
# Unsupervised Learning

Pierre Gaillard – ENS Paris

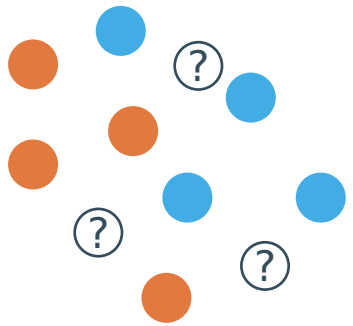
September 28, 2018

# Supervised vs unsupervised learning

**Two main categories** of machine learning algorithms:

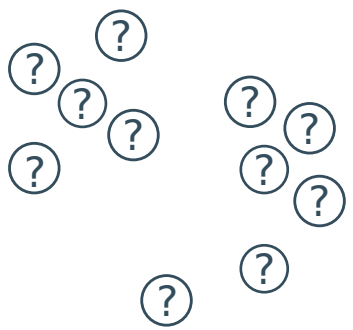
- **Supervised learning:** predict output $Y$ from some input data $X$. The training data has a known label $Y$.

  Examples:
  - $X$ is a picture, and $Y$ is a cat or a dog
  - $X$ is a picture, and $Y \ 2 \ f0; : : : ; 9g$ is a digit
  - $X$ is are videos captured by a robot playing table tennis, and $Y$ are the parameters of the robots to return the ball correctly



- **Unsupervised learning:** training data is not labeled and does not have a known result

  Examples:
  - detect change points in a non-stationary time-series
  - detect outliers
  - clustering: group data in homogeneous groups
  - principal component analysis: compress data without loosing much information
  - density estimation
  - dictionary learning



- **Others:** reinforcement learning, semi-supervised learning, online learning,...

# Reminder of last class on supervised learning: OLS and logistic regression

# Least square Linear regression

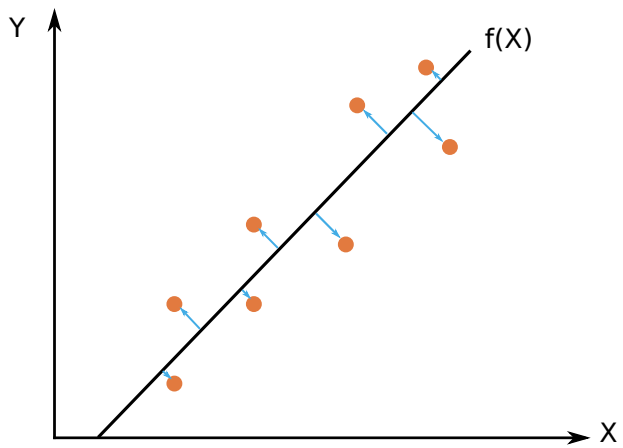
Given training data $(X_i; Y_i)$ for $i = 1; : : : ; n$, with $X_i \ 2 \ \mathrm{R}^d$ and $Y_i \ 2 \ \mathrm{R}$ learn a predictor $f$ such that our expected square loss

$$\mathrm{E} \ (f(X) \quad Y)^2$$

is small.

We assume here that $f$ is a linear combination of the input $x = (x_1; : : : ; x_d)$

$$f \ (x) = \sum_{i=1}^{\not{X}} {}_i x_i = \ ^{>} x$$



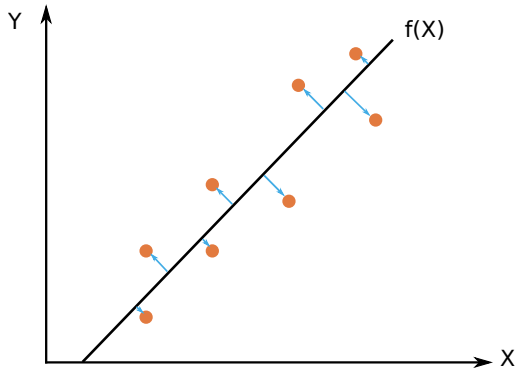
# Ordinary Least Square

Input $X \in \mathbb{R}^d$, output $Y \in \mathbb{R}$, and $\ell$ is the square loss: $\ell(a;y) = (a - y)^2$.

The Ordinary Least Square regression (OLS) minimizes the empirical risk

$$\hat{R}_n(w) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - w^\top X_i)^2$$

This is minimized in $w \in \mathbb{R}^d$ when $X^\top X w - X^\top Y = 0$; where $X = [X_1; \dots; X_n]^\top \in \mathbb{R}^{n \times d}$ and $Y = [Y_1; \dots; Y_n]^\top \in \mathbb{R}^n$.




Assuming $X$ is injective (i.e., $X^\top X$ is invertible) and there is an exact solution

$$\hat{w} = (X^\top X)^{-1} X^\top Y .$$

What happens if $d \gg n$?

# Ordinary Least Square: how to compute $\widehat{\theta}_n$?

If the design matrix $X^\top X$ is invertible, the OLS has the closed form:

$$\hat{\theta}_n \in \arg\min_{\theta} \hat{R}_n(\theta) = (X^\top X)^{-1} X^\top Y\,.$$

**Question:** how to compute it?

- inversion of $(X^\top X)$ can be prohibitive (the cost is $O(d^3)$!)
- $QR$-decomposition: we write $X = QR$, with $Q$ an orthogonal matrix and $R$ an upper-triangular matrix. One needs to solve the linear system:

$$R\hat{\theta} = Q^\top Y\,; \qquad \text{with} \qquad R = \begin{pmatrix} x & x & \cdots & x \\ & \ddots & \ddots & \vdots \\ & & \ddots & x \\ 0 & & & x \end{pmatrix}$$

- iterative approximation with convex optimization algorithms [Bottou, Curtis, and Nocedal 2016]: (stochastic)-gradient descent, Newton,...

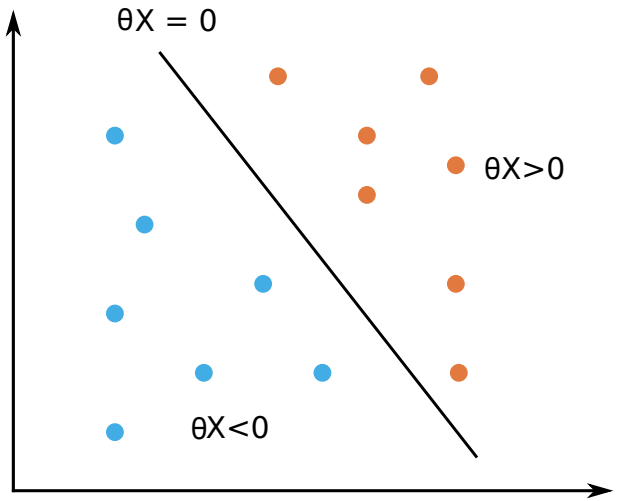
$$\theta_{i+1} = \theta_i - \gamma \nabla \hat{R}_n(\theta_i)$$

# Classification

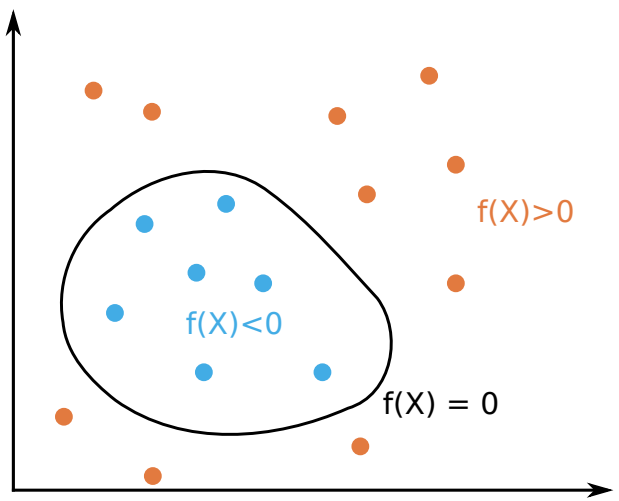
Given training data $(X_i; Y_i)$ for $i = 1; : : : ; n$, with $X_i \ 2 \ \mathrm{R}^d$ and $Y_i \ 2 \ f0; 1g$ learn a classifier $f(x)$ such that

$$f(X_i) \begin{cases} > 0 & ) \quad Y_i = +1 \\ < 0 & ) \quad Y_i = 0 \end{cases}$$

