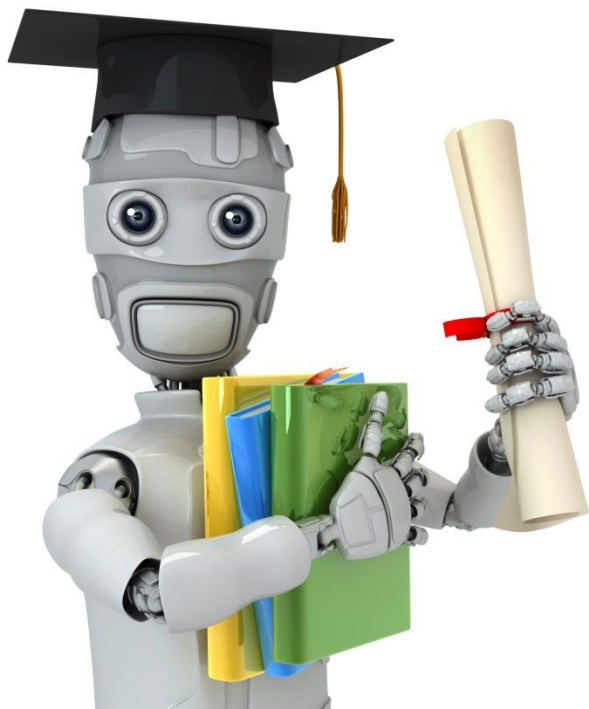


Machine Learning

# Clustering

---

Fouad Hadj Selem



Machine Learning

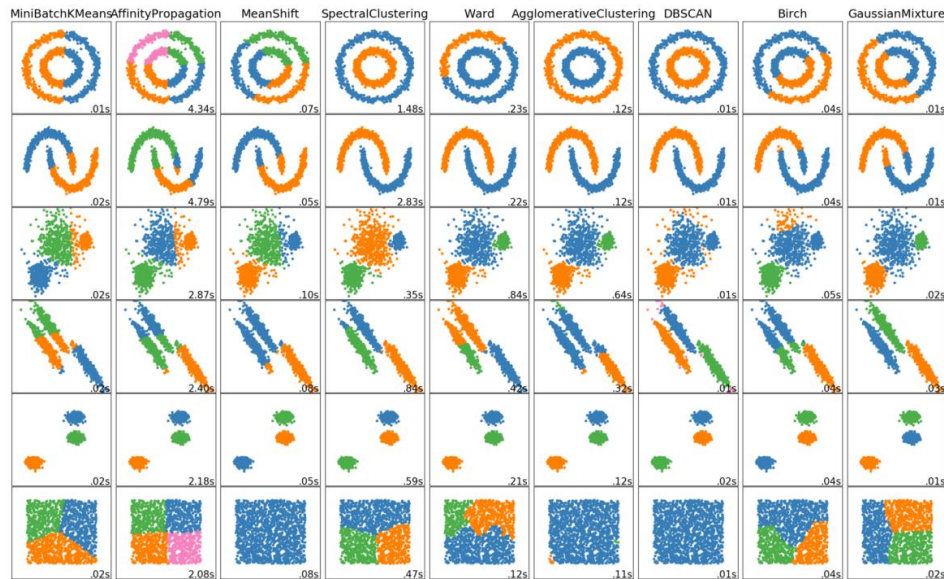
# Introduction to Data Clustering

---

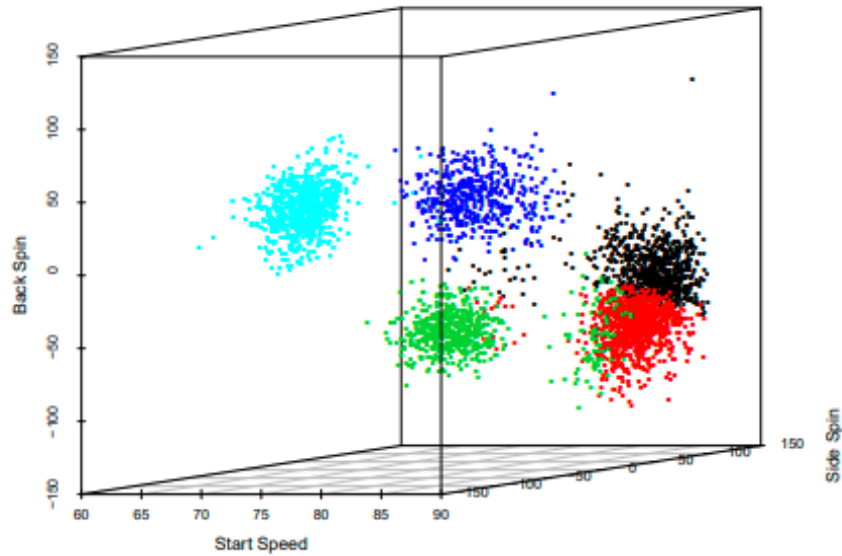
# What is Data Clustering?

---

- ▶ Data clustering is to partition data into groups, where the data in the same group are similar to one another and the data from different groups are dissimilar [Han and Kamber, 2000].
- ▶ To segment data into clusters so that the *intra-cluster similarity* is maximized and that the *inter-cluster similarity* is minimized.
- ▶ The groups obtained are a partition of data, which can be used for customer segmentation, document categorization, etc.



# Example : Clustering of Baseball Pitches



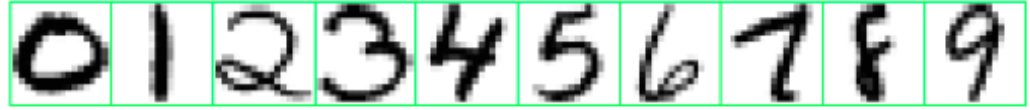
Inferred meaning of clusters: black – fastball, red – sinker, green – changeup, blue – slider, light blue – curveball

(Example from Mike Pane, former data mining student)

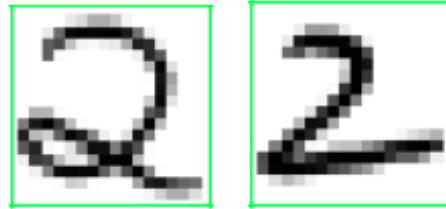
# Difference Between Clustering and Classification

---

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



# Main metric :Dissimilarity and Within- Cluster Scatter

---

Given observations  $X_1, \dots, X_n$ , and **dissimilarities**  $d(X_i, X_j)$ . (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  $X_1, \dots, X_n$  is a function  $C$  that assigns each observation  $X_i$  to a group  $k \in \{1, \dots, K\}$

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

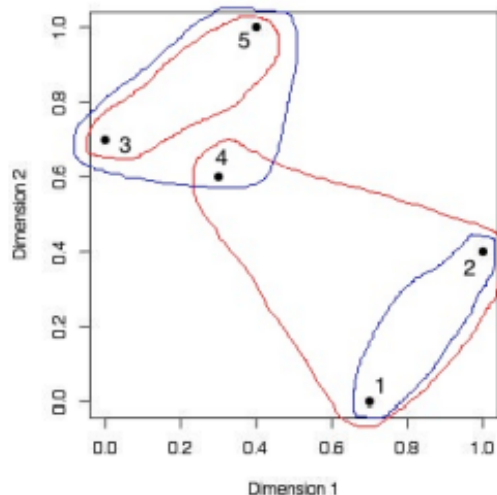
$$W = \frac{1}{2} \sum_{k=1}^K \frac{1}{n_k} \sum_{C(i)=k, C(j)=k} d_{ij}$$

