Today

- (Ch 12) Clustering
  - The curse of dimensionality
  - Multivariate normal distribution
  - The clustering problem
  - k-means algorithm

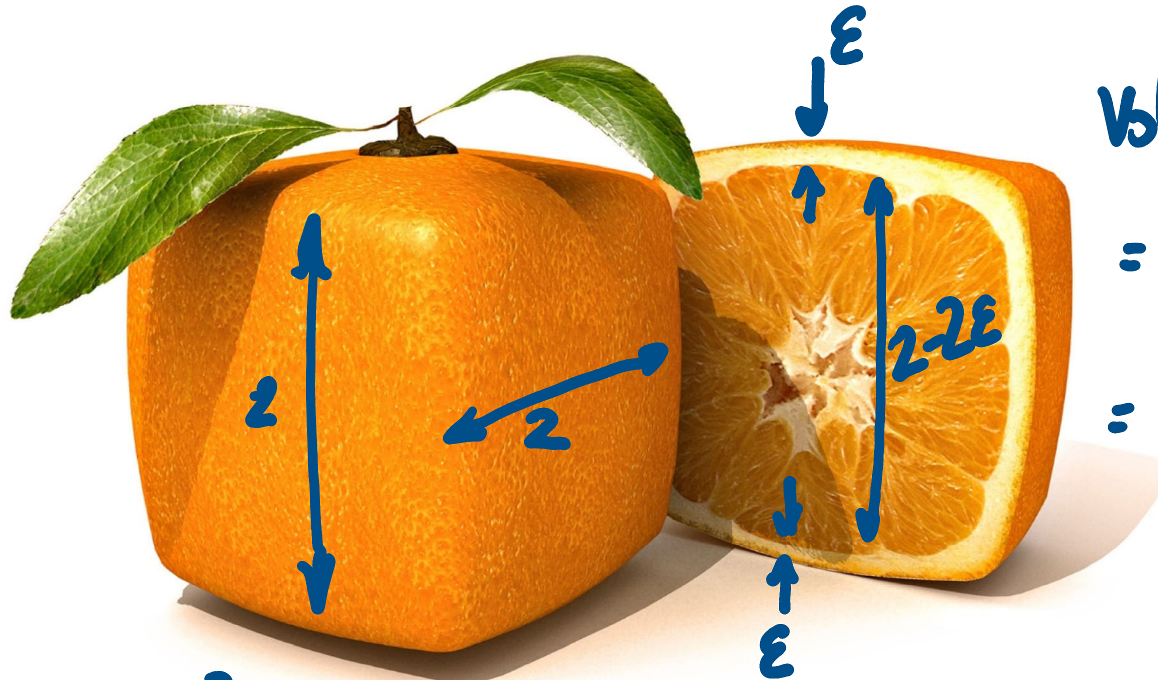
Next lecture

- (Ch 12) Clustering
  - k-means algorithm
  - Vector quantization

How much of a cubic orange is peel?

Volume of orange

$= 2^3$



Volume of fruity part

$= (2-2\varepsilon)^3$

$= 2^3(1-\varepsilon)^3$

Fraction that is peel $= \dfrac{2^3 - 2^3(1-\varepsilon)^3}{2^3} = 1-(1-\varepsilon)^3$

What about a d-dimensional cubic orange?

- Total amount of orange $= 2^d$
- Amount of fruity part $= (2-2\varepsilon)^d = 2^d(1-\varepsilon)^d$
- Fraction of orange that is peel $= 1-(1-\varepsilon)^d \to 1$ as $d \to \infty$

A high dimensional orange is virtually all peel

The curse of dimensionality

- If a dataset is uniformly distributed in a high-dimensional cube (or some other shape), the vast majority of data is far from the origin

- We can also prove that the distance between data points grows with increasing dimensions

- A d-dimensional histogram of the dataset is not very useful because
  - Most bins will be empty
  - Some bins will contain a single data point
  - Very few bins will contain more than one point

