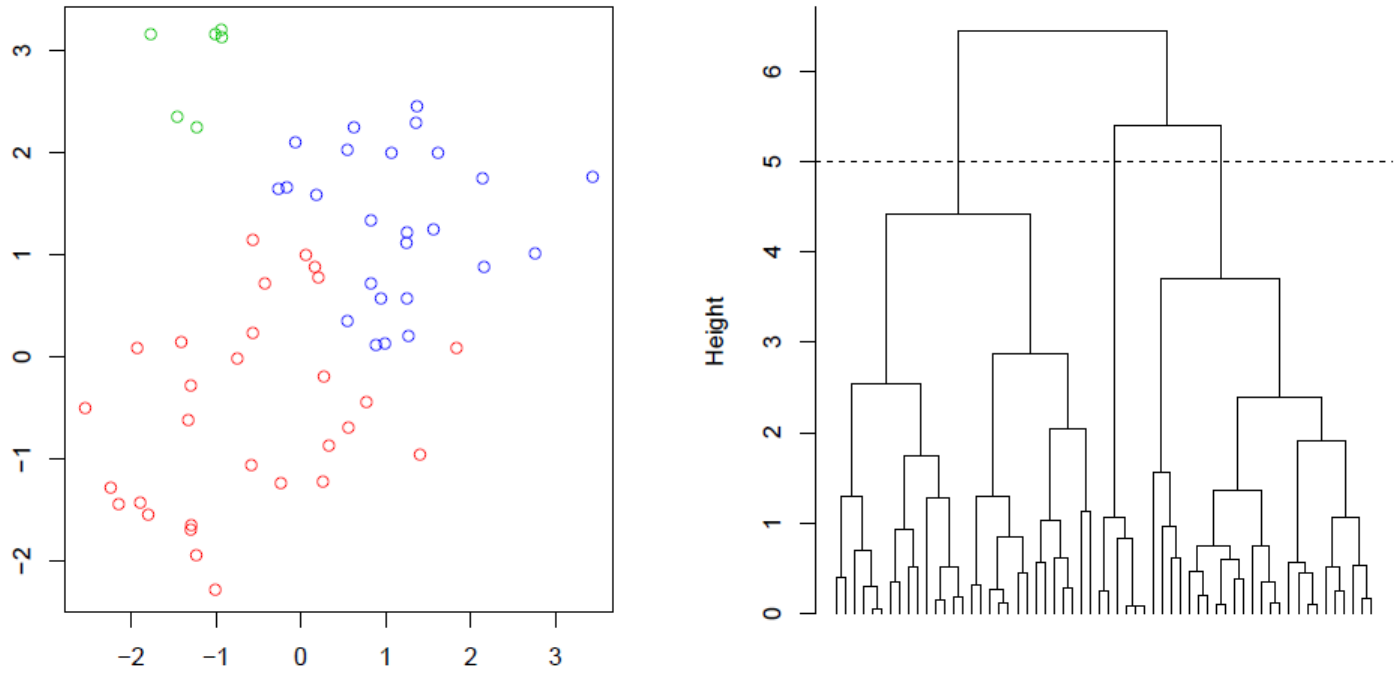
$$d_{\text{complete}}(G, H) = \max_{i \in G,\, j \in H} d_{ij}$$