Smaller  $W$  is better

# Simple Simulated 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

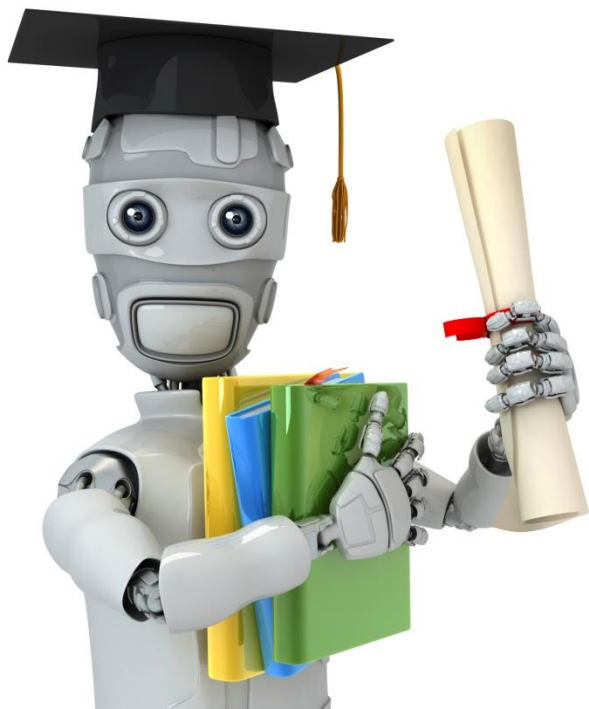
$$A(n, K) = \frac{1}{K!} \sum_{k=1}^K (-1)^{K-k} \binom{K}{k} k^n$$

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**





Machine Learning

# K-Means:

---

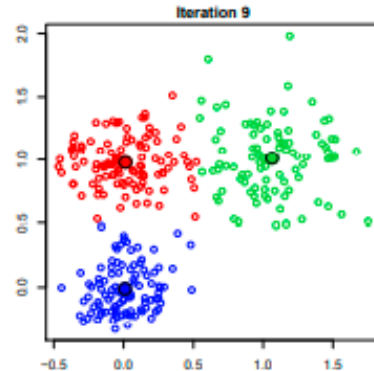
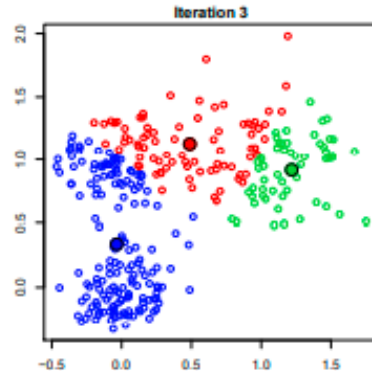
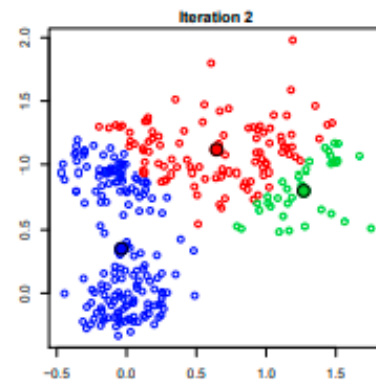
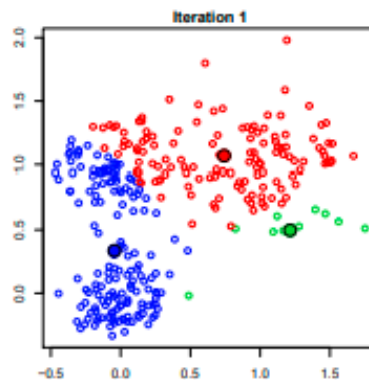
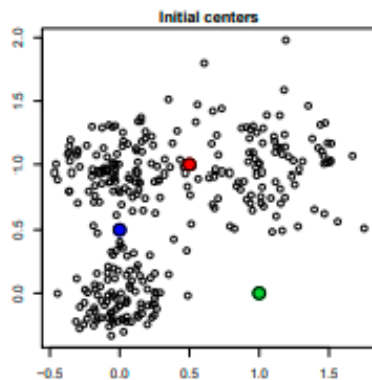
# k-Means Algorithm



- ▶ *k*-means: a classic partitioning method for clustering
- ▶ First, it selects *k* objects from the dataset, each of which initially represents a cluster center.
- ▶ Each object is assigned to the cluster to which it is most similar, based on the distance between the object and the cluster center.
- ▶ The means of clusters are computed as the new cluster centers.
- ▶ The process iterates until the criterion function converges.

# K-Means Example

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



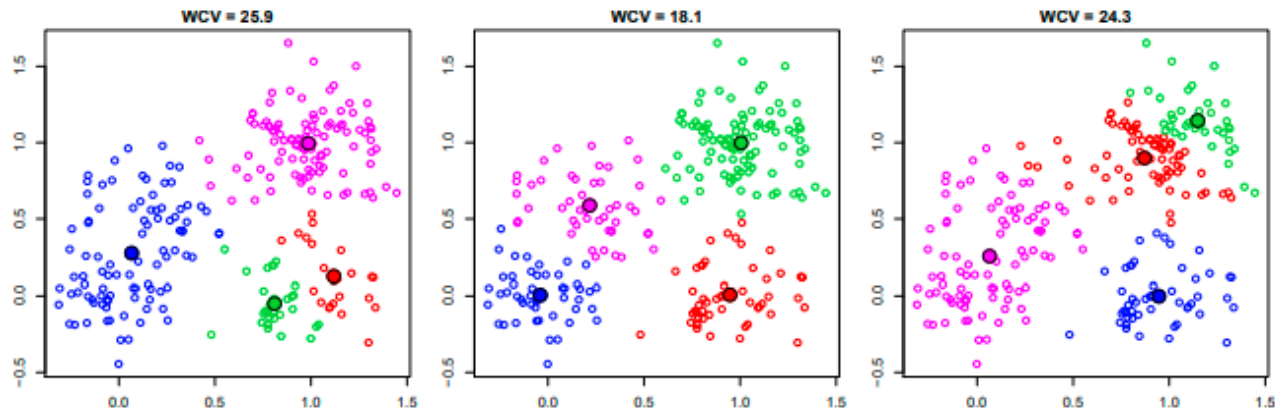
# Properties of K-Means

---

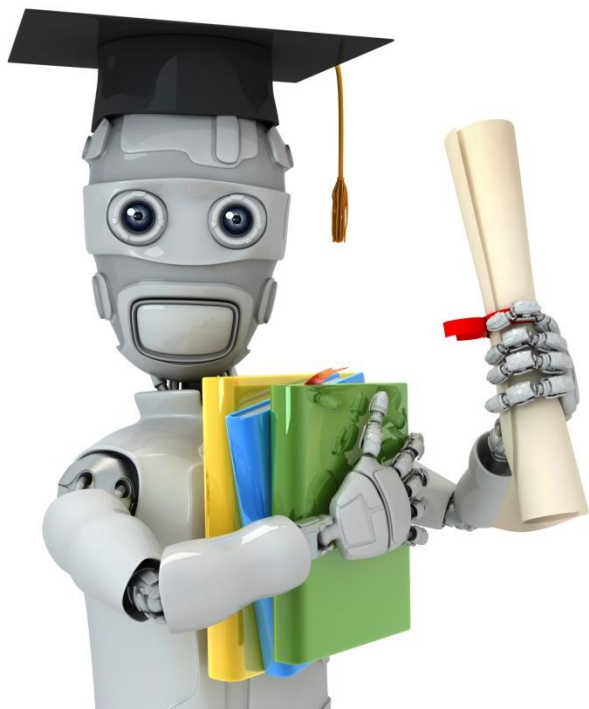
- ▶ Within-cluster variation **decreases** with each iteration of the algorithm. I.e., if  $W_t$  is the within-cluster variation at iteration  $t$ , then  $W_{t+1} \leq W_t$  (Homework 1)
- ▶ The algorithm **always converges**, no matter the initial cluster centers. In fact, it takes  $\leq K^n$  iterations (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**