Dealing with data in high dimensions

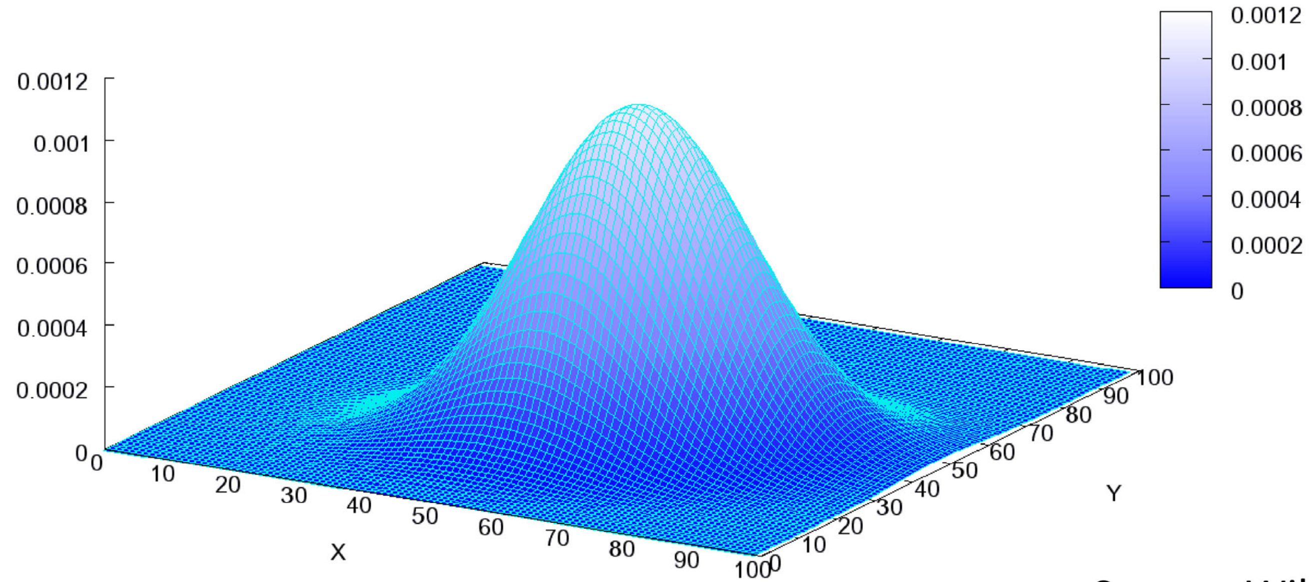
- Collect as much data as possible

- Cluster data points together into one or more blobs

- Fit a simple probability model to each blob

Multivariate normal distribution

- Extension of the normal distribution to multiple dimensions
- Example: bivariate (2-dimensional) normal distribution



Source: Wikipedia

Multivariate normal probability density

A multivariate normal random vector $\mathbf{X}$ of dimension d has density

$$P(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right)$$

$$-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}$$

where

- $\boldsymbol{\mu} = E[\mathbf{X}]$ is a d-dimensional vector called the mean
- $\Sigma = E[(\mathbf{X}-\boldsymbol{\mu})(\mathbf{X}-\boldsymbol{\mu})^T]$ is a $d \times d$ symmetric and positive semidefinite matrix called the covariance matrix

Multivariate MLE

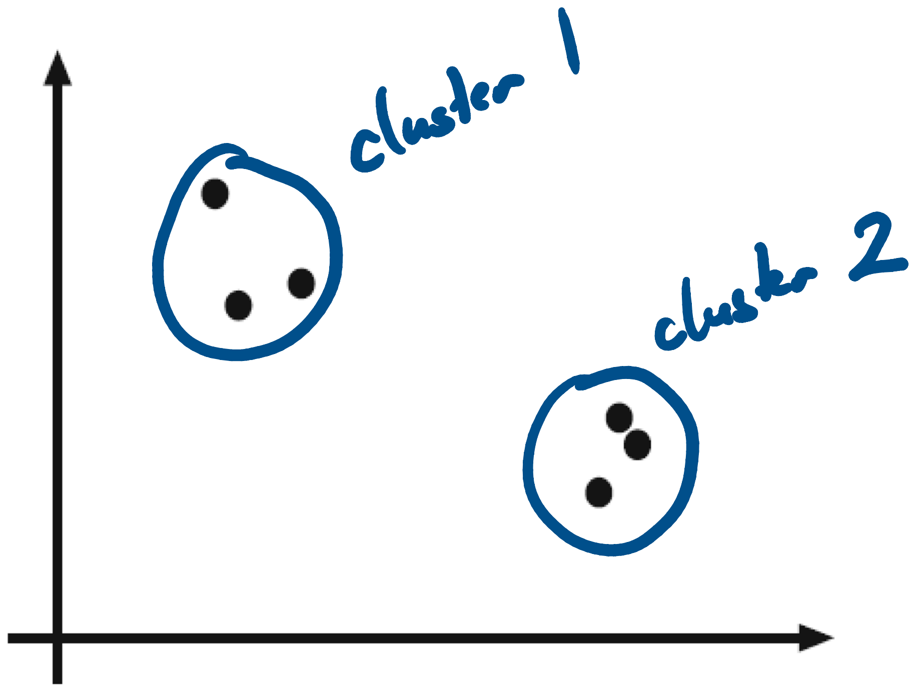
Given a d-dimensional dataset $\{\mathbf{x}\}$ consisting of N items, we can fit a multivariate normal distribution using maximum likelihood estimation