Linearly separable




**Non** Linearly separable



# Linear classification

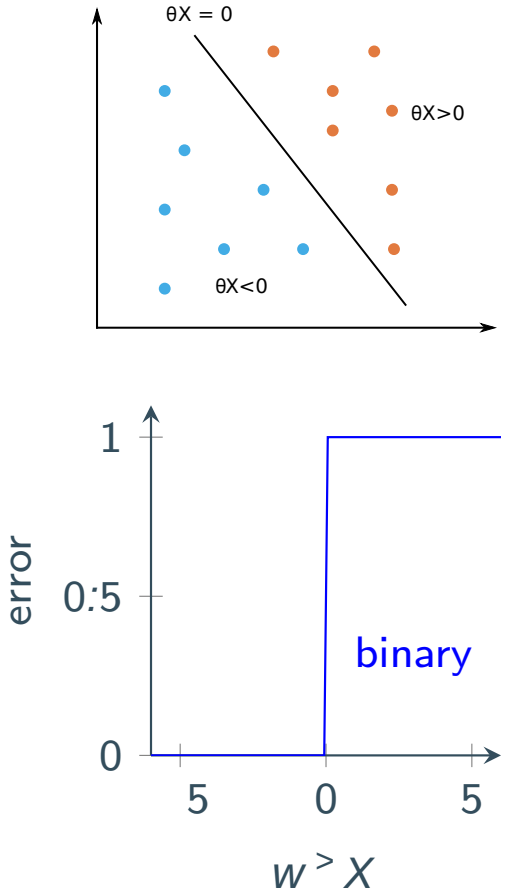
We would like to find the best linear classifier such that

$$f(X) = {}^{>}X \begin{cases} > 0 & \Rightarrow \quad Y = +1 \\ < 0 & \Rightarrow \quad Y = 0 \end{cases}$$

**Empirical risk minimization** with the binary loss?

$$\hat{b}_n = \arg\min_{\beta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{Y_i \neq \mathbb{1}_{\beta^\top X_i \geqslant 0}}.$$

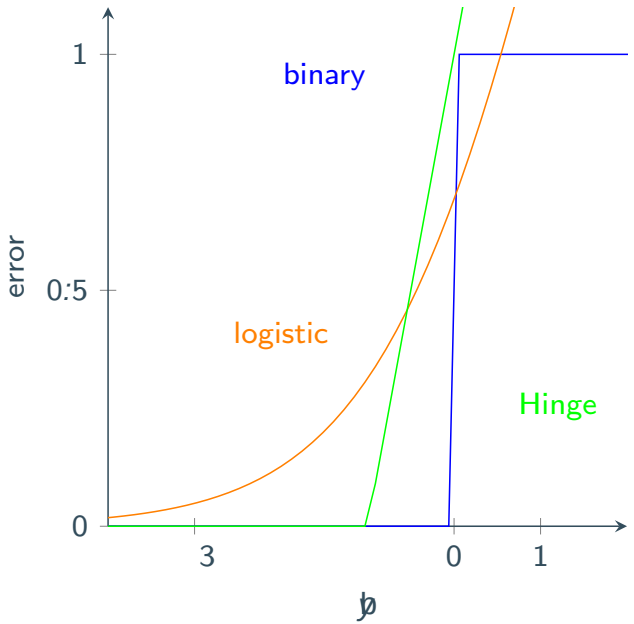
This is not convex in $\beta$. Very hard to compute!

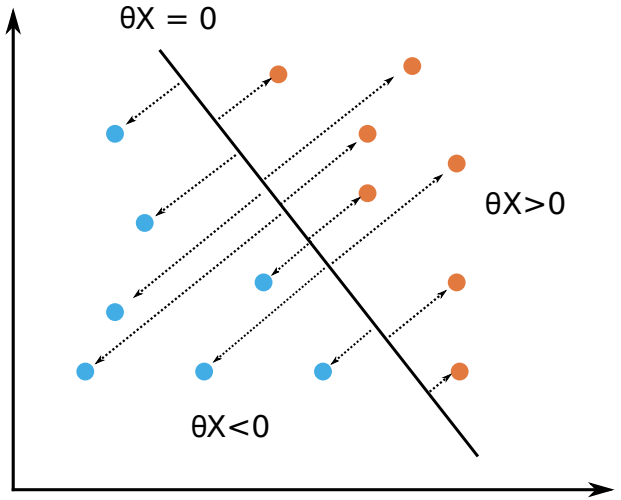


# Logistic regression

**Idea:** replace the loss with a convex loss

$$\ell(\theta^\top x, y) = y \log\left(1 + e^{-\theta^\top x}\right) + (1-y)\log\left(1 + e^{\theta^\top x}\right)$$




**Probabilistic interpretation:** based on likelihood maximization of the model:

$$P(Y = 1 | X) = \frac{1}{1 + e^{-\theta^\top x}} \quad \theta \in [0; 1]$$

Satisfied for many distributions of $X|Y$: Bernoulli, Gaussian, Exponential, ...

**Computation of the minimizer of the empirical risk** (No closed form of the solution)

- Use a convex optimization algorithm (Newton, gradient descent,...)

# Clustering

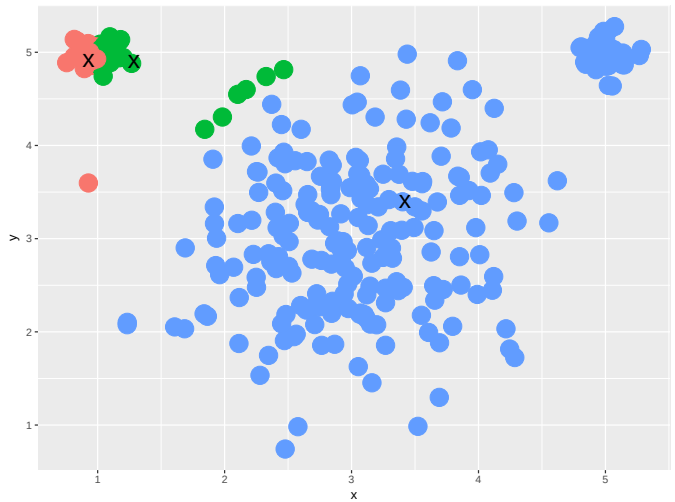
# Outline

# What is clustering?

**Goal:** group together similar instances

Requires data but no labels

Useful when you don't know what you are looking for




**Types of clustering algorithms**:

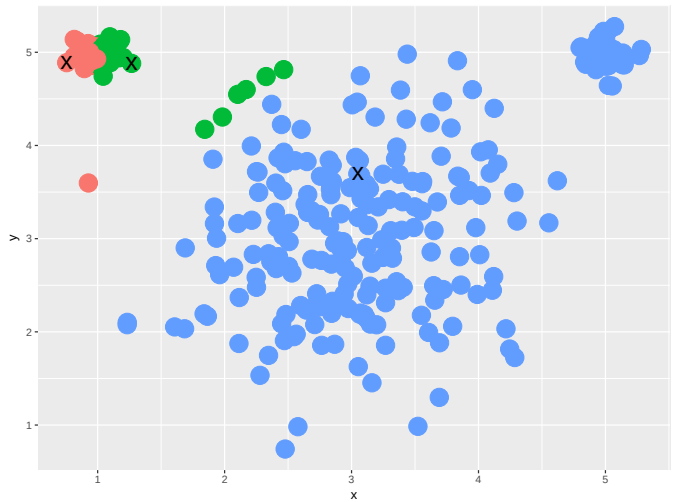
- model based clustering: mixture of Gaussian
- hierarchical clustering: a hierarchy of nested clusters is build using divisive or agglomerative approach
- Flat clustering: no hierarchy: k-means, spectral clustering