# K-Medoids:

---

Machine Learning

# K-Medoids Algorithm

---

In some applications we want each center to be **one of the points** itself. This is where ***K*-medoids** comes in—an algorithm similar to the *K*-means algorithm, except when fitting the centers  $c_1, \dots, c_K$ , we restrict our attention to the points themselves

Initial guess for centers  $c_1, \dots, c_K$  (e.g., randomly select  $K$  of the points  $X_1, \dots, X_n$ ), 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  $c_k = X_k^*$ , the **medoid** of points in cluster  $k$ , i.e., the point  $X_i$  in cluster  $k$  that minimizes  $\sum_{C(j)=k} \|X_j - X_i\|_2^2$

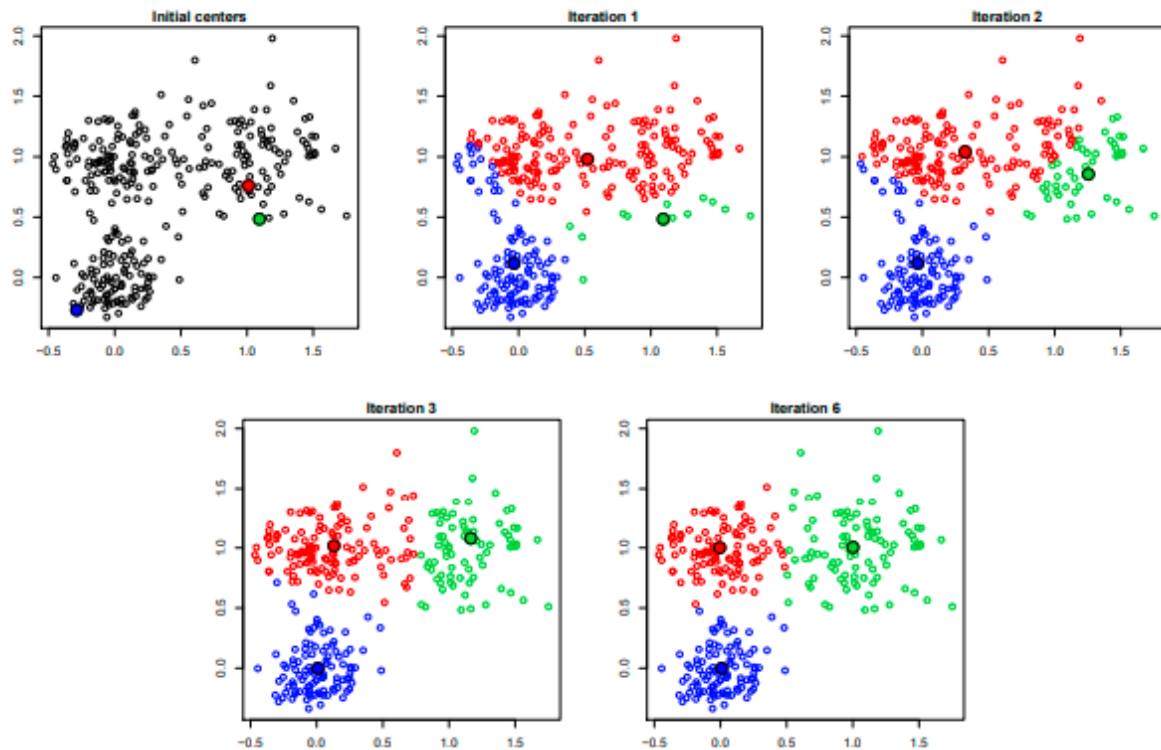
Stop when within-cluster variation doesn't change

In words:

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

# K-Medoids Example

---



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



# Properties of K- Medoids

---

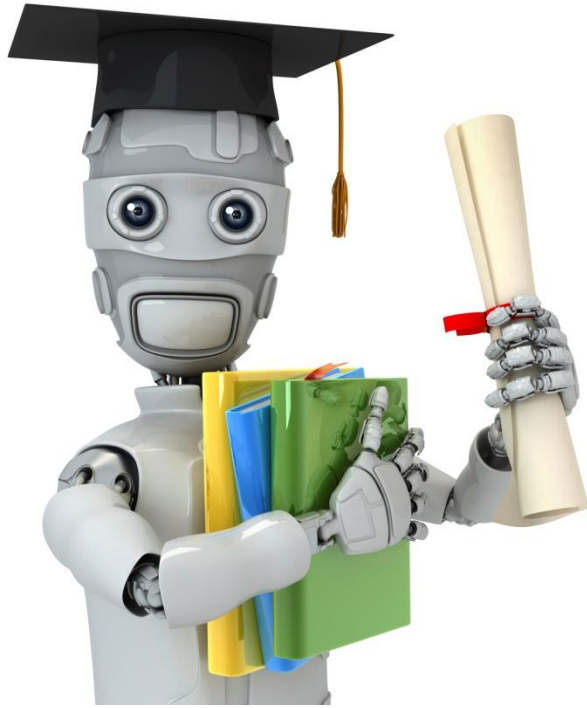
The  $K$ -medoids algorithm **shares the properties** of  $K$ -means that we discussed (each iteration decreases the criterion; the algorithm always converges; different starts gives different final answers; it does not achieve the global minimum)

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

Remember,  $K$ -medoids has the (potentially important) property that the centers are located among the data points themselves



# K-means and K-medoids in R:

---

Machine Learning

# k-means in R

---

```
## k-means clustering set a seed for random number generation to
## make the results reproducible
set.seed(8953)
## make a copy of iris data
iris2 <- iris
## remove the class label, Species
iris2$Species <- NULL
## run kmeans clustering to find 3 clusters
kmeans.result <- kmeans(iris2, 3)
```

```
## K-means clustering with 3 clusters of sizes 38, 50, 62
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1      6.850000      3.073684      5.742105      2.071053
## 2      5.006000      3.428000      1.462000      0.246000
## 3      5.901613      2.748387      4.393548      1.433871
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2...
##  [31] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 3 3 3 3...
##  [61] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3...
##  [91] 3 3 3 3 3 3 3 3 3 3 1 3 1 1 1 1 3 1 1 1 1 1 1 3 3 1 1...
## [121] 1 3 1 3 1 1 3 3 1 1 1 1 1 3 1 1 1 1 3 1 1 1 3 1 1 1 3...
##
## Within cluster sum of squares by cluster:
## [1] 23.87947 15.15100 39.82097
## (between_SS / total_SS =  88.4 %)
```

Check clustering result against class labels (Species)

```
table(iris$Species, kmeans.result$cluster)
##
##           1  2  3
##   setosa    0 50  0
##   versicolor 2  0 48
##   virginica 36  0 14
```

# Results

- ▶ Class “setosa” can be easily separated from the other clusters
- ▶ Classes “versicolor” and “virginica” are to a small degree overlapped with each other.