$$\hat{\boldsymbol{\mu}}_{MLE} = \text{mean}(\{\mathbf{x}\}) = \frac{\sum_i \mathbf{x}_i}{N}$$

$$\hat{\Sigma}_{MLE} = \text{Covmat}(\{\mathbf{x}\}) = \frac{\sum_i (\mathbf{x}_i - \text{mean}(\{\mathbf{x}\}))(\mathbf{x}_i - \text{mean}(\{\mathbf{x}\}))^T}{N}$$

The clustering problem

- Given a dataset $\{\mathbf{x}\}$, separate the data items into clusters so that
  - Items within a cluster are close to each other
  - Items in different clusters are far from each other
- There are two problems to solve
  - Determine the number of clusters
  - Assign each item to a cluster
- Note that we are taking unlabeled data and assigning a class label to each item



Clustering approaches

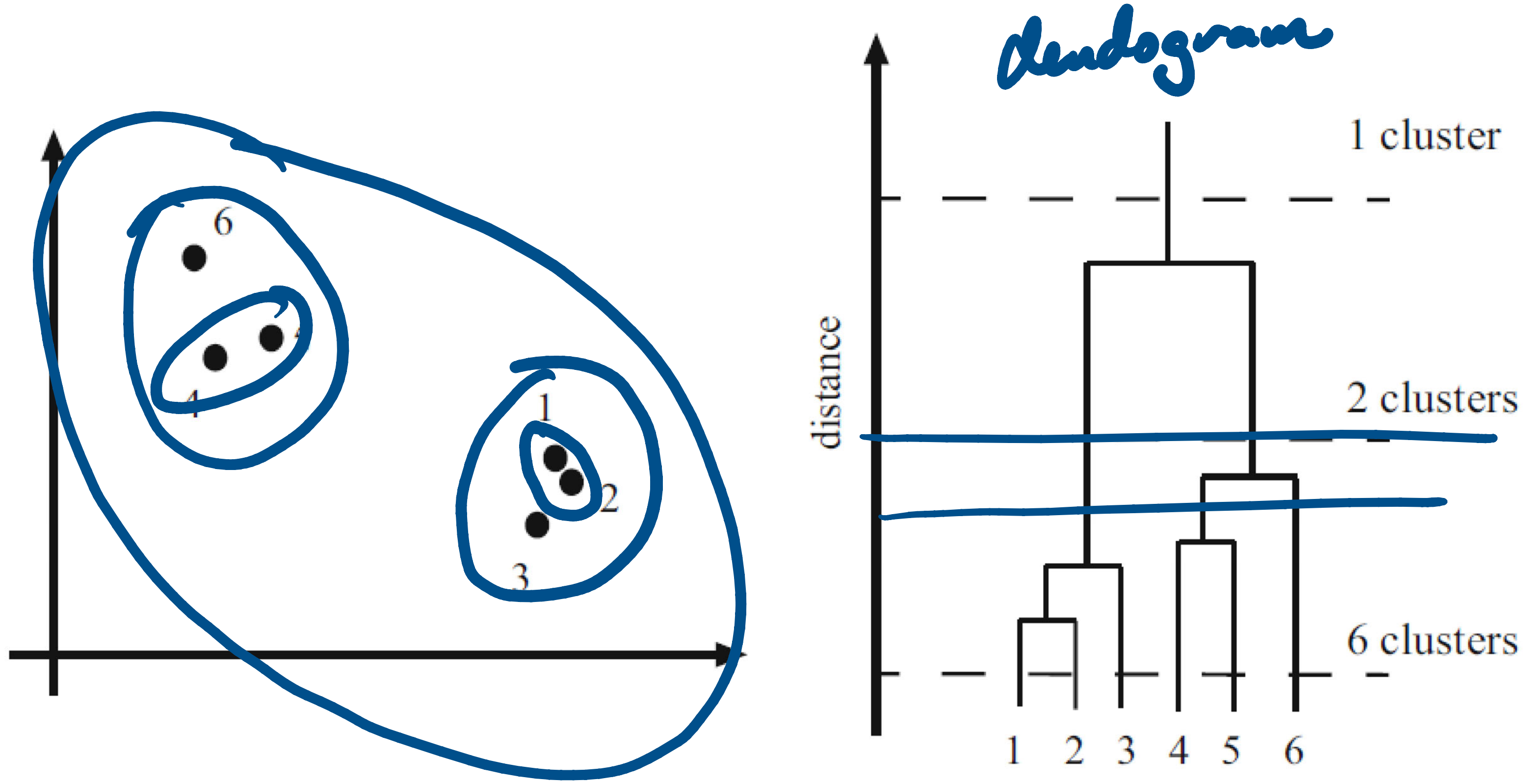
- Divisive clustering
  - Treat the whole dataset as a single cluster
  - Then split the dataset recursively until you get a satisfactory clustering