# What is clustering?

**Goal:** group together similar instances

Requires data but no labels

Useful when you don't know what you are looking for




**Types of clustering algorithms**:

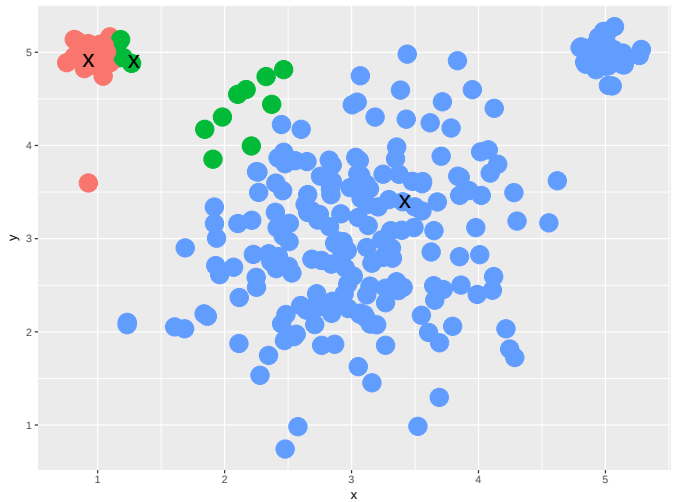
- model based clustering: mixture of Gaussian
- hierarchical clustering: a hierarchy of nested clusters is build using divisive or agglomerative approach
- Flat clustering: no hierarchy: k-means, spectral clustering

# What is clustering?

**Goal:** group together similar instances

Requires data but no labels

Useful when you don't know what you are looking for




**Types of clustering algorithms**:

- model based clustering: mixture of Gaussian
- hierarchical clustering: a hierarchy of nested clusters is build using divisive or agglomerative approach
- Flat clustering: no hierarchy: k-means, spectral clustering

# Examples of applications

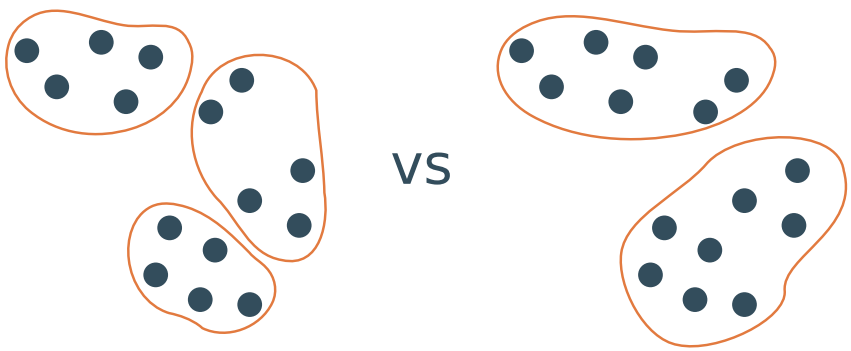
Clustering has a variety of goals which all relates to grouping or segmenting a collection of objects into homogeneous objects.

- **Biology**: group species and generate phylogenies
- **Marketing**: group similar shopping items and similar customers *!* market outreach, recommender systems
- **Computer science:** image segmentation

# What do we need for clustering?

Clustering does not need any labeled data. But:

1. **A proximity measure**: it can be
   - similarity measure between items $s(X_i; X_j)$ which is large if $X_i$ and $X_j$ are similar. Example: correlation
   - dissimilarity measure $d(X_i; X_j)$ which is large if $X_i$ and $X_j$ are different. Example: distance from a metric $d(X_i; X_j) = kX_i \quad X_j k$
2. **A criterion to evaluate the clustering**:




3. **A clustering algorithm** optimizing the criterion

# Proximity measure

**Proximity matrix**: the data may be represented directly in terms of the proximity between pairs of objects.
This is the case for categorical variables.
Example: experiment where participants are asked to judge how much objects differ from one another.

The proximity matrix is assumed to be symmetric: $D \;\; ! \;\; (D + D^T)$=2

**Distance measure**: the similarity can be measured by a metric

- Euclidean distance: $d(X_i; X_j) = kX_i \quad X_j k_2$
- Correlation
- Manhattan distance: $d(X_i; X_j) = kX_i \quad X_j k_1$
- Minkowvski distance: $d(X_i; X_j) = kX_i \quad X_j k_p$

The results crucially depends on the metric choice: depends on data.

# Clustering performance evaluation

This is often a hard problem. Much harder than for supervised learning!

A trade-off **Intra-cluster homogeneity**: objects within the same cluster should be

similar from one another

**Inter-clusters heterogeneity**: clusters should be well-separeted, their centers should be far from one another

In many application, we ask experts

# Examples of performance criterion

**Known ground truth**: mutual information: if $U; V : f1; : : : ; ng \;\; ! \;\; P(f1; : : : ; ng)$ are two clustering of $n$ objects, the mutual information measures their agreement as:

$$MI(U; V) = \sum_{i=1}^{} \sum_{j=1}^{} P(i; j) \log \frac{P(i; j)}{P(i)P^0(j)}$$

where $P(i) = jU_i j=n$ (resp. $P^0(j) = jV_j j=n$ and $P(i; j) = jV_j j \setminus jU_i j=n$) is the probability that an object picked at random falls into class $U_i$ (resp. $V_j$ and into both classes). Drawback: the ground truth is rarely known

**Unknown ground truth**: Silhouette score defined as

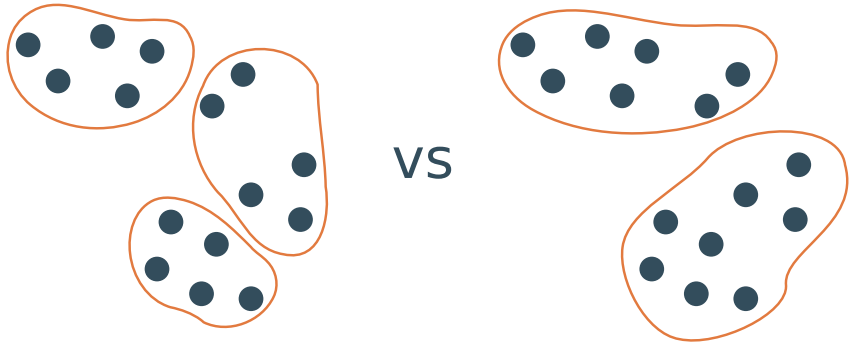
$$Score = \frac{b \quad a}{\max fa; bg}$$

where

- $a$: the mean distance between a sample and all other points in the same class.
- $b$: the mean distance between a sample and all other points in the next nearest cluster.

The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

# How many clusters?




Possibles approaches:

- Fix the number of clusters $k$
- Choose the number of clusters by optimizing the performance criterion such as the silhouette score