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 $d_{ij} \leq 5$

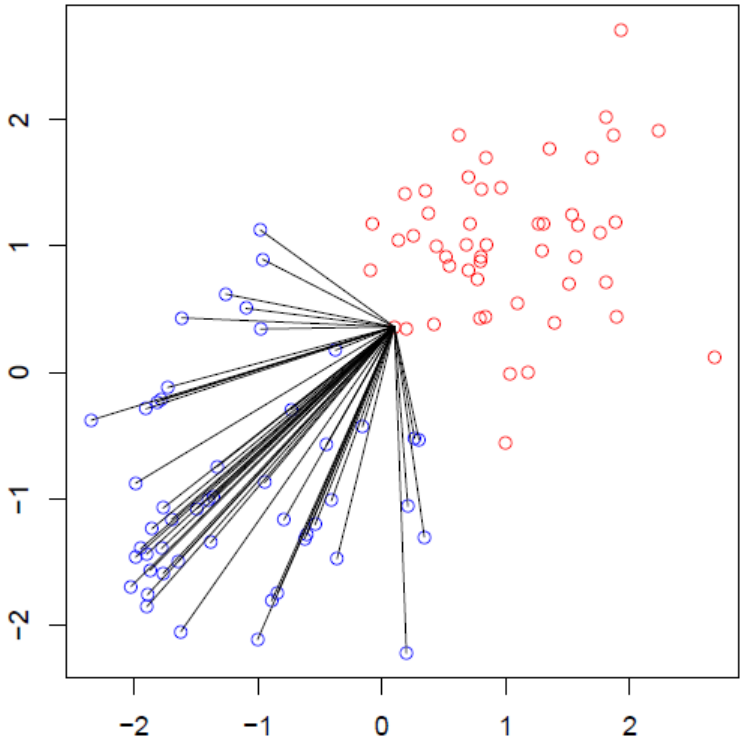
Average Linkage

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

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

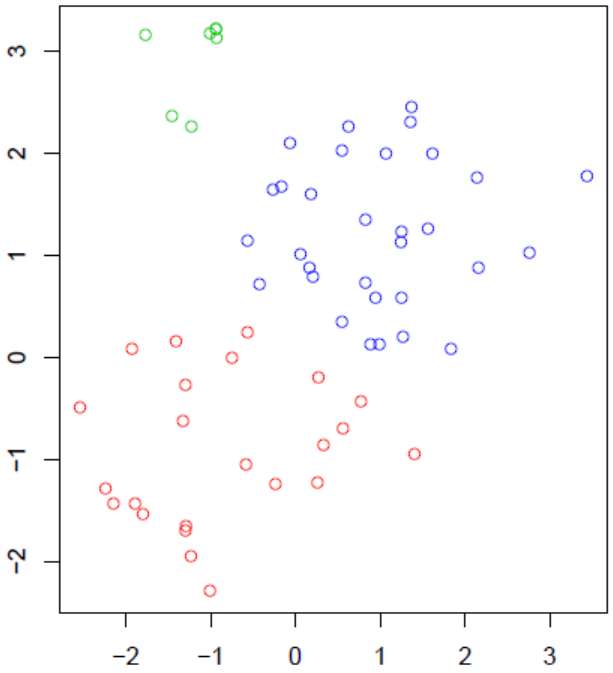
Example (dissimilarities d_{ij} are distances, groups are marked by colors): average linkage score $d_{\text{average}}(G, H)$ is the average distance across all pairs

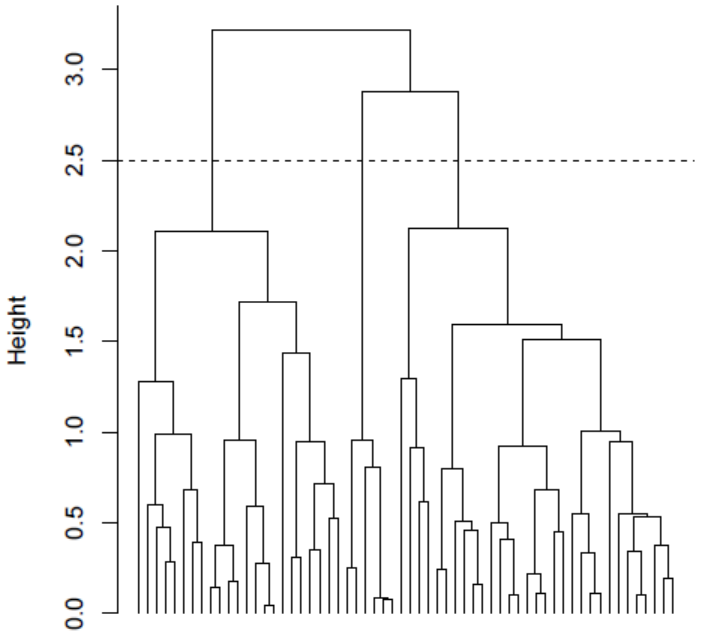
(Plot here only shows distances between the blue points and one red point)



Average Linkage Example

Same data as before. Cutting the tree at $h = 2.5$ gives clustering assignments marked by the colors





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

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

Shortcomings of Average Linkage

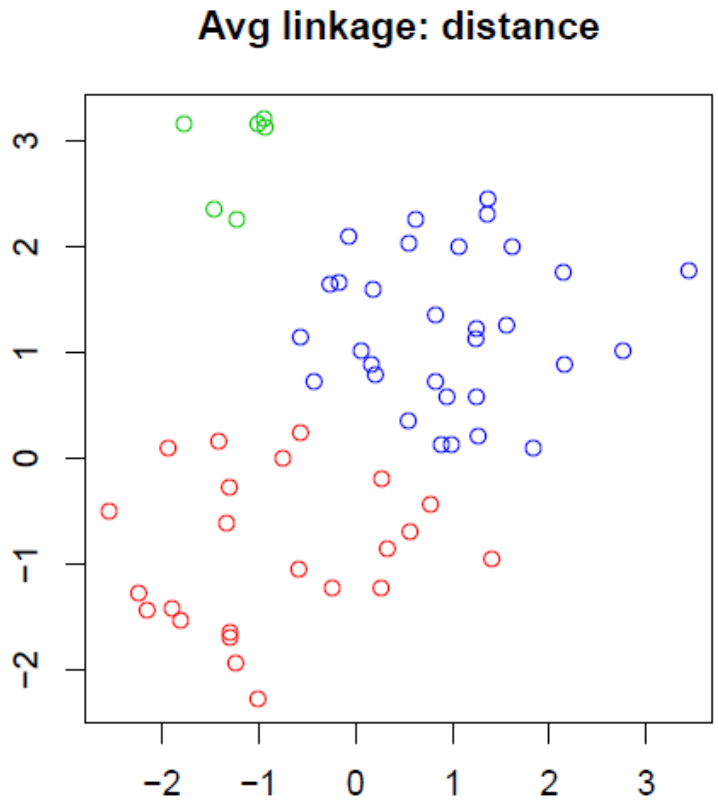
Average linkage isn't perfect, it has its own problems:

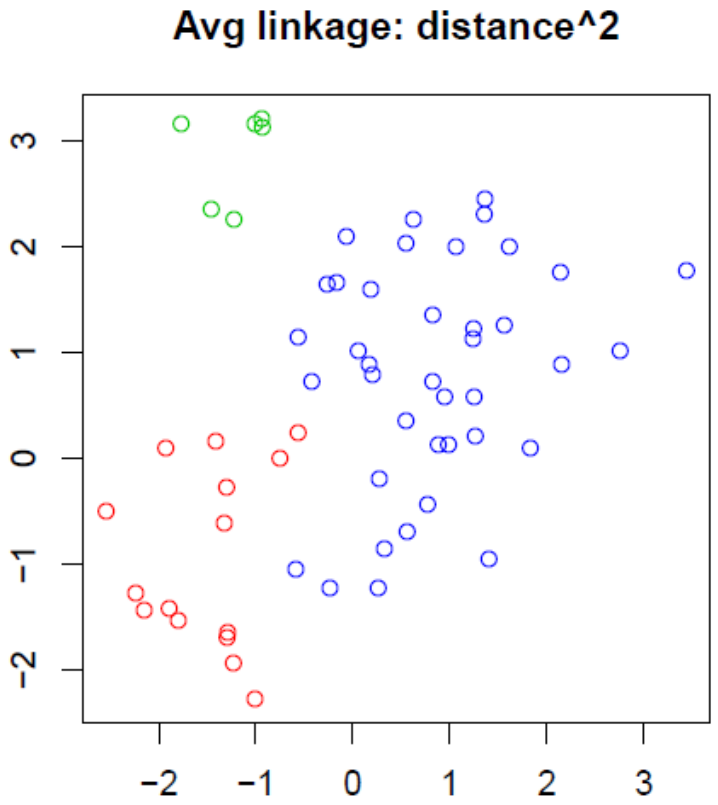
- It is not clear what properties the resulting clusters have when we cut an average linkage tree at given height h. Single and complete linkage trees each had simple interpretations
- Results of average linkage clustering can change with a monotone increasing transformation of dissimilarities d_{ij}. i.e., if h is such that $h(x) \leq h(y)$ whenever $x \leq y$, and we used dissimilarites $h(d_{ij})$ 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





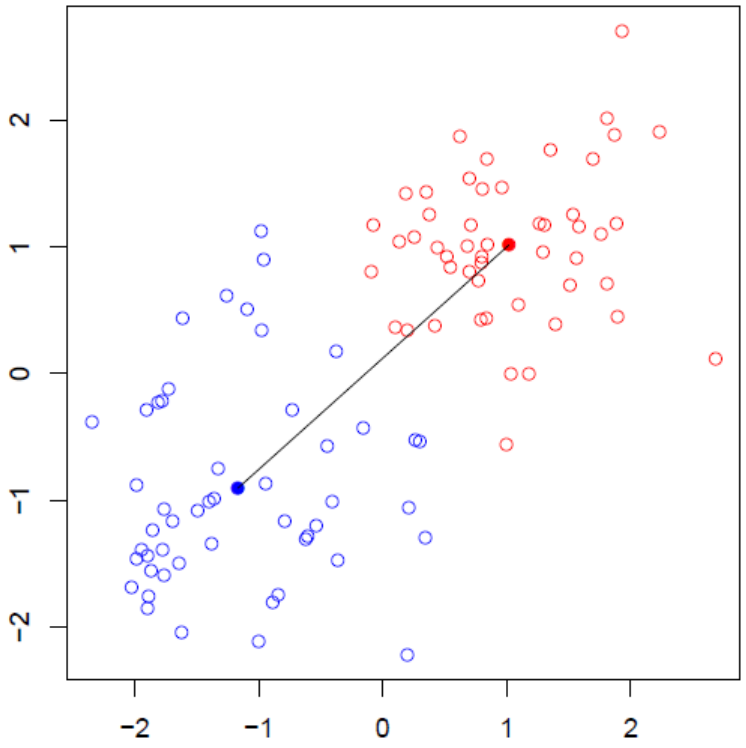
Centroid Linkage

Centroid linkage[1] 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:

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

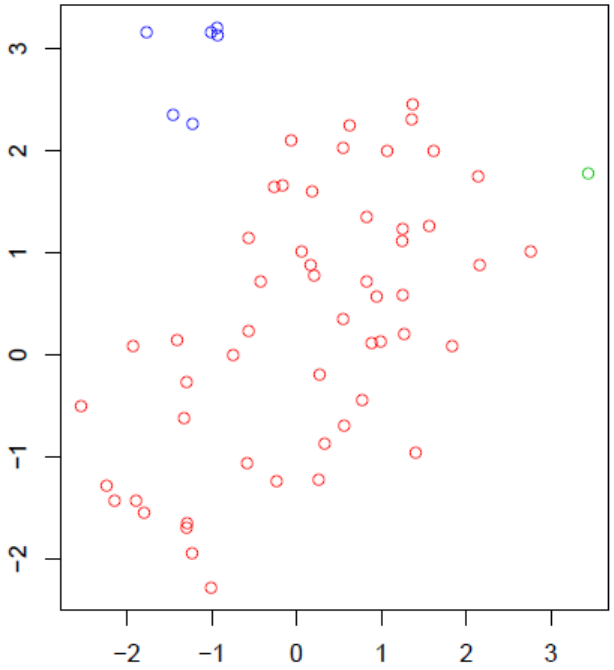
Example (dissimilarities d_{ij} are distances, groups are marked by colors): centroid linkage score $d_{\text{centroid}}(G, H)$ is the distance between the group centroids (i.e., group averages)