- Agglomerative clustering
  - Treat each data item as its own cluster
  - Then merge clusters until you get a satisfactory clustering

- Iterative clustering (such as k-means)

Agglomerative clustering: example

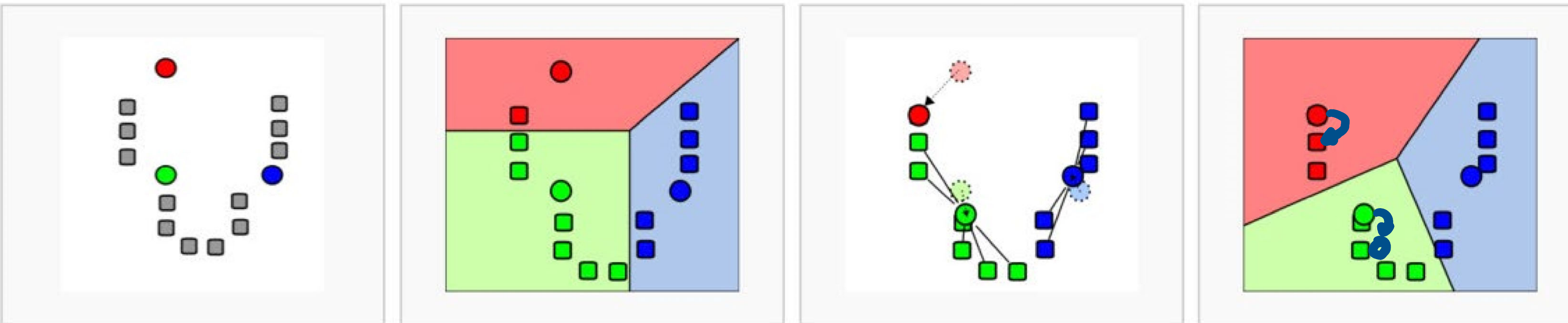
In this example the closest pair of clusters is merged at each step



k-means clustering

- Pick a value for k, which is the number of clusters
- Select k random cluster centers
- Iterate the following two steps until convergence
  - Assign each data item to the nearest cluster center
  - Update each cluster center as the mean of the items assigned to its cluster

k-means clustering: example



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

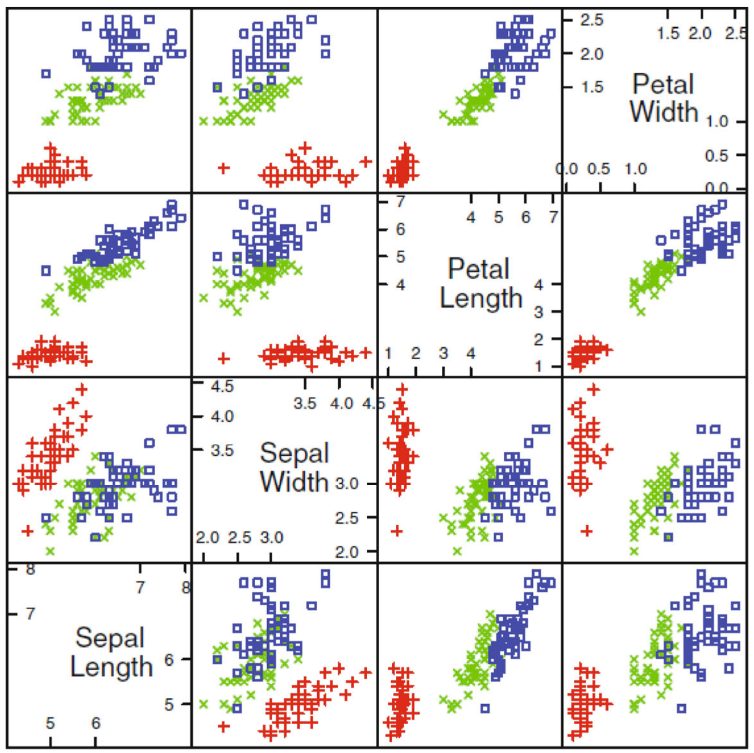
3. The centroid of each of the *k* clusters becomes the new mean.