# K-Means With Estimating k and Initialisations in R

---

- ▶ `kmeansruns()` in package *fpc* [Hennig, 2014]
- ▶ calls `kmeans()` to perform *k*-means clustering
- ▶ initializes the *k*-means algorithm several times with random points from the data set as means
- ▶ estimates the number of clusters by Calinski Harabasz index or average silhouette width

```
library(fpc)
kmeansruns.result <- kmeansruns(iris2)
kmeansruns.result

## K-means clustering with 3 clusters of sizes 62, 50, 38
##
## Cluster means:
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1    5.901613    2.748387    4.393548    1.433871
## 2    5.006000    3.428000    1.462000    0.246000
## 3    6.850000    3.073684    5.742105    2.071053
##
## Clustering vector:
##   [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2...
##  [31] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2...
##  [61] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1...
##  [91] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1...
## [121] 3 1 3 1 3 3 1 1 3 3 3 3 3 3 1 3 3 3 3 3 1 3 3 3 3 1 3 3 3 1...
##
## Within cluster sum of squares by cluster:
## [1] 39.82097 15.15100 23.87947
## (between_SS / total_SS =  88.4 %)
```

# K-medoids in R:

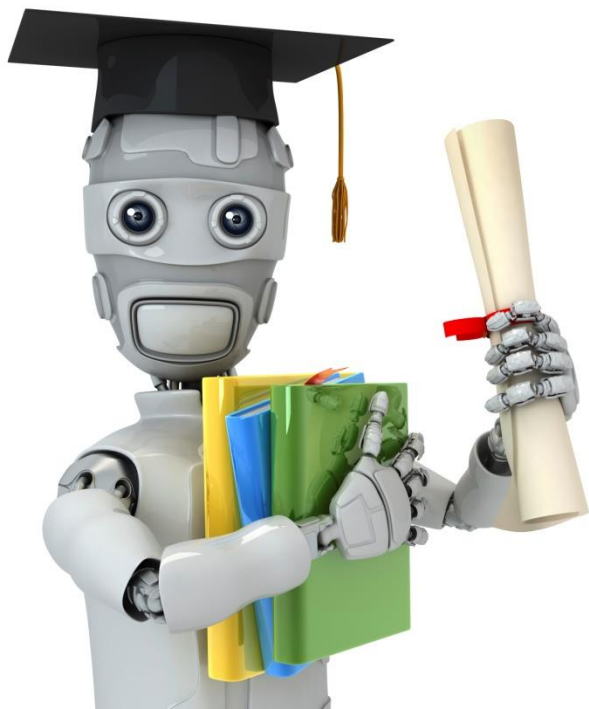


- ▶ more robust than  $k$ -means in presence of outliers
- ▶ PAM (Partitioning Around Medoids) is a classic algorithm for  $k$ -medoids clustering.
- ▶ The CLARA algorithm is an enhanced technique of PAM by drawing multiple samples of data, applying PAM on each sample and then returning the best clustering. It performs better than PAM on larger data.
- ▶ Functions `pam()` and `clara()` in package *cluster* [Maechler et al., 2016]
- ▶ Function `pamk()` in package *fpc* does not require a user to choose  $k$ .

## Clustering with `pam()`

```
## clustering with PAM
library(cluster)
# group into 3 clusters
pam.result <- pam(iris2, 3)
# check against actual class label
table(pam.result$clustering, iris$Species)

##
##      setosa versicolor virginica
##  1       50           0          0
##  2        0          48         14
##  3        0           2         36
```



# Hierarchical Clustering:

---

Machine Learning

# 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 dissimilarities  $d_{ij}$  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

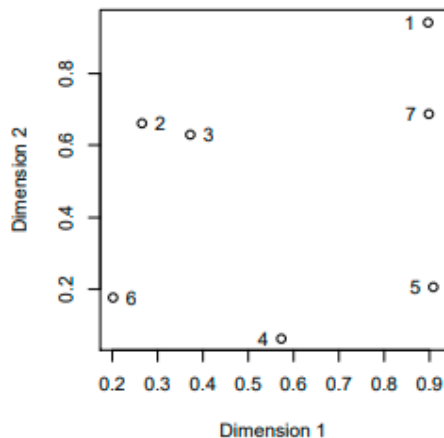
**Divisive** (i.e., top-down):

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

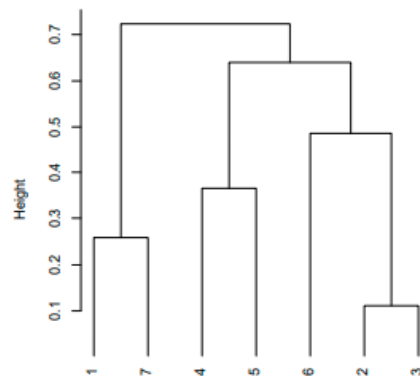
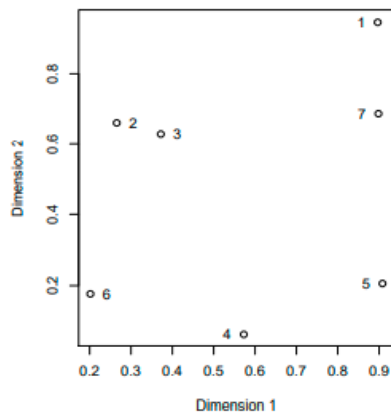
Step 3:  $\{1, 7\}, \{2, 3\}, \{4\}, \{5\}, \{6\}$ ;

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



We can also represent the sequence of clustering assignment **dendrogram**:

Note that **cutting the dendrogram** horizontally partitions the 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 dissimilarity between its two daughter nodes

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 distance, not squared distance)

# 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

Agglomerative clustering, given the linkage:

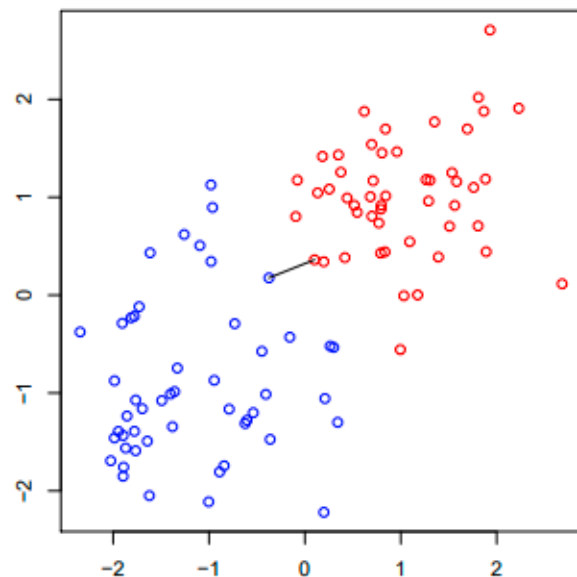
- ▶ Start with all points in their own group
- ▶ Until there is only one cluster, repeatedly: merge the two groups  $G, H$  such that  $d(G, H)$  is smallest

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

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

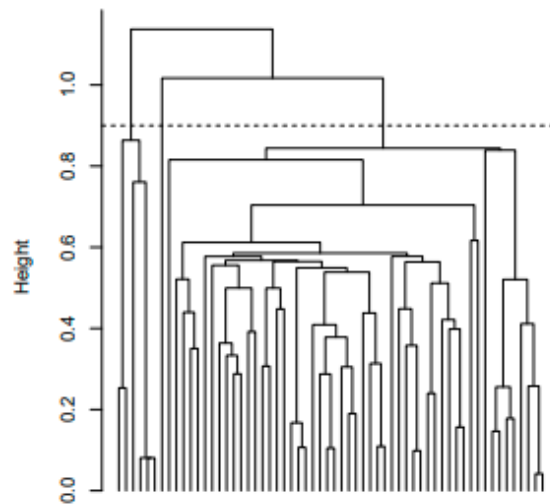
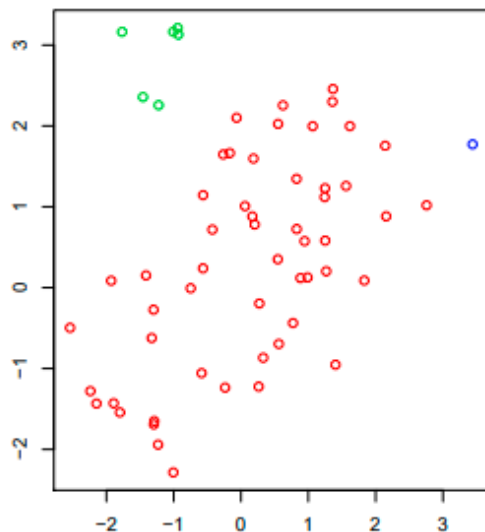
# Single Linkage