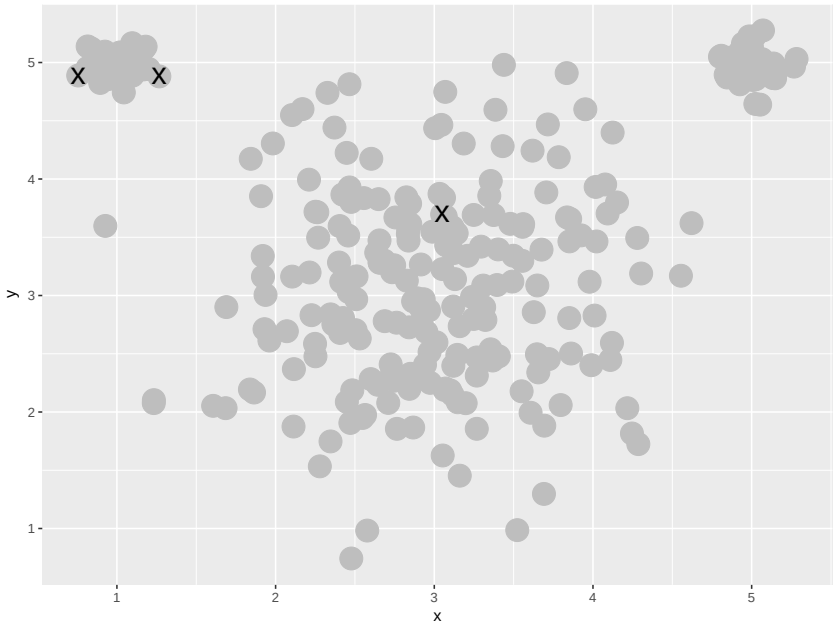
# Outline

# Clustering algorithms

**Types of clustering algorithms**:

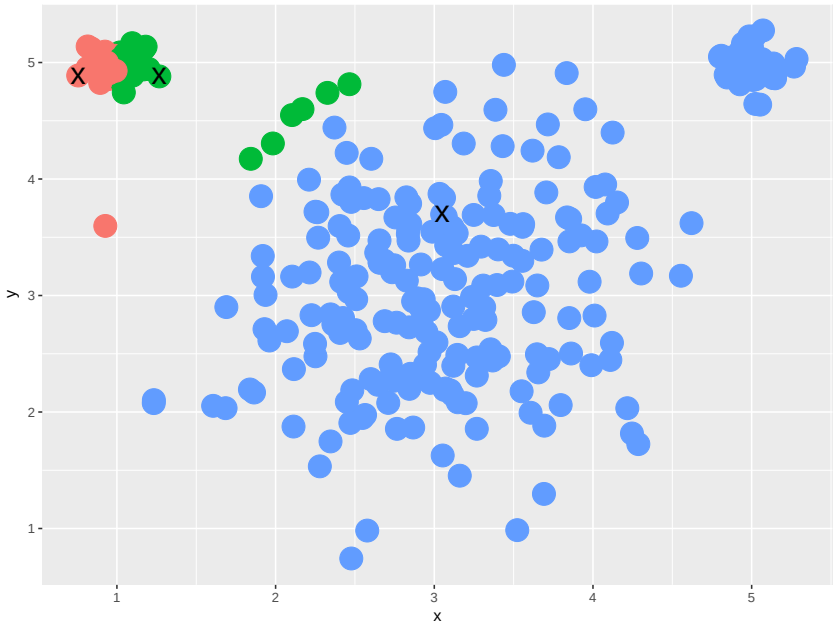
- Kmeans: flat clustering
- Spectral clustering
- Combinatorial clustering
- Gaussian mixture: model based clustering
- Hierarchical clustering: a hierarchy of nested clusters is build using divisive or agglomerative approach
- Affinity propagation: based on message passing in a graph