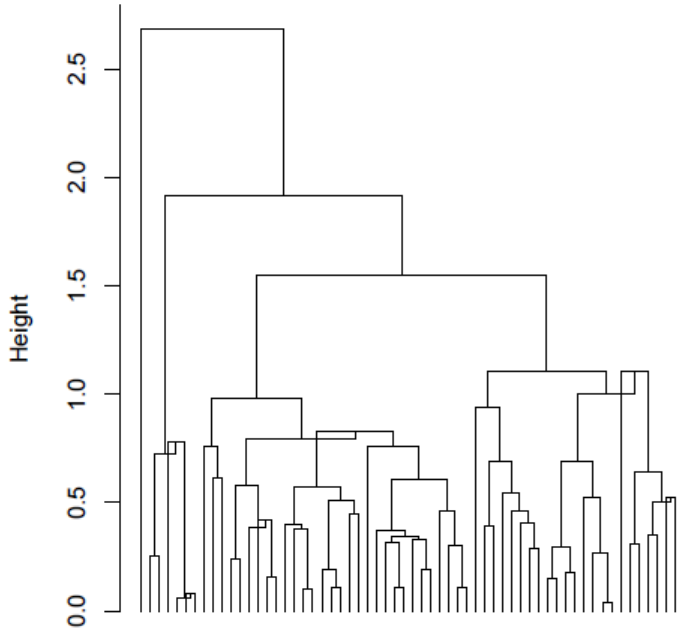


[1] Eisen et al. (1998), "Cluster Analysis and Display of Genome-Wide Expression Patterns"

Centroid Linkage Example

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 ouptut with 3 clusters



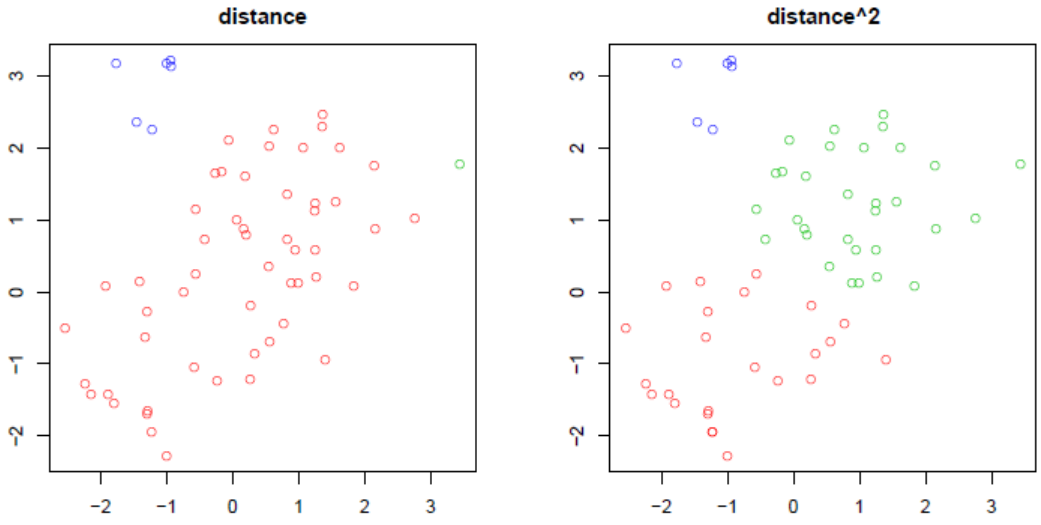


Cut interpretation: there isn't one, even with no inversions

Shortcomings of Centroid Linkage

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

- 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 $\tilde{d}_{ij} = \|X_i - X_j\|_2^2$ would give a different clustering

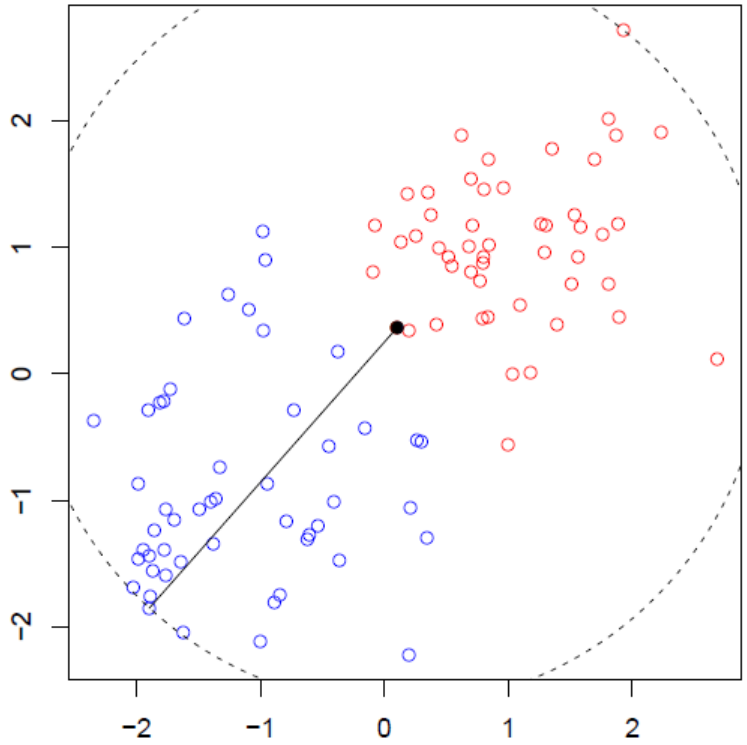


MiniMax Linkage

Minimax linkage[2] is a newcomer. First define radius of a group of points G around X_i as $r(X_i, G) = \max_{j \in G} d_{ij}$. Then:

$$d_{\text{minimax}}(G, H) = \min_{i \in G \cup H} r(X_i, G \cup H)$$

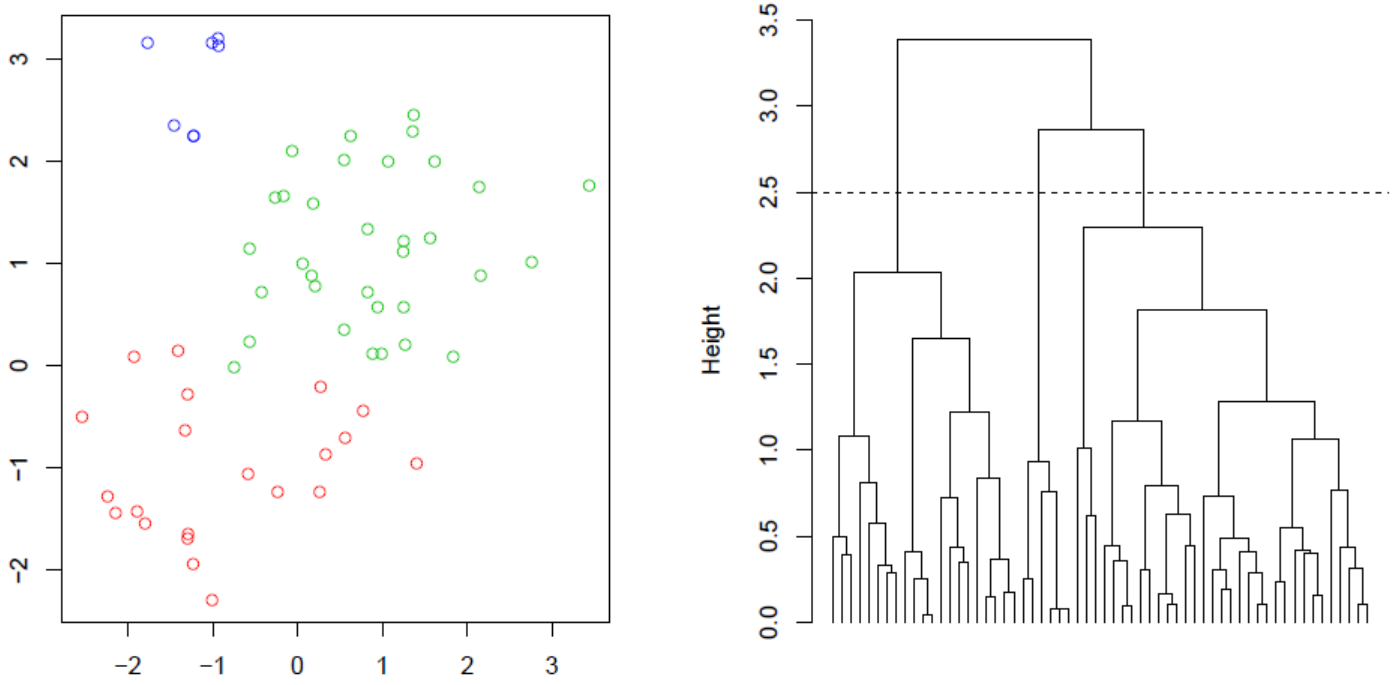
Example (dissimilarities d_{ij} are distances, groups marked by colors): minimax linkage score $d_{\text{minimax}}(G, H)$ is the smallest radius encompassing all points in G and H. The center X_c is the black point



[2] Bien et al. (2011), "Hierarchical Clustering with Prototypes via Minimax Linkage"

MiniMax Linkage Example

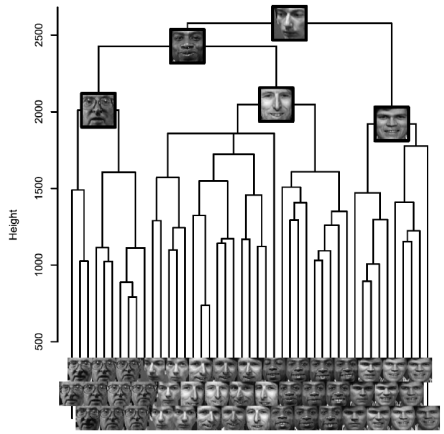
Same data as before. Cutting the tree at $h = 2.5$ gives clustering assignments marked by the colors



Cut interpretation: each point X_i belongs to a cluster whose center X_c satisfies $d_{ic} \leq 2.5$

Properties of MiniMax Linkage

- Cutting a minimax tree at a height h a nice interpretation: each point is $\leq h$ in dissimilarity to the center of its cluster. (This is related to a famous set cover problem)
- Produces dendrograms with no inversions
- Unchanged by monotone transformation of dissimilarities d_{ij}
- Produces clusters whose centers are chosen among the data points themselves. Remember that, depending on the application, this can be a very important property. (Hence minimax clustering is the analogy to K-medoids in the world of hierarchical clustering)



(From Bien et al. (2011))

Linkage Summary

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

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

What's missing here: a detailed empirical comparison of how they perform. On top of this, remember that choosing a linkage can be very situation dependent

How many clusters?

Sometimes, using K-means, K-medoids, or hierarchical clustering, we might have no problem specifying the number of clusters K ahead of time, e.g.,

- Segmenting a client database into K clusters for K salesman
- Compressing an image using vector quantization, where K controls the compression rate

Other times, K is implicitly defined by cutting a hierarchical clustering tree at a given height

But in most exploratory applications, the number of clusters K is unknown. So we are left asking the question: what is the "right" value of K?

This is a Hard Problem

Determining the number of clusters is a hard problem!

Why is it hard?

- Determining the number of clusters is a hard task for humans to perform (unless the data are low-dimensional). Not only that, it's just as hard to explain what it is we're looking for.

Why is it important?