4. Steps 2 and 3 are repeated until convergence has been reached.

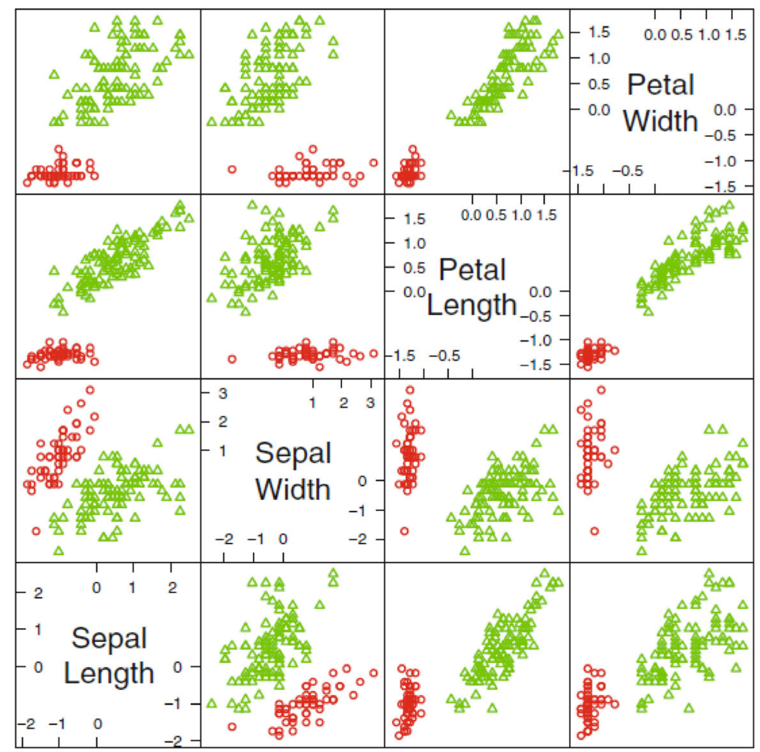
Source: Wikipedia

k-means clustering result: iris example

true labels

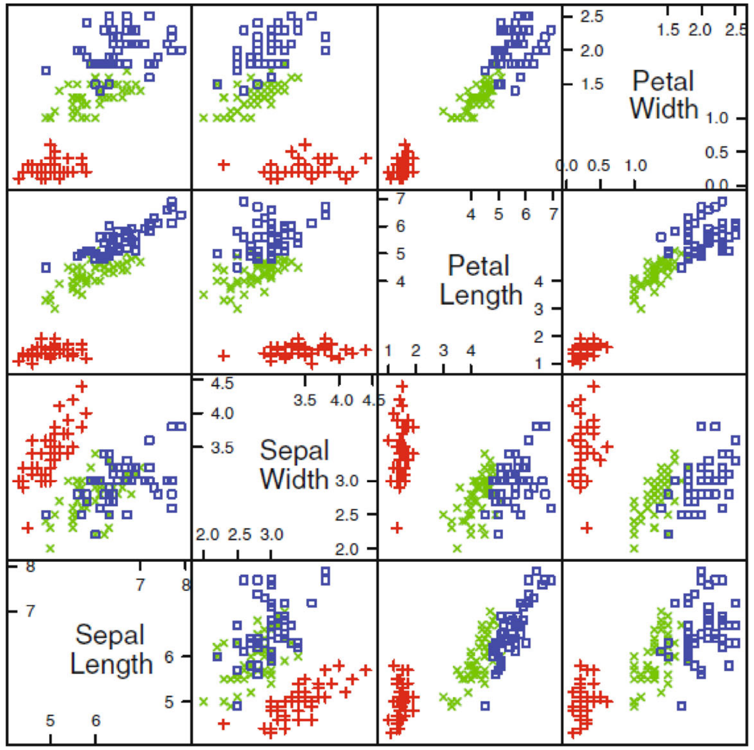


k-means with $k = 2$ clusters

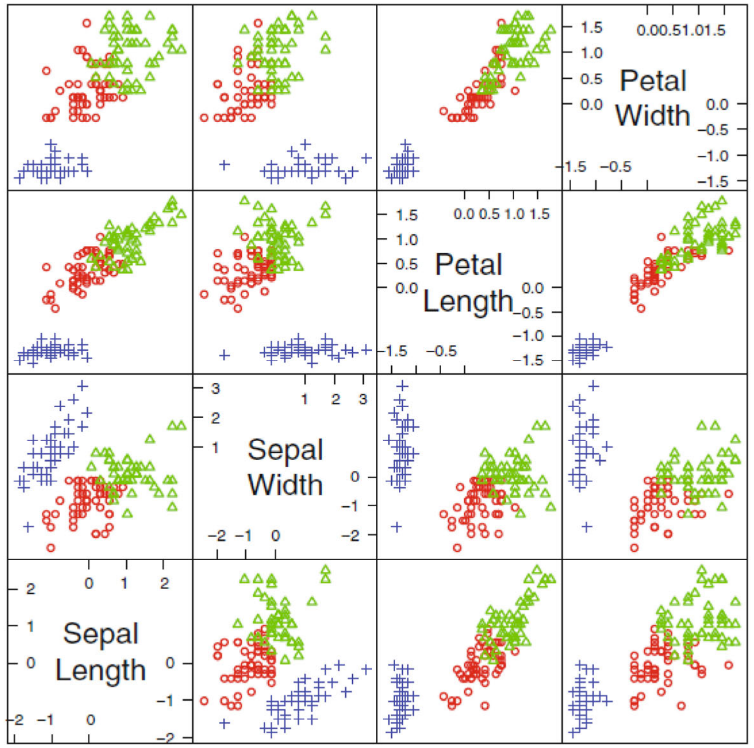


k-means clustering result: iris example

true labels

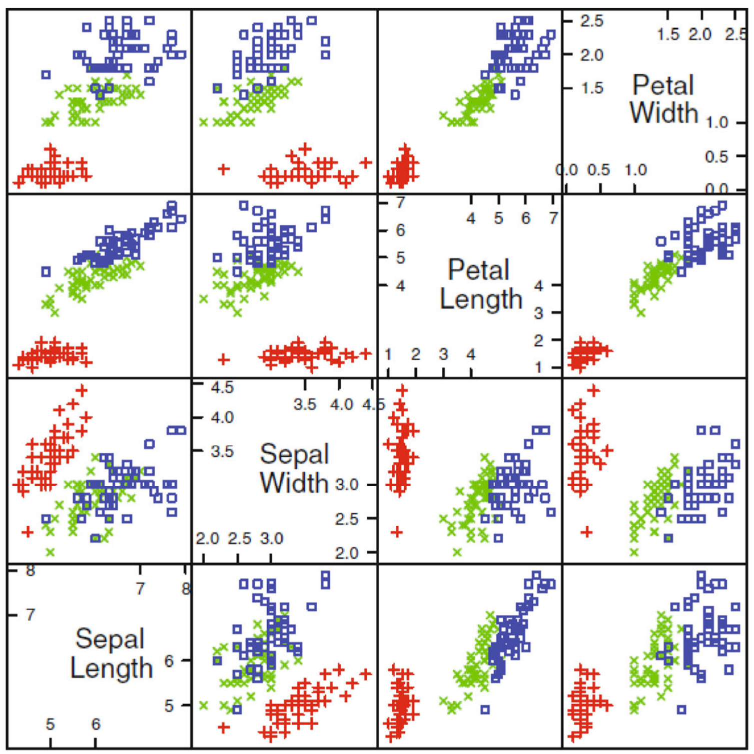


k-means with $k = 3$ clusters

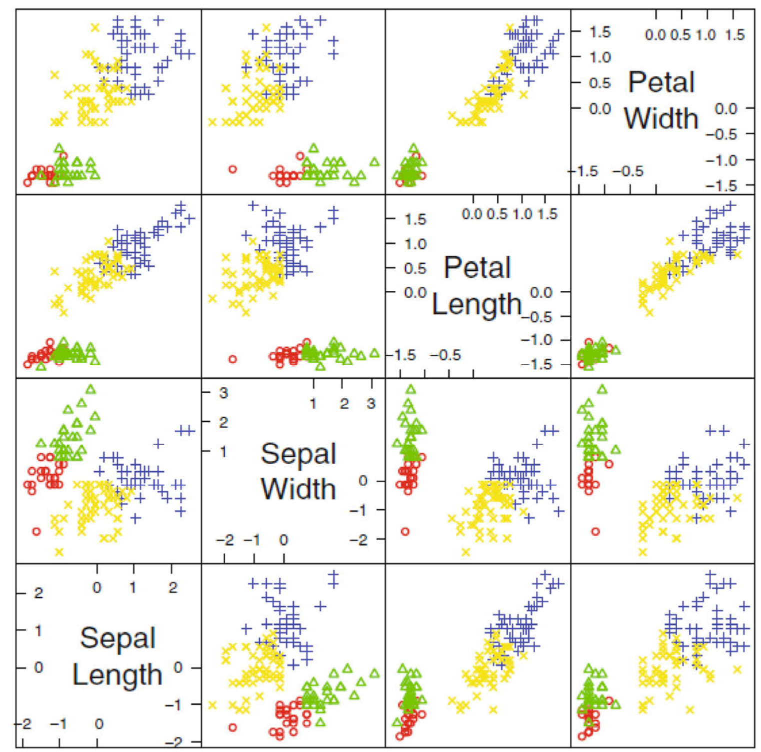