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  $d_{ij} \leq 0.9$

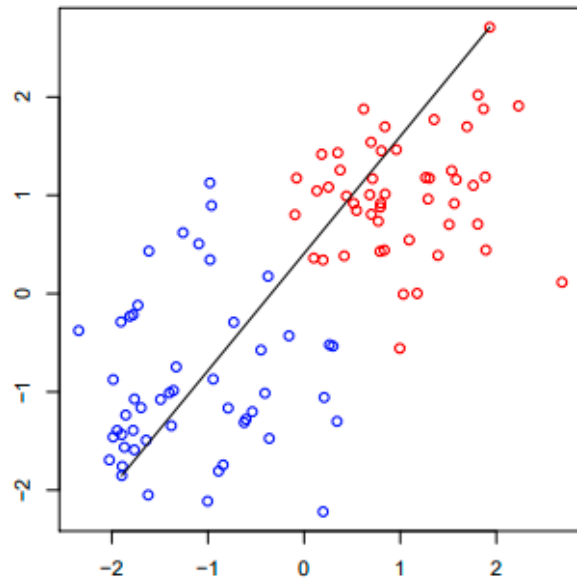
In **complete linkage** (i.e., furthest-neighbor linkage), dissimilarity between  $G, H$  is the largest dissimilarity between two points in opposite groups:

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

# Complete Linkage

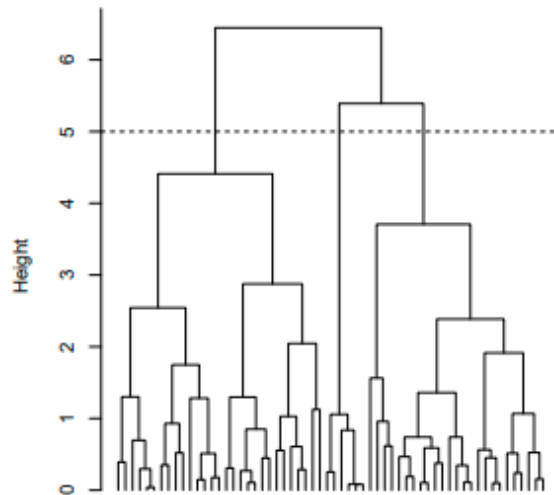
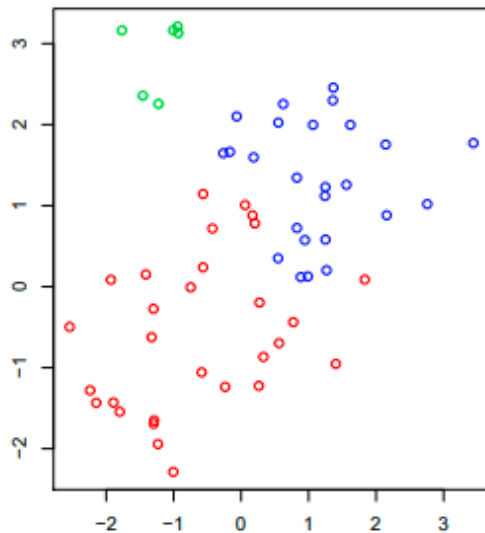
---

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  $d_{ij} \leq 5$



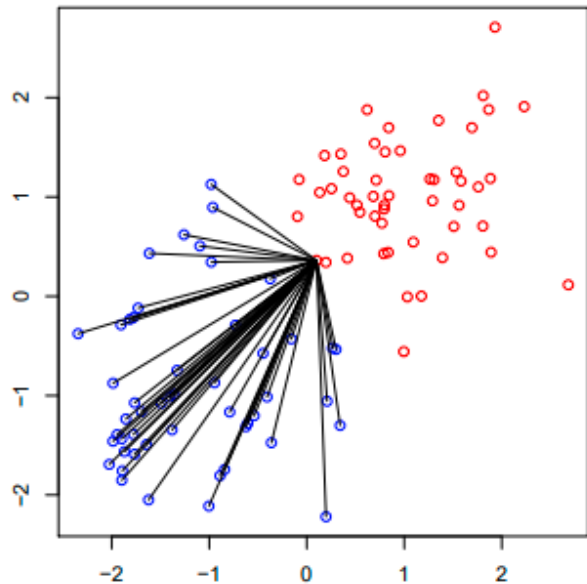
# Average Linkage

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

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

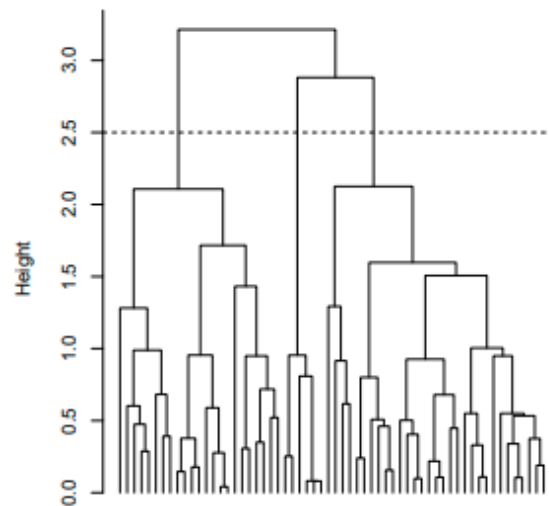
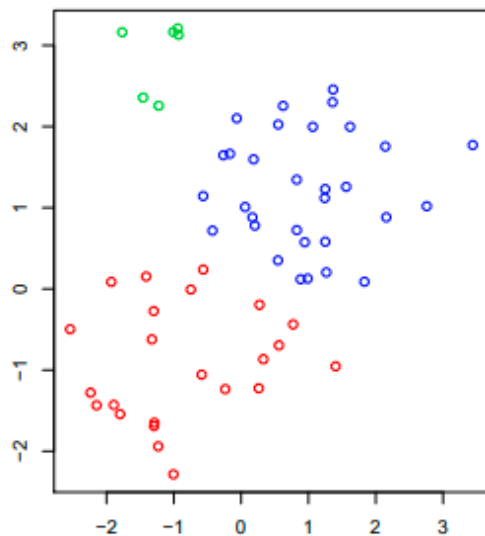
Example (dissimilarities  $d_{ij}$  are distances, groups are marked by colors): average linkage score  $d_{\text{average}}(G, H)$  is the **average distance** across all pairs

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



# Average Linkage Example

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



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

# Shortcomings of Single, Complete Linkage

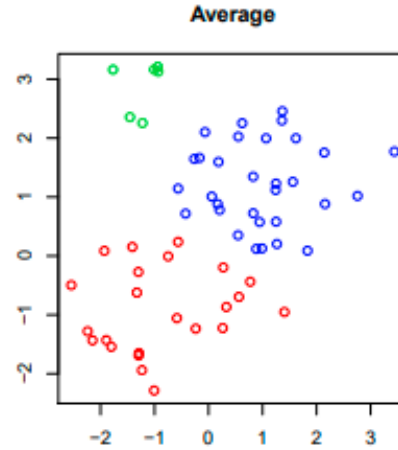
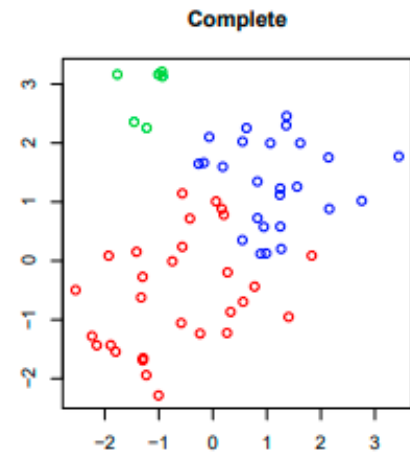
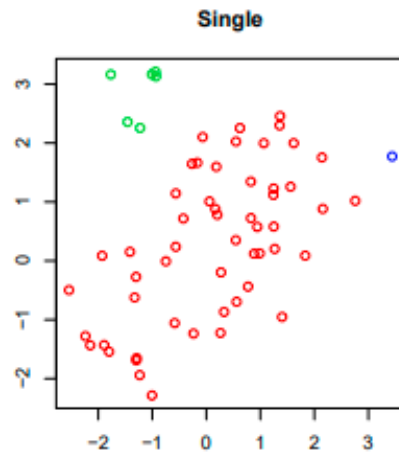
Single and complete linkage can have some practical problems:

- ▶ 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

# Example of Chaining and Crowding



# Shortcomings of Average Linkage

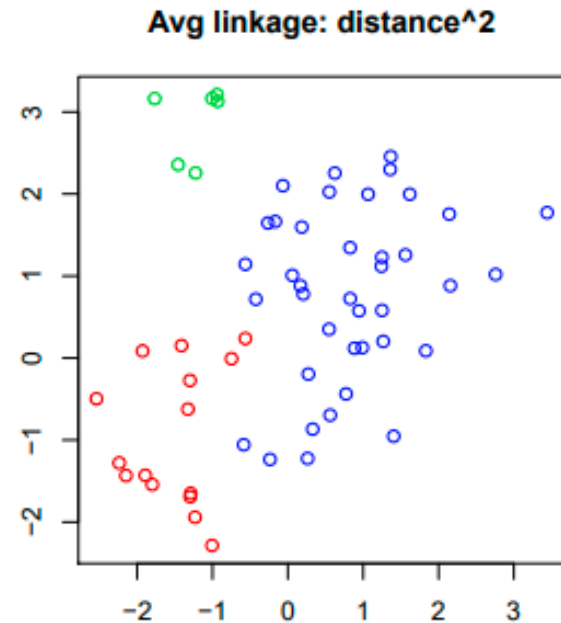
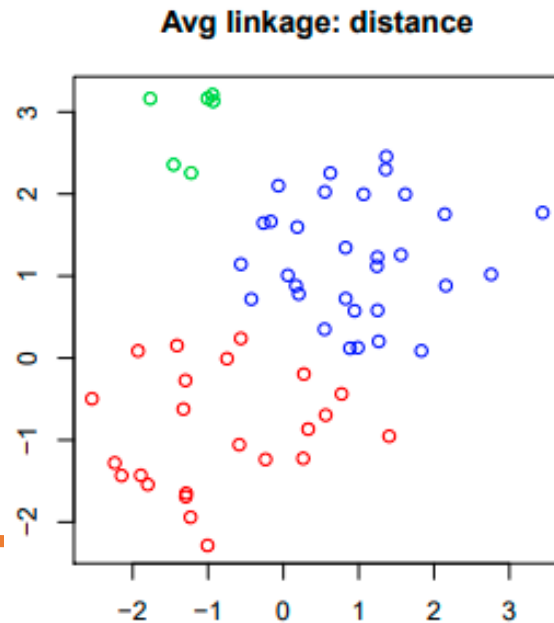
Average linkage isn't perfect, it has its own problems:

- ▶ 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 (Homework 1)

# Example of a Change with Monotone Increasing Transformation

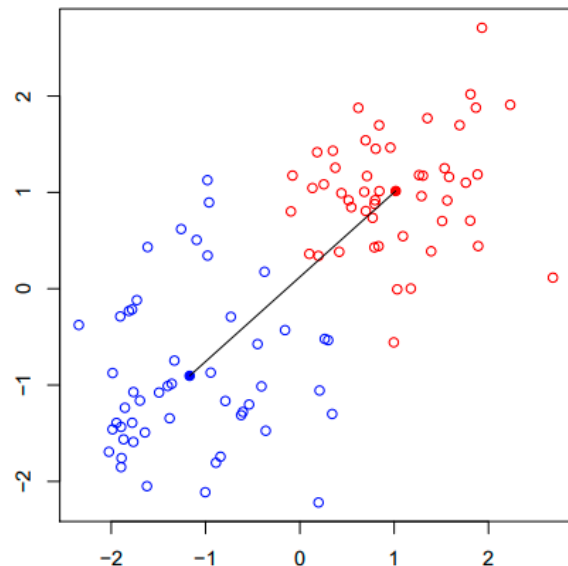


# Centroid Linkage

Centroid linkage<sup>1</sup> 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:

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

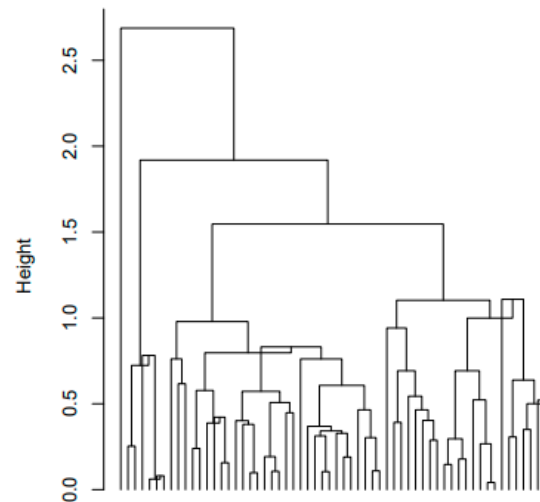
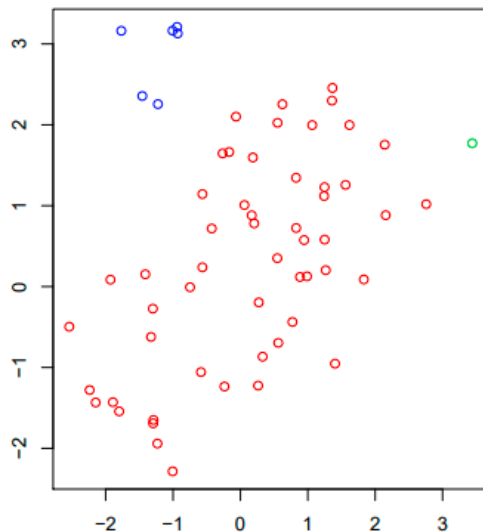
Example (dissimilarities  $d_{ij}$  are distances, groups are marked by colors): centroid linkage score  $d_{\text{centroid}}(G, H)$  is the **distance between** the group centroids (i.e., group averages)



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

# Centroid Linkage Example

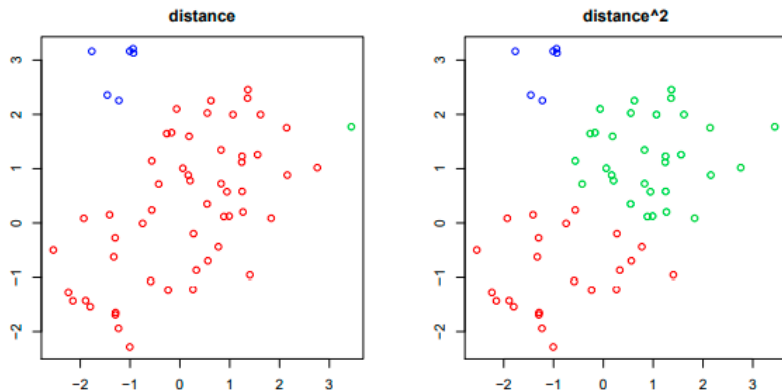
Here  $n = 60$ ,  $X_i \in \mathbb{R}^2$ ,  $d_{ij} = \|X_i - X_j\|_2$ . Cutting the tree at some heights wouldn't make sense ... because the dendrogram has **inversions**! But we can, e.g., still look at output with 3 clusters