# K-means



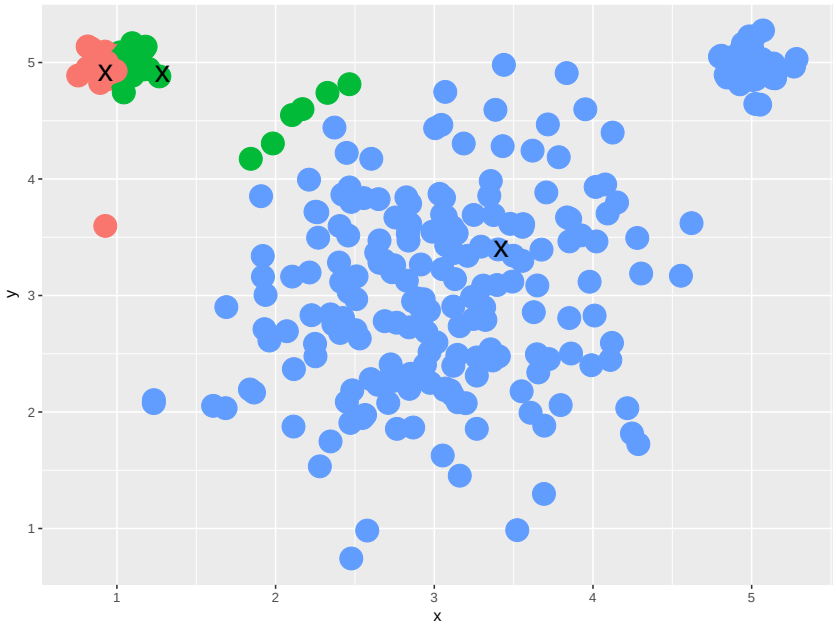

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



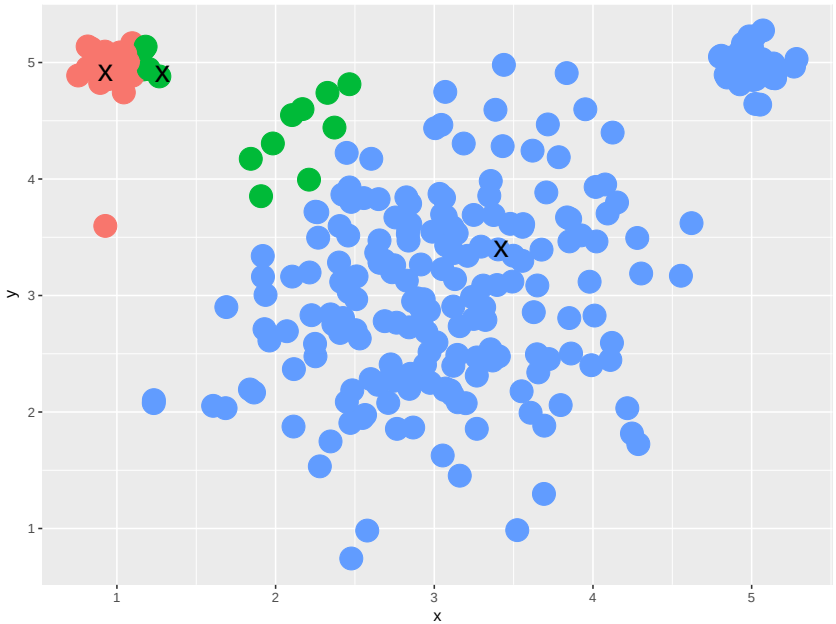

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



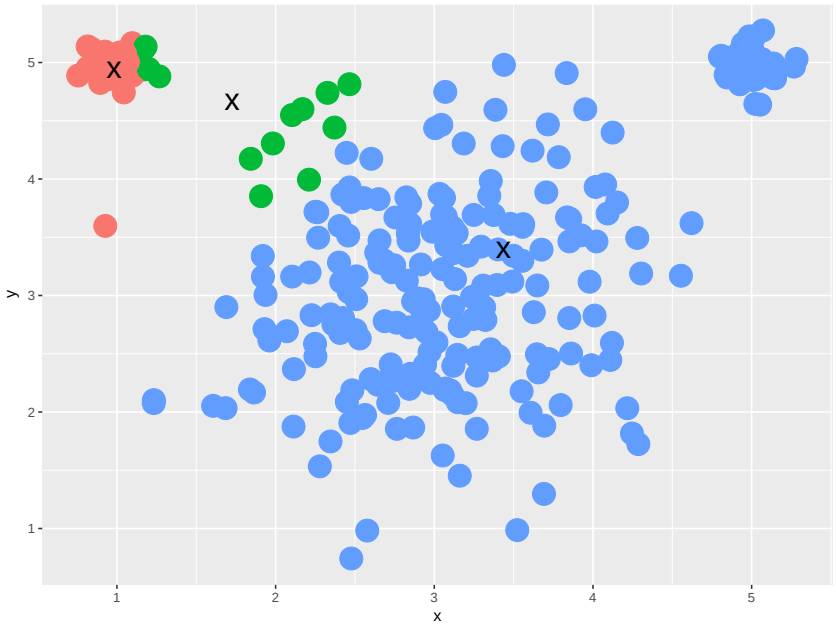

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



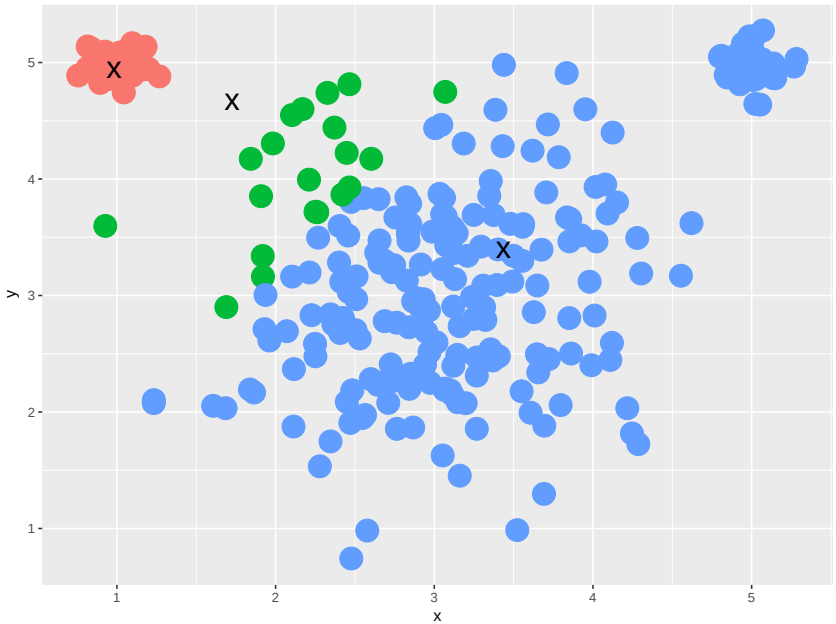

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



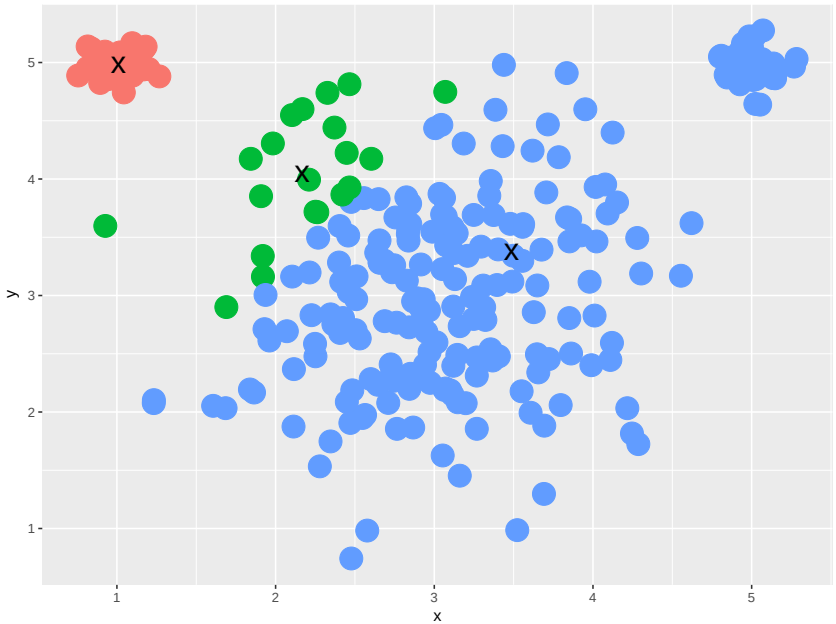

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



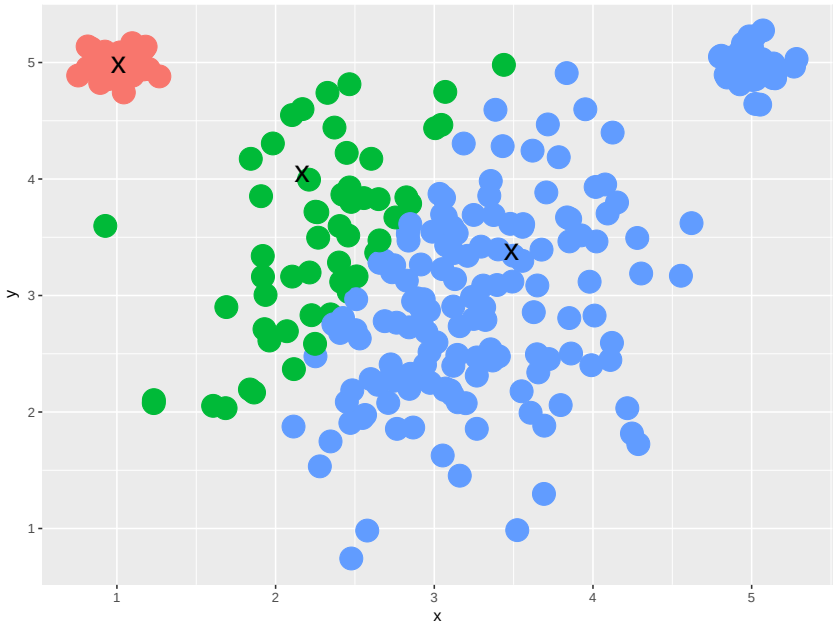

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



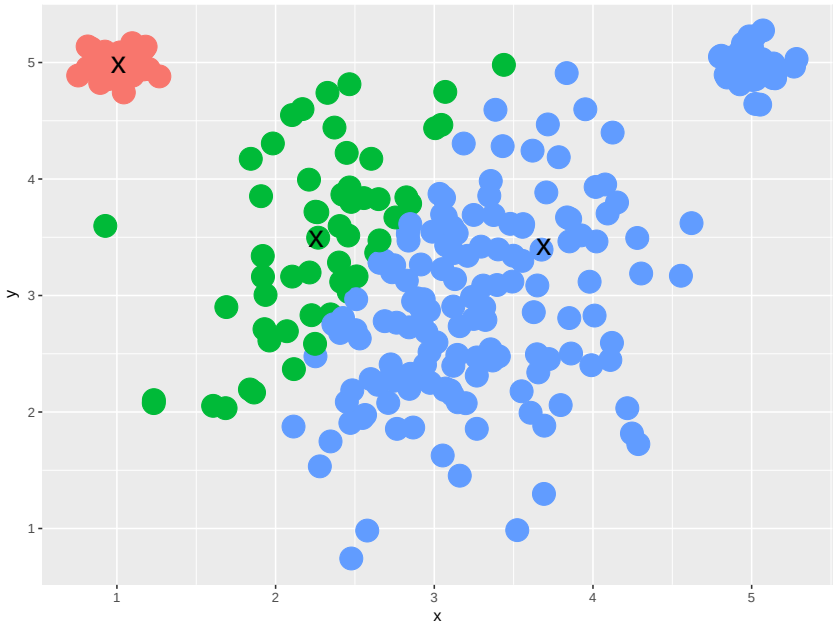

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



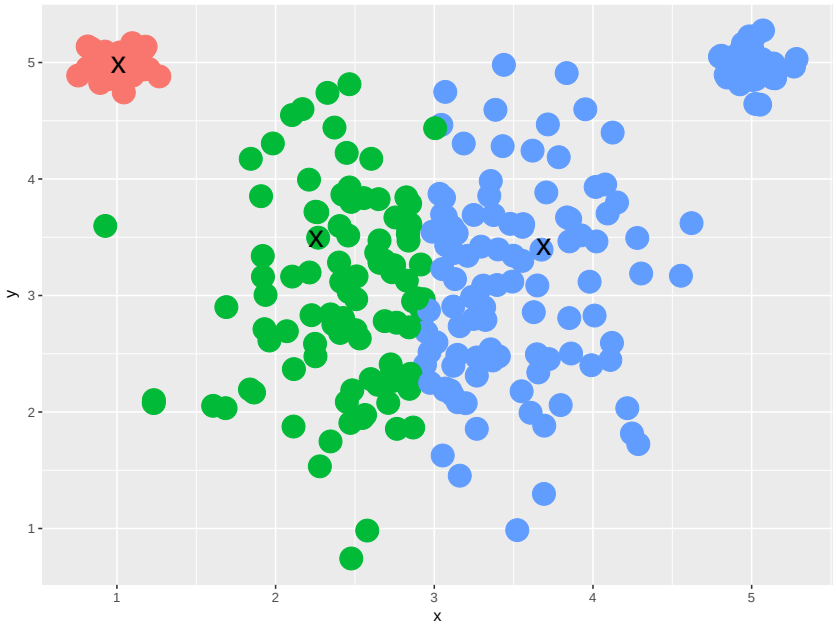

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



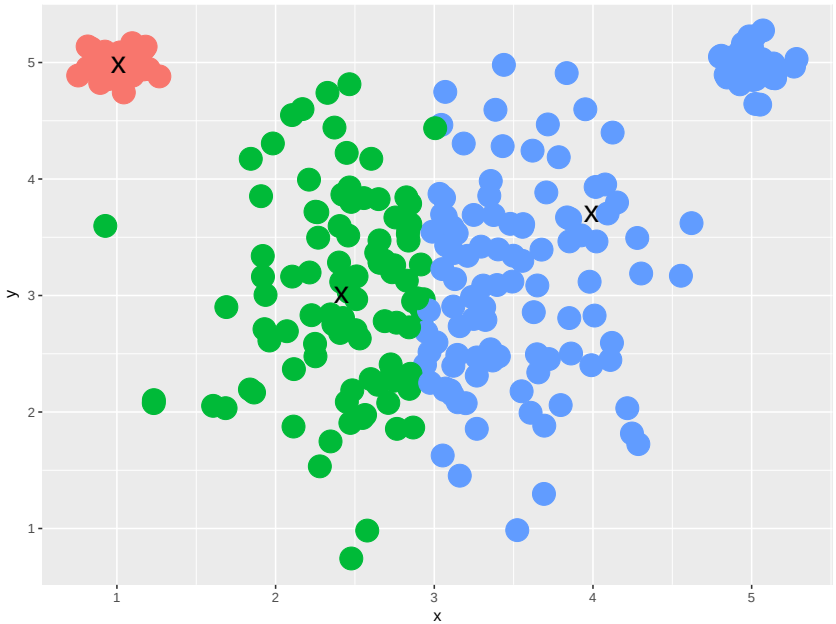

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



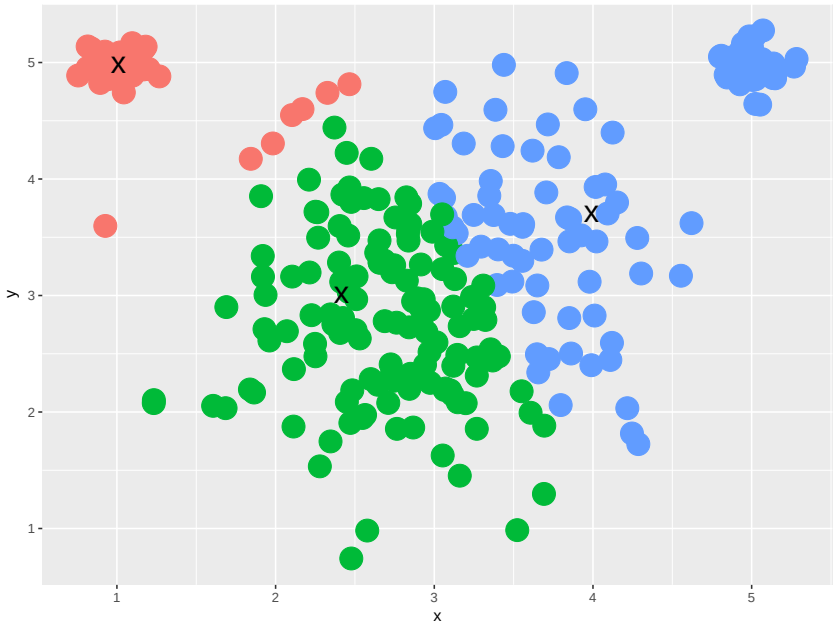

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



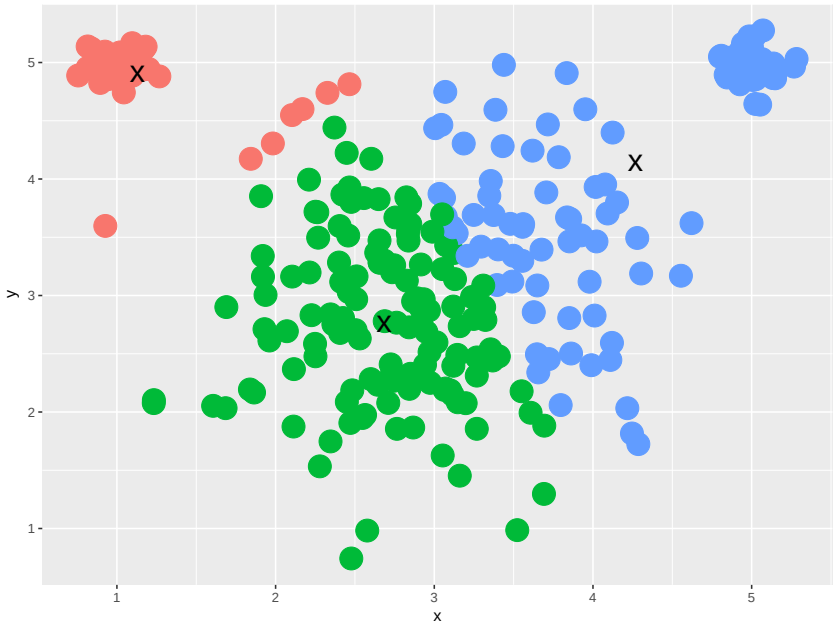

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



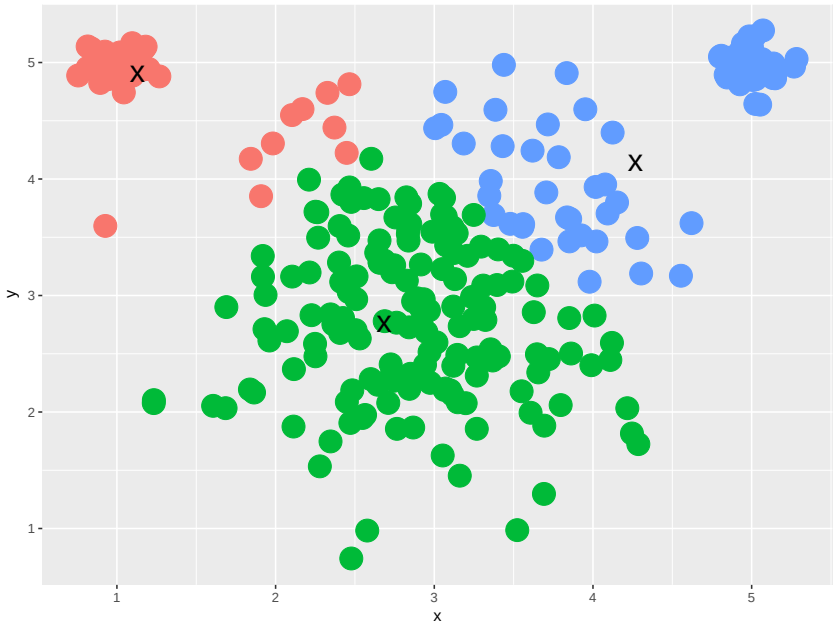

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



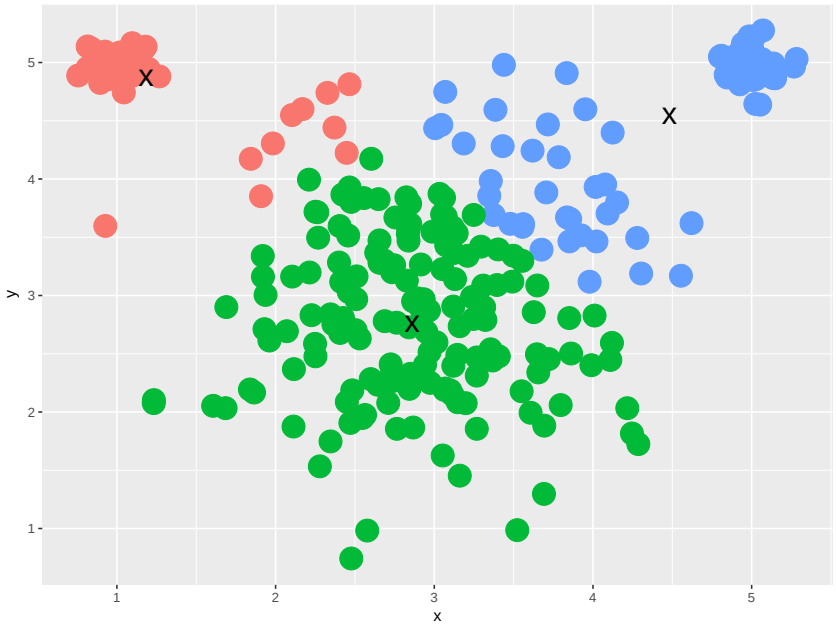

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



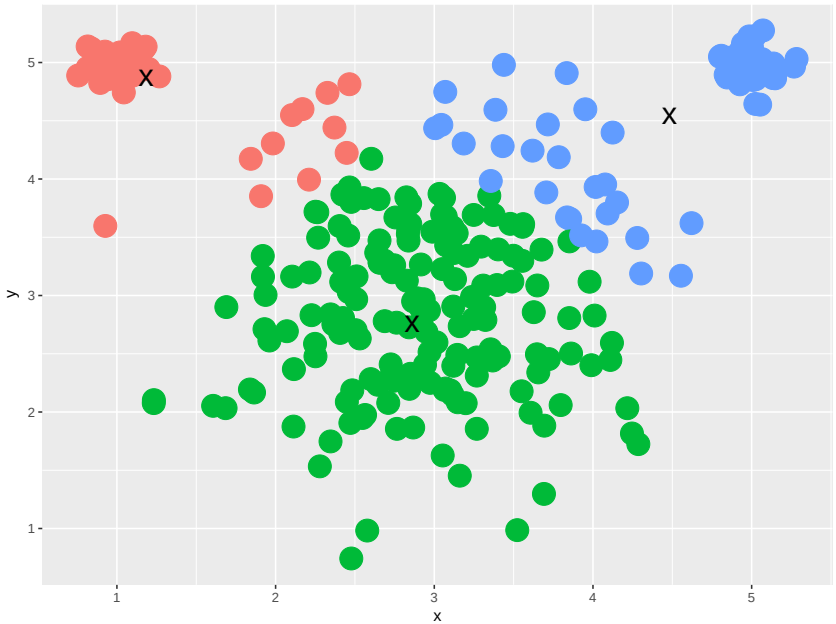

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



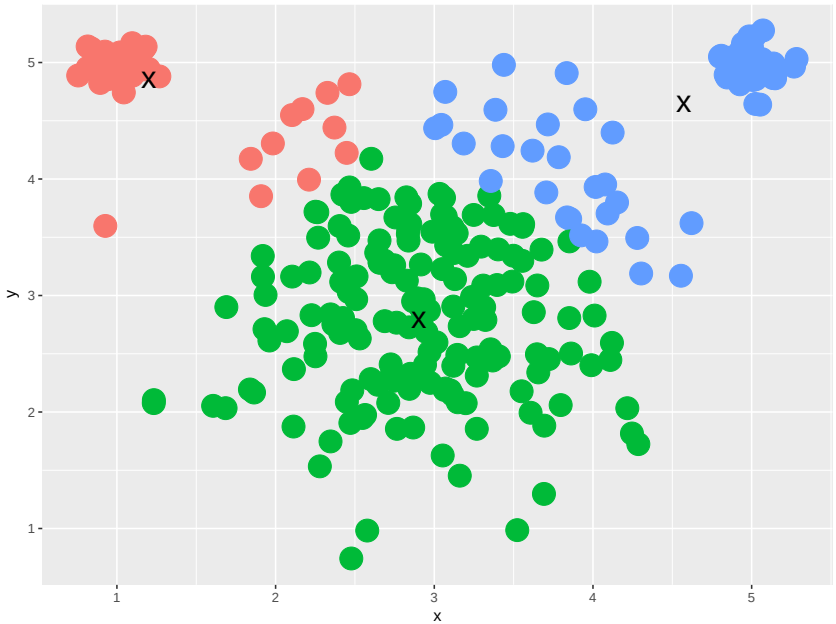

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



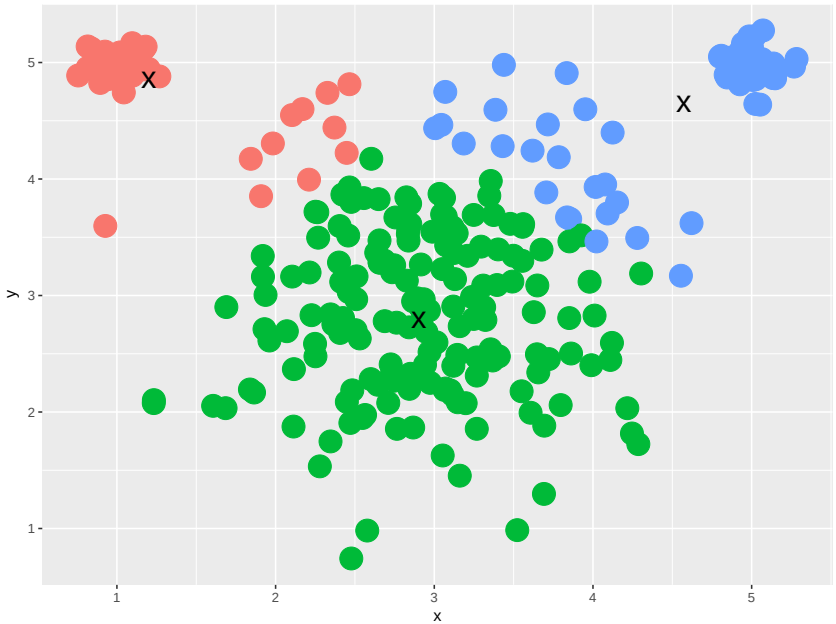