k-means clustering result: iris example

true labels

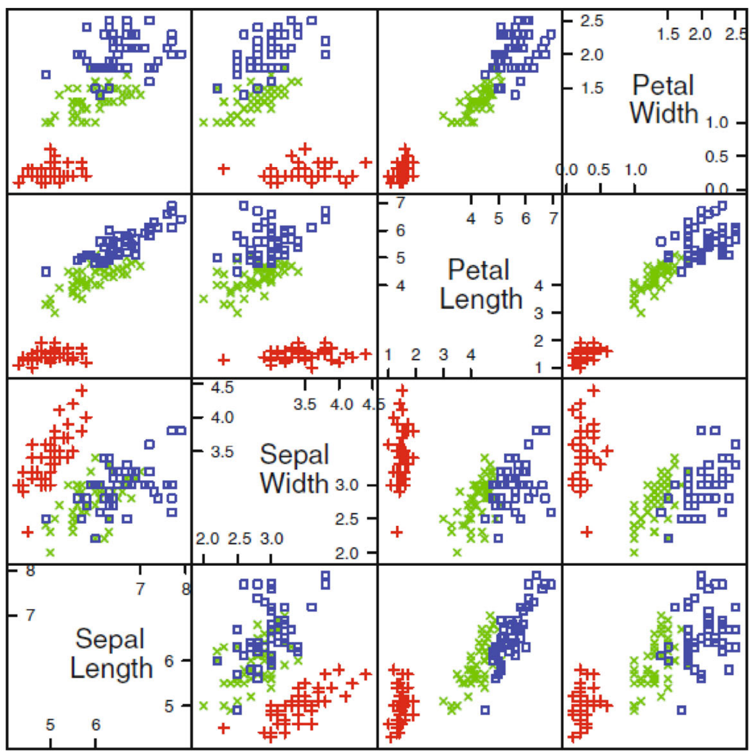


k-means with $k = 4$ clusters

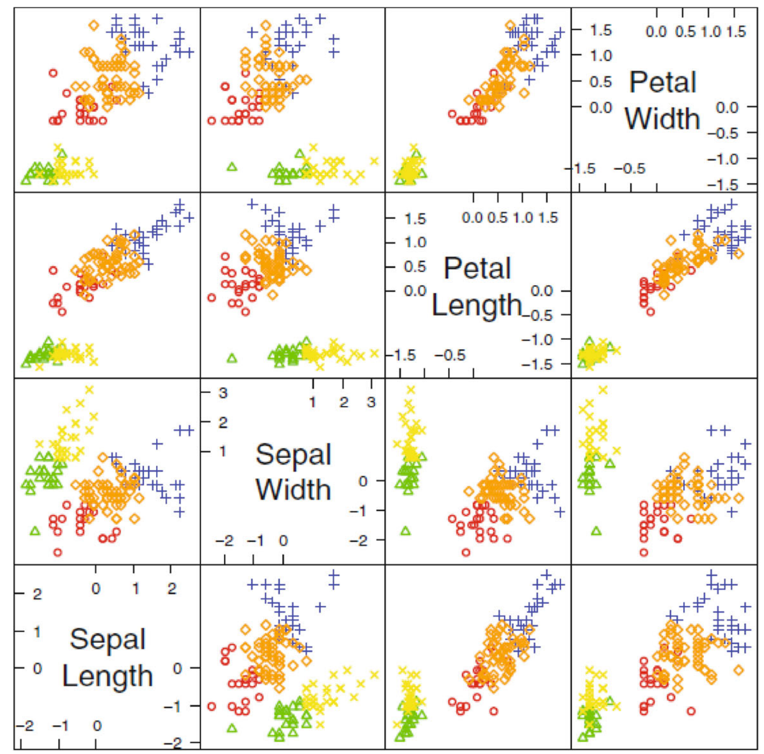


k-means clustering result: iris example

true labels



k-means with $k = 5$ clusters



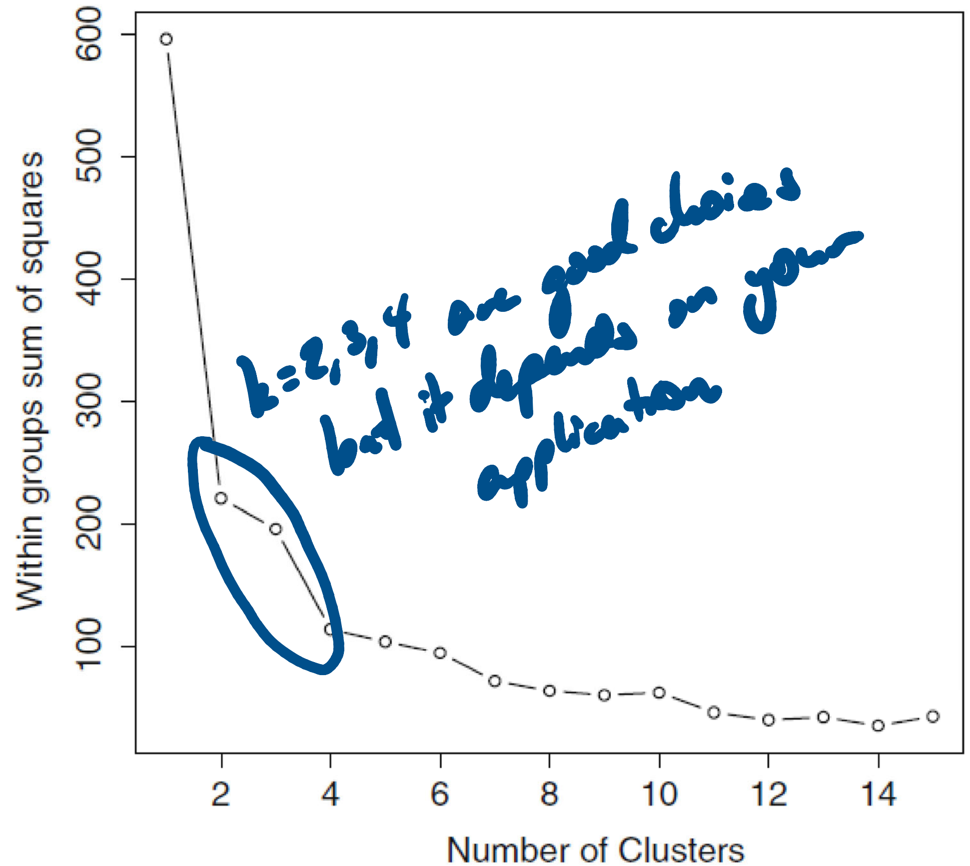
Choosing a value of k

- Given a k-means clustering of N data items $\mathbf{x}_i$ to k cluster centers $\mathbf{c}_j$, define the sum of square distances from each $\mathbf{x}_i$ to its cluster center as a cost function