# Shortcomings of centroid linkage

- ▶ Can produce dendrograms with **inversions**, which really messes up the visualization
- ▶ Even if we were 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

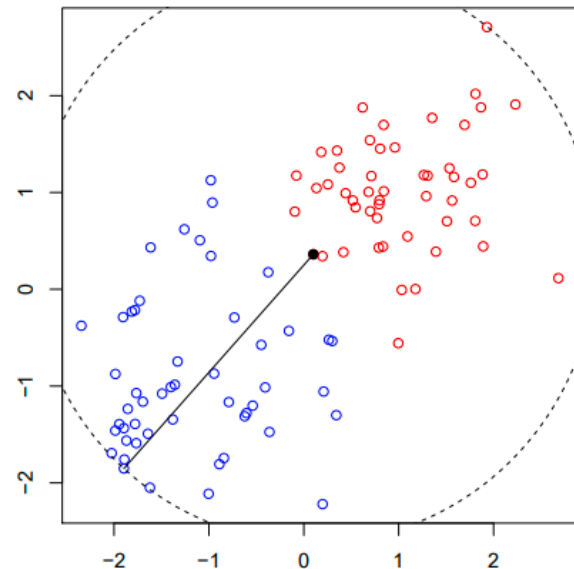


**Minimax linkage** is a newcomer. First define radius of a group of points  $G$  around  $X_i$  as  $r(X_i, G) = \max_{j \in G} d_{ij}$ . Then:

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

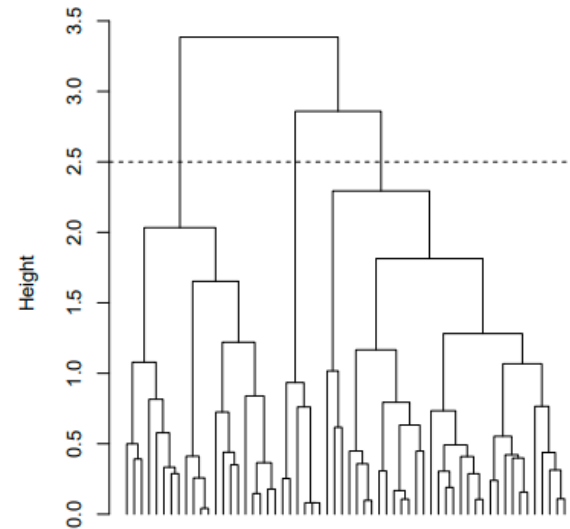
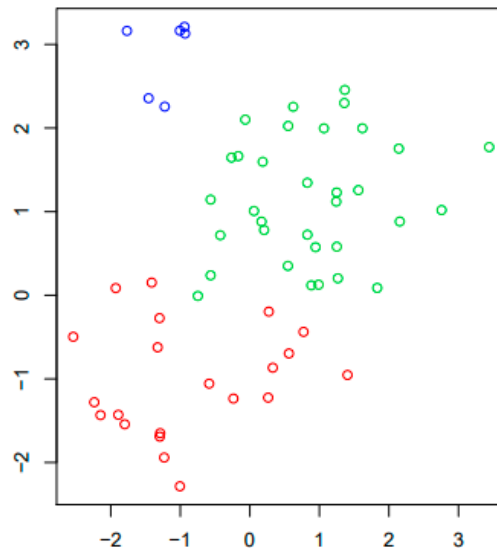
# Minimax Linkage

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



# Minimax Linkage Example

Same data  $s$  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)

# Recup

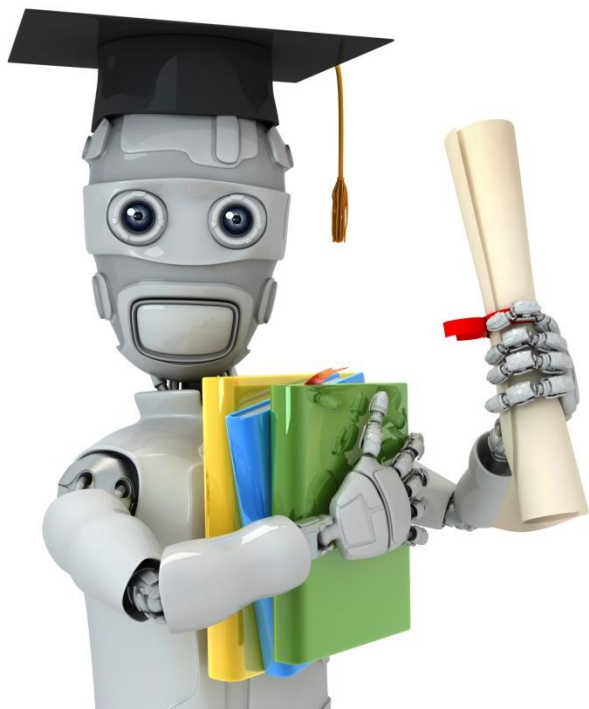
**Hierarchical agglomerative clustering:** start with all data points in their own groups, and repeatedly merge groups, based on linkage function. Stop when points are in one group (this is agglomerative; there is also divisive)

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

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



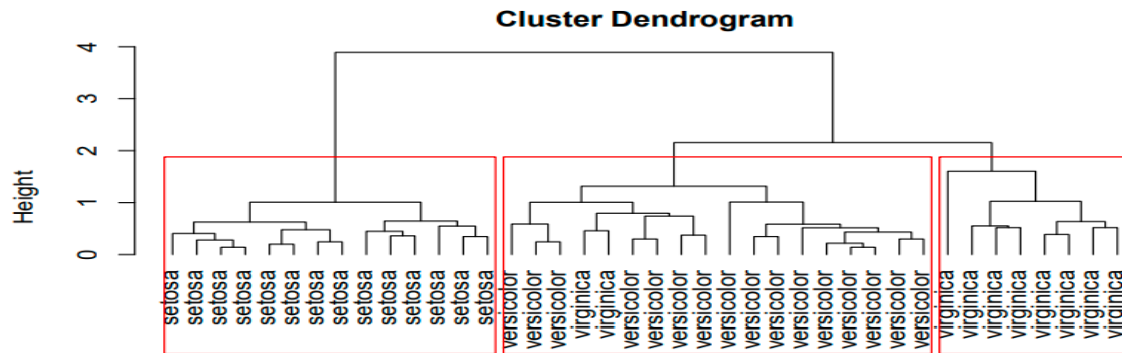
# Hierarchical Clustering In R:

---

Machine Learning

# Hierarchical Clustering of the Iris Data

```
## hiercrchical clustering
set.seed(2835)
# draw a sample of 40 records from the iris data, so that the
# clustering plot will not be over crowded
idx <- sample(1:dim(iris)[1], 40)
iris3 <- iris[idx, ]
# remove class label
iris3$Species <- NULL
# hierarchical clustering
hc <- hclust(dist(iris3), method = "ave")
# plot clusters
plot(hc, hang = -1, labels = iris$Species[idx])
# cut tree into 3 clusters
rect.hclust(hc, k = 3)
# get cluster IDs
groups <- cutree(hc, k = 3)
```