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



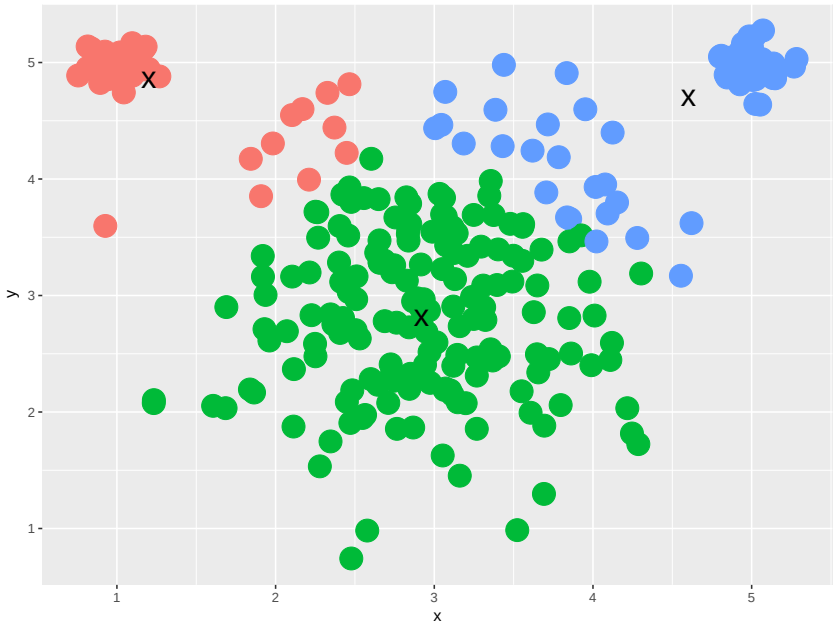

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means



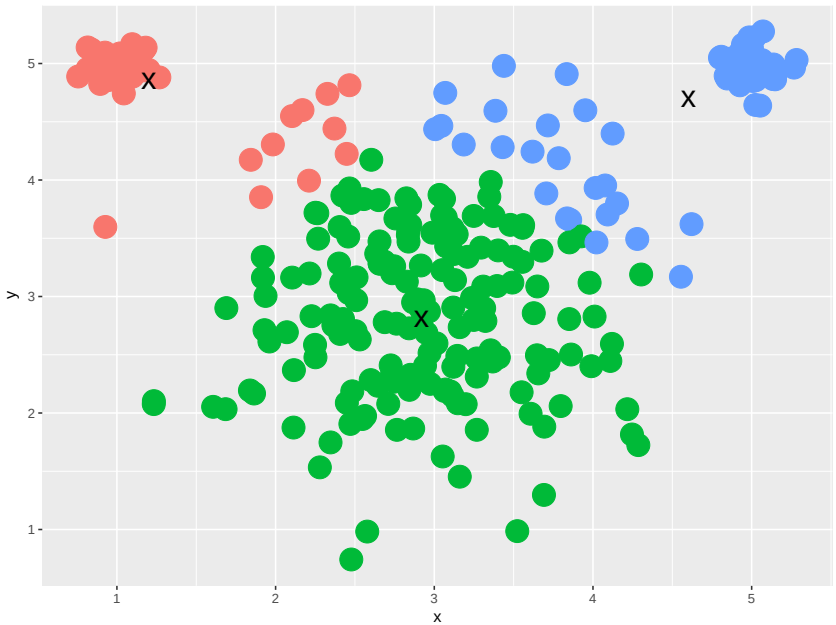

- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means




- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

# K-means




- **Initialization**: sample $K$ points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

Guaranteed to converge in a finite number of iterations.

Initialization is crucial.

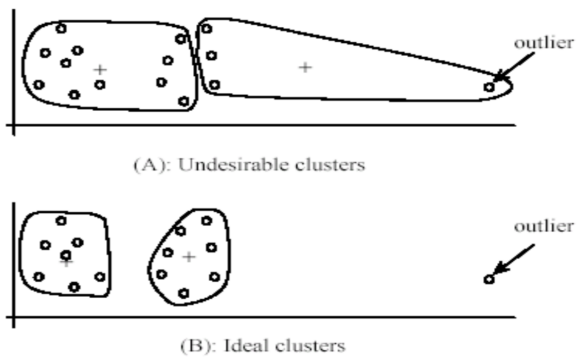
# Kmeans: Pros and cons

**Pros**

- simple easy to implement
- efficient: guaranteed to converge in finite number of iteration. Complexity: $O(tkn)$, $k$: number of clusters (small), $n$: number of data, $t$ number of iteration (small)
- popular

**Cons**

- the means need to be defined
- the number of cluster $k$ need to be specified
- sensitivity to outliers
  - *!* perform subsampling or outlier detection
- sensitivity to initial seeds: often get stuck in local minima
  - *!* initialize with different random seeds
- not suited for special data structure
- fails if clusters are not round



# Kmeans: Pros and cons

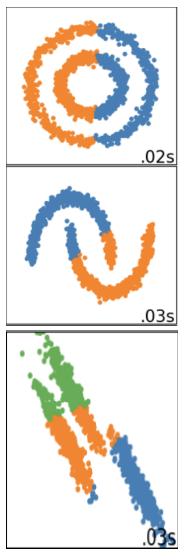
**Pros**

- simple easy to implement
- efficient: guaranteed to converge in finite number of iteration. Complexity: $O(tkn)$, $k$: number of clusters (small), $n$: number of data, $t$ number of iteration (small)
- popular

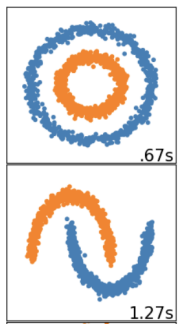