$$\sum_{i=1}^{N} \sum_{j=1}^{k} \delta_{i,j} \left\| \mathbf{x}_i - \mathbf{c}_j \right\|^2 \quad \text{where} \quad \delta_{i,j} = \begin{cases} 1 & \text{if } \mathbf{x}_i \in \text{cluster } j \\ 0 & \text{if } \mathbf{x}_i \notin \text{cluster } j \end{cases}$$

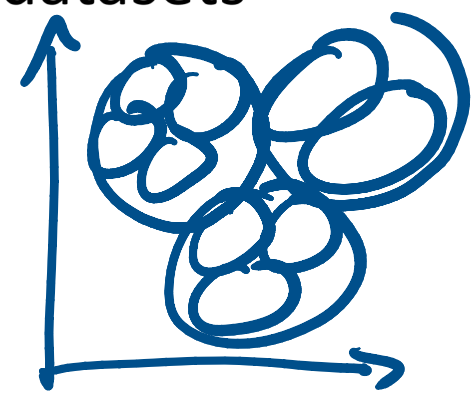
- Perform k-means clustering for many values of k and find the knee in the cost function curve

Choosing a value of k: iris example



Some variants of k-means clustering

- Soft assignment allows some data items to belong to multiple clusters with weights associated with each cluster
- Hierarchical k-means speeds up clustering for very large datasets
  - Sample the dataset and apply k-means with a small value of k
  - Assign all the data to one of the clusters
  - Subcluster each individual cluster
  - Repeat until you have a tree of clusters of your desired depth



- k-medioids allows clustering of data that cannot be averaged