- E.g., 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 (larger) goals of data mining/statistical learning is automatic inference; choosing K is certainly part of this

Reminder: within-cluster variation

We're going to focus on K-means, but most ideas will carry over to other settings

Recall: given the number of clusters K, the K-means algorithm approximately minimizes the within-cluster variation:

$$W = \sum_{k=1}^{K} \sum_{C(i)=k} \|X_i - \bar{X}_k\|_2^2$$

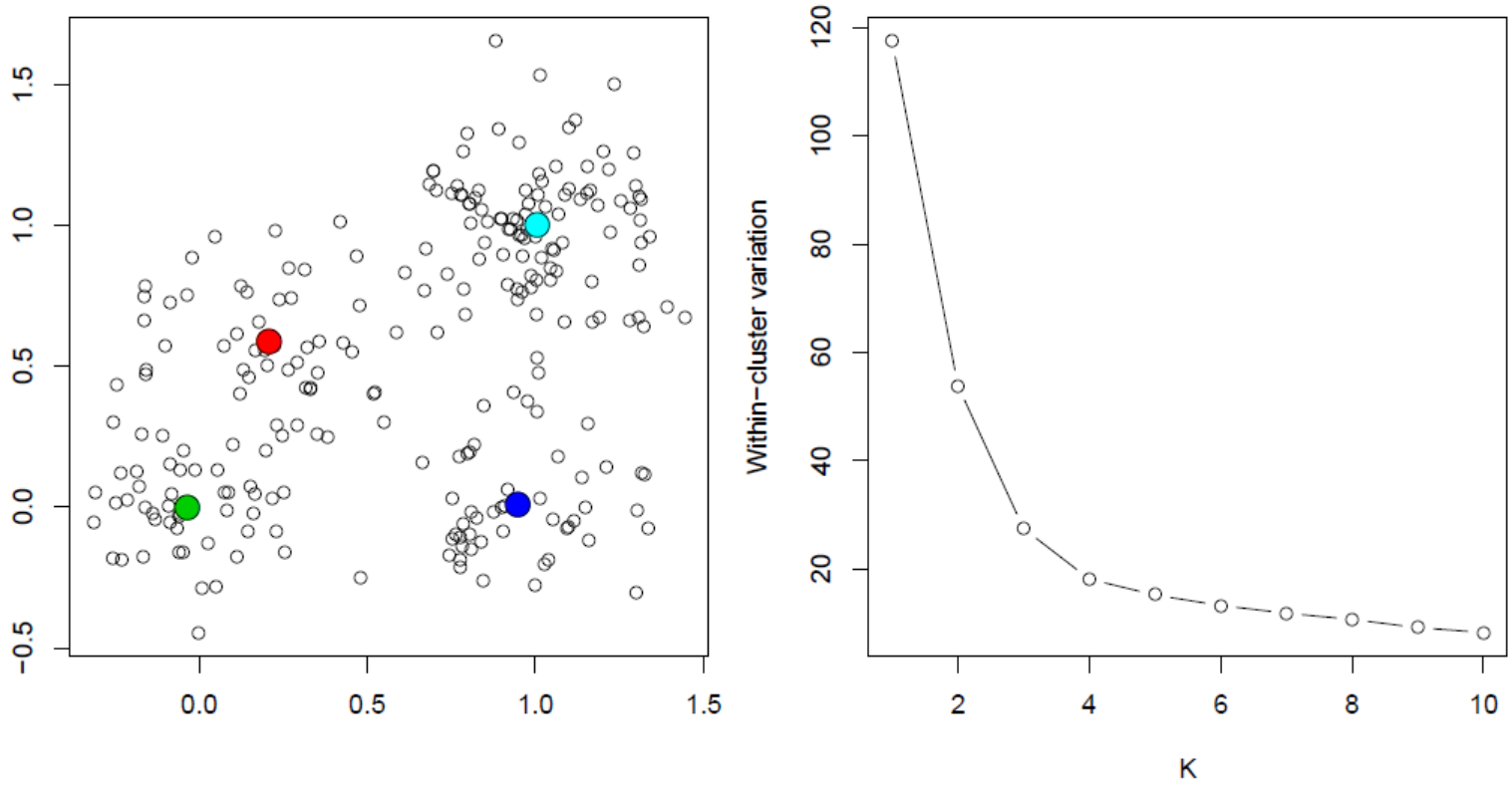
over clustering assignments C, where $\bar{X}_k$ is the average of points in group k, $\bar{X}_k = \frac{1}{n_k} \sum_{C(i)=k} X_i$

Clearly a lower value of W is better. So why not just run K-means for a bunch of different values of K, and choose the value of K that gives the smallest $W(K)$?

THAT'S NOT GOING TO WORK

Problem: within-cluster variation just keeps decreasing

Example: $n = 250$, $p = 2$, $K = 1, \ldots 10$



BETWEEN-CLUSTER VARIATION

Within-cluster variation measures how tightly grouped the clusters are. As we increase the number of clusters K, this just keeps going down. What are we missing?