**Cons**

- the means need to be defined
- the number of cluster $k$ need to be specified
- sensitivity to outliers
  - *!* perform subsampling or outlier detection
- sensitivity to initial seeds: often get stuck in local minima
  - *!* initialize with different random seeds
- not suited for special data structure
- fails if clusters are not round

Kmeans




Spectral clustering




Changing the feature (or distance) may help

# Gaussian mixture

**What is a Gaussian mixture?**

- Parameter $\quad = (w; \quad ; \Sigma)$
- There are $K$ groups.
- Group $k$ is associated a mean $\quad_k$ and a variance $\Sigma_k$
- The data of each group is generated from a Gaussian $N(\quad_k; \Sigma_k)$
- A data point $X^{(i)}$ is generated as follows:
  - randomly sample the group $Z^{(i)} \; 2 \; f0; 1g^K$ with $\sum_k Z_k^{(i)} = 1$ the group $P(Z_k^{(i)} = 1) = w_k$
  - If $Z_k^{(i)} = 1$, sample $X^{(i)} \quad N(\quad_k; \Sigma_k)$
- The points $X^{(i)}$ are revealed but the groups $Z^{(i)}$ are hidden

The density of $X^{(i)}$ is

$$f(x; \quad) = \sum_{k=1}^{K} w_k N(x; \quad_k; \Sigma_k)$$

The density of $X_i; Z_i$ is

$$f(x; z; \quad) = \prod_{k=1}^{K} w_k N(x; \quad_k; \Sigma_k)^{z_k}$$

# Gaussian Mixture – EM algorithm

We want to maximize the log-likelihood of $f(x;\ ) = \sum_{k=1}^{K} w_k N(x;\ _k; \Sigma_k)$ but the expression of the log-likelihood $L(X;\ )$ is complicated.

Instead, we consider the complete density of $X^{(i)}; Z^{(i)}$

$$f(x; z;\ ) = \prod_{k=1}^{K} w_k N(x;\ _k; \Sigma_k)^{z_k}$$

whose log-likelihood is