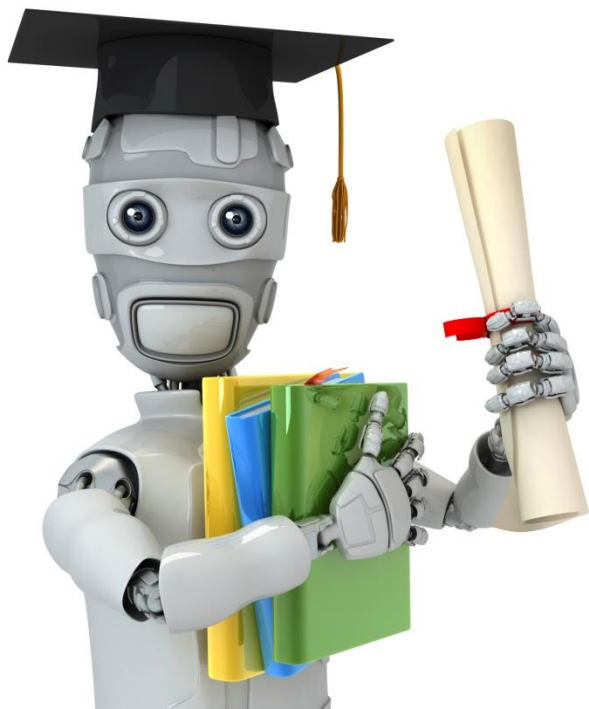
# Parameters



```
hclust(d, method = "complete", members = NULL)
```

- ▶ `method = "ward.D"` or `"ward.D2"`: Ward's minimum variance method aims at finding compact, spherical clusters [R Core Team, 2015].
- ▶ `method = "complete"`: complete-link distance; finds similar clusters.
- ▶ `method = "single"`: single-link distance; adopts a "friends of friends" clustering strategy.
- ▶ `method = "average"`: average distance
- ▶ `method = "centroid"`: centroid distance
- ▶ `method = "median"`:
- ▶ `method = "mcquitty"`:





Machine Learning

# How Many Clusters?

---

# 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, e.g., designing a clever radio system or placing cell phone towers

---

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. Usually, statistical learning is successful when at least one of these is possible

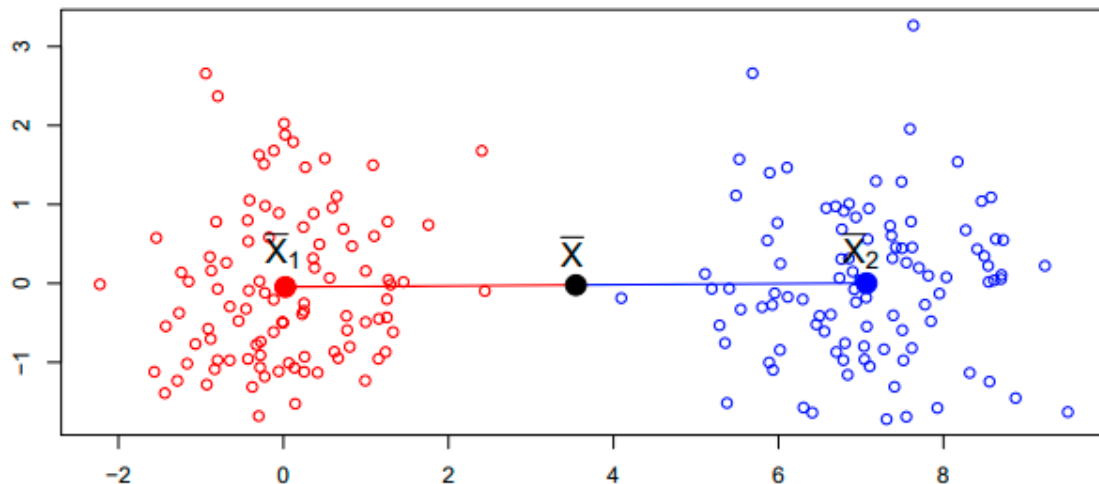
Why is it important?

- ▶ 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

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

# Reminder: Within and Between Cluster Variation

---



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

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


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**.<sup>3</sup> For clustering assignments coming from  $K$  clusters, we record CH score:

## CH Index

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

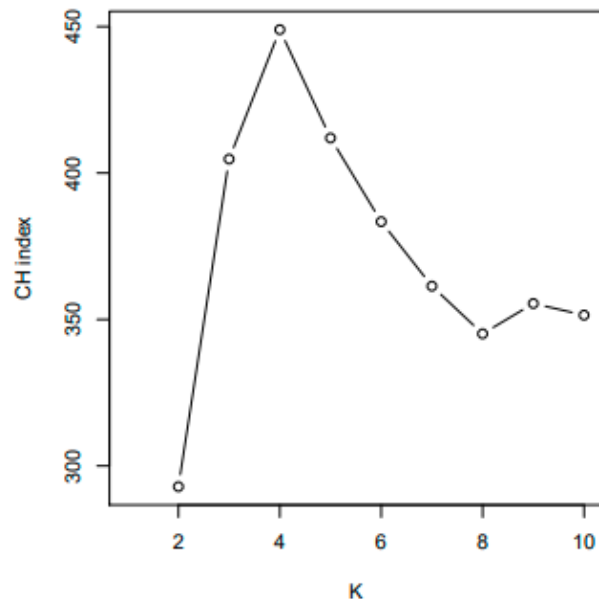
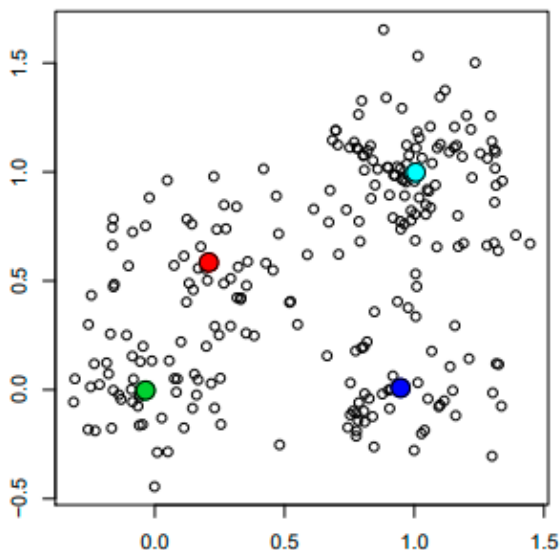
To choose  $K$ , just pick some maximum number of clusters to be considered  $K_{\max}$  (e.g.,  $K = 20$ ), and choose the value of  $K$  with the largest score  $\text{CH}(K)$ , i.e.,



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

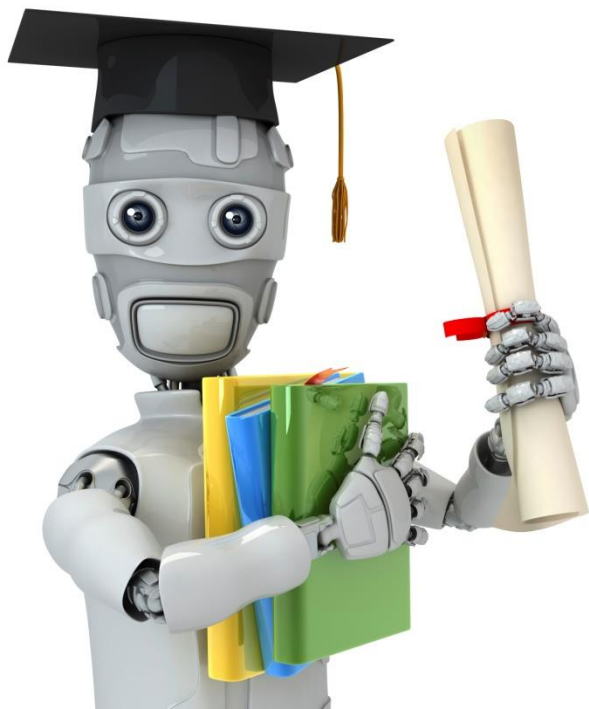
Running example:  $n = 250$ ,  $p = 2$ ,  $K = 2, \dots, 10$ .

## Example: CH Index



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)!



Machine Learning

# See you next chapter

---

# Thanks :)