Between-cluster variation measures how spread apart the groups are from each other:

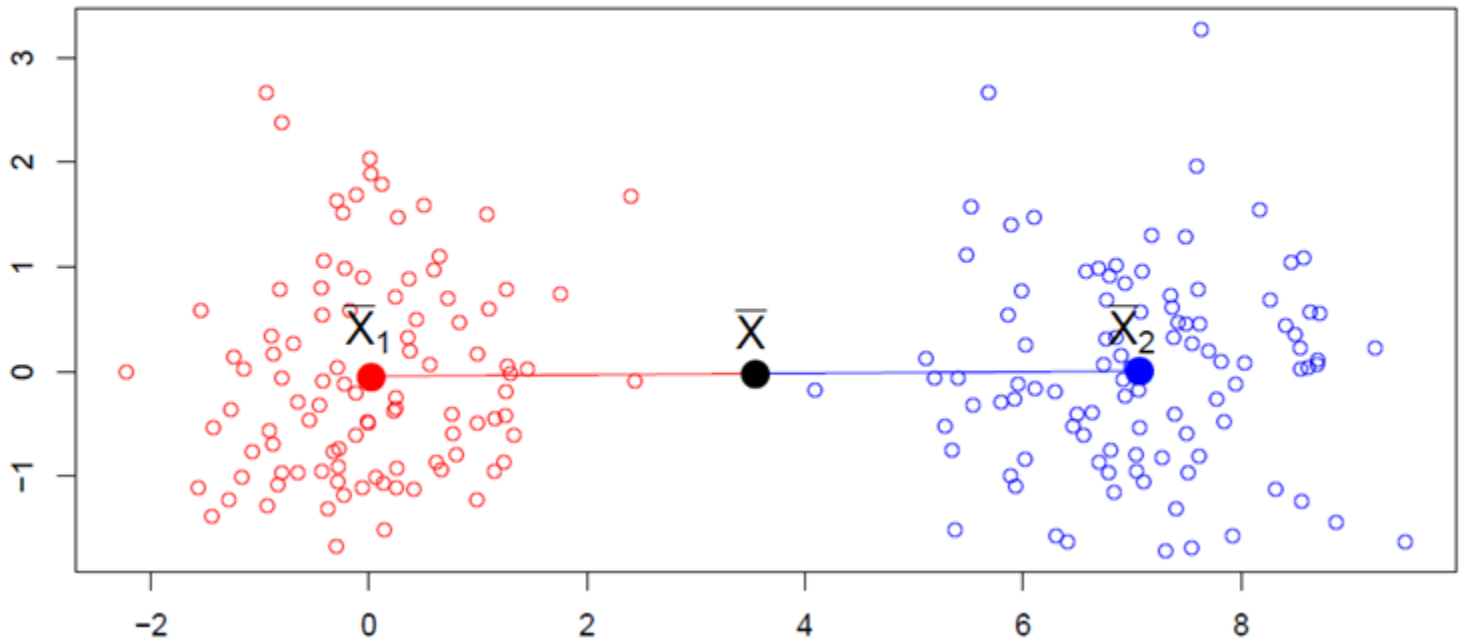
$$B = \sum_{k=1}^{K} n_K \|\bar{X}_k - \bar{X}\|_2^2$$

where as before $\bar{X}_k$ is the average of points in group k, and $\bar{X}$ is the overall average, i.e.

$$\bar{X}_k = \frac{1}{n_k} \sum_{C(i)=k} X_i \quad \text{and} \quad \bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$$

Example: between and Within Cluster variation

Example: $n = 100,\ p = 2,\ K = 2$



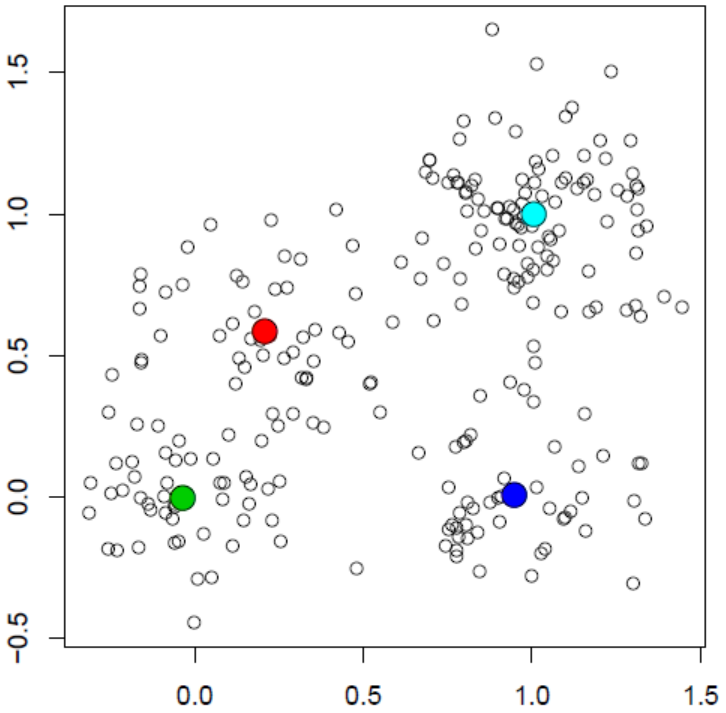
$$W = \sum_{C(i)=1} \|X_i - \bar{X}_1\|_2^2 + \sum_{C(i)=2} \|X_i - \bar{X}_2\|_2^2$$

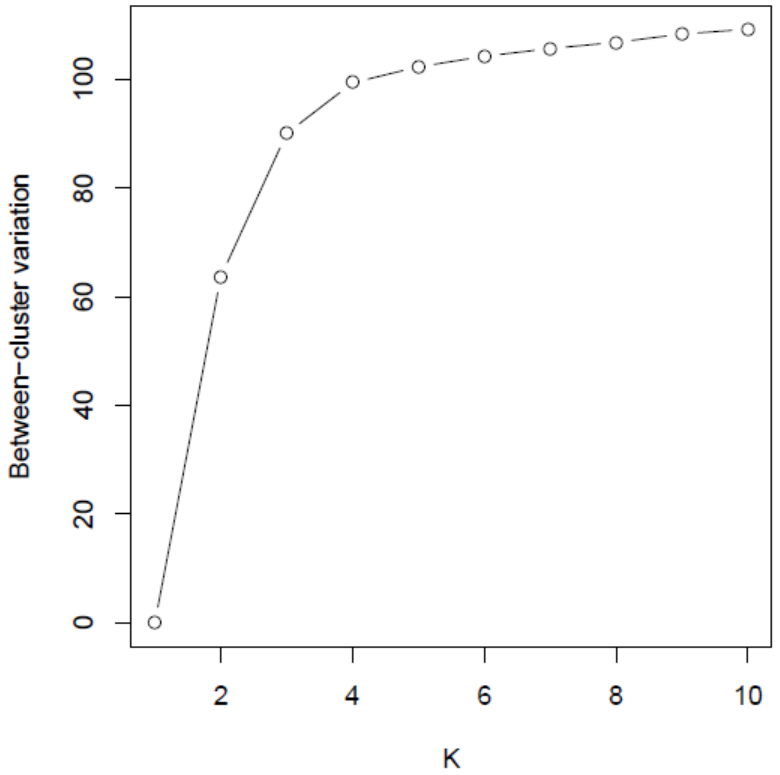
$$B = n_1\|\bar{X}_1 - \bar{X}\|_2^2 + n_2\|\bar{X}_2 - \bar{X}\|_2^2$$

STILL NOT GOING TO WORK

Bigger B is better, can we use it to choose K? Problem: between-cluster variation just keeps increasing

Running example: $n = 250$, $p = 2$, $K = 1, \ldots 10$





CH INDEX

Ideally we'd like our clustering assignments C to simultaneously have a small W and a large B

This is the idea behind the CH index.[3] For clustering assignments coming from K clusters, we record CH score:

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

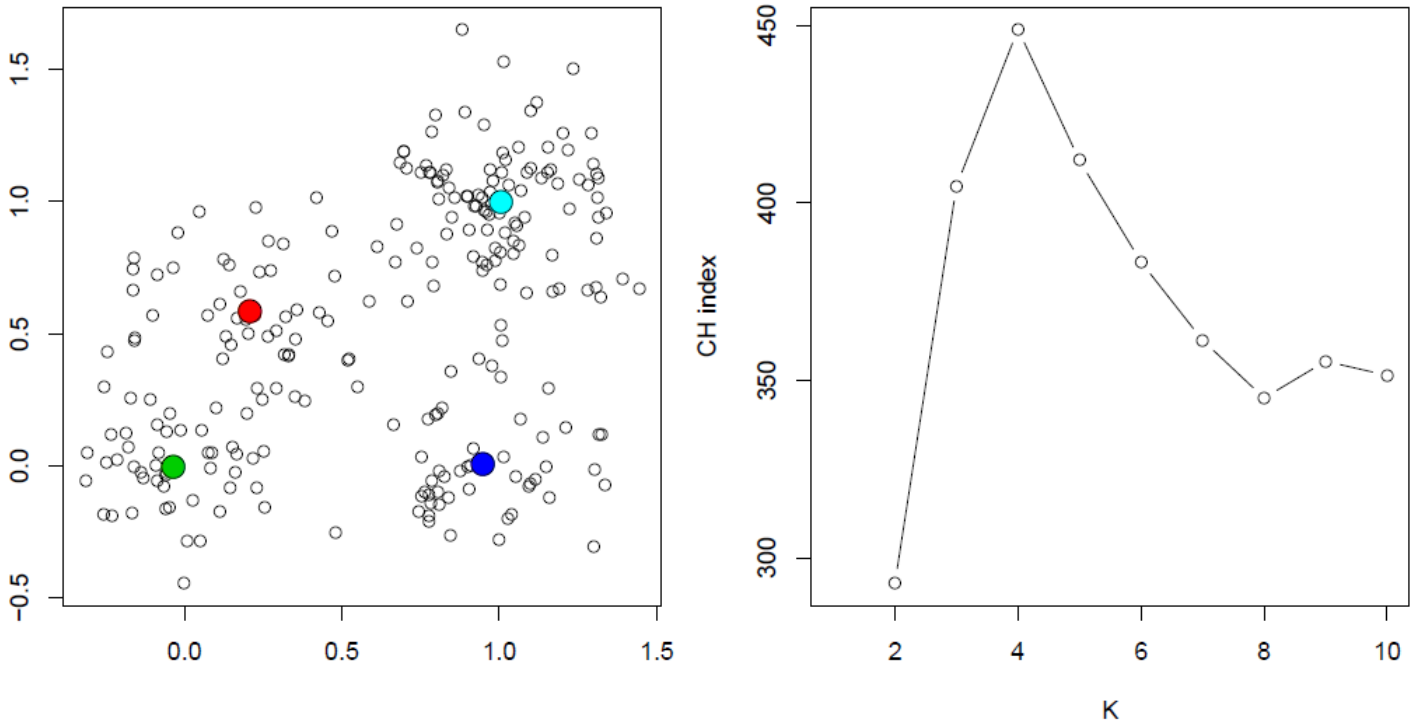
To choose K, just pick some maximum number of clusters to be considered $K_{\max}$ (e.g., $K = 20$), and choose the value of K with the largest score $\mathsf{CH}(K)$, i.e.,

$$\hat{K} = \underset{K \in \{2,\ldots K_{\max}\}}{\operatorname{argmax}} \mathsf{CH}(K)$$

[3] Calinski and Harabasz (1974), "A dendrite method for cluster analysis"

EXAMPLE: CH INDEX

Running example: $n = 250$, $p = 2$, $K = 2, \ldots 10$.



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)!

QUESTIONS?

- ANY QUESTION?