$$L(X; Z;\ ) = \log \prod_{i=1}^{n} f(X^{(i)}; Z^{(i)}; w;\ ; \Sigma) = \sum_{i=1}^{n}\sum_{k=1}^{K} Z_k^{(i)} \log N(X^{(i)};\ _k; \Sigma_k) + \sum_{i=1}^{n}\sum_{k=1}^{K} Z_k^{(i)} \log w_k$$

**How to maximize $L$?**

1. If we knew the $Z^{(i)}$ we could maximize $L$ in $\ = (w;\ ; \Sigma)$
2. If we knew $\ $ we could choose the best $Z^{(i)}$ maximizing their probability

$$P\left(Z_k^{(i)} = 1 | X;\ \right) = \frac{w_k N(X^{(i)};\ _k; \Sigma_k)}{\sum_{j=1}^{K} w_j N(X^{(i)};\ _k; \Sigma_k)}$$

The expectation-maximization algorithm alternates between those two.

# Gaussian Mixture – EM algorithm

We want to maximize the log-likelihood of $f(x; \ ) = \sum_{k=1}^{K} w_k N(x; \ _k; \Sigma_k)$ but the expression of the log-likelihood $L(X; \ )$ is complicated.

Instead, we consider the complete density of $X^{(i)}; Z^{(i)}$

$$f(x; z; \ ) = \prod_{k=1}^{K} w_k N(x; \ _k; \Sigma_k)^{z_k}$$

whose log-likelihood is

$$L(X; Z; \ ) = \log \prod_{i=1}^{n} f(X^{(i)}; Z^{(i)}; w; \ ; \Sigma) = \sum_{i=1}^{n} \sum_{k=1}^{K} Z_k^{(i)} \log N(X^{(i)}; \ _k; \Sigma_k) + \sum_{i=1}^{n} \sum_{k=1}^{K} Z_k^{(i)} \log w_k$$

**Expectation-Maximization algorithm**

Initialize: $= \ _0$

Until convergence do

- Expectation step: $q(Z) = \mathrm{P}\, fZjX; \ g$
- Maximization step:

$$_t = \arg\max \mathrm{E}_{Z \ q} \ L(X; Z; \ ) \ :$$

There are closed form formulas!

# Hierarchical clustering

Produce a hierarchical representation in which the clusters at each level of the hierarchy are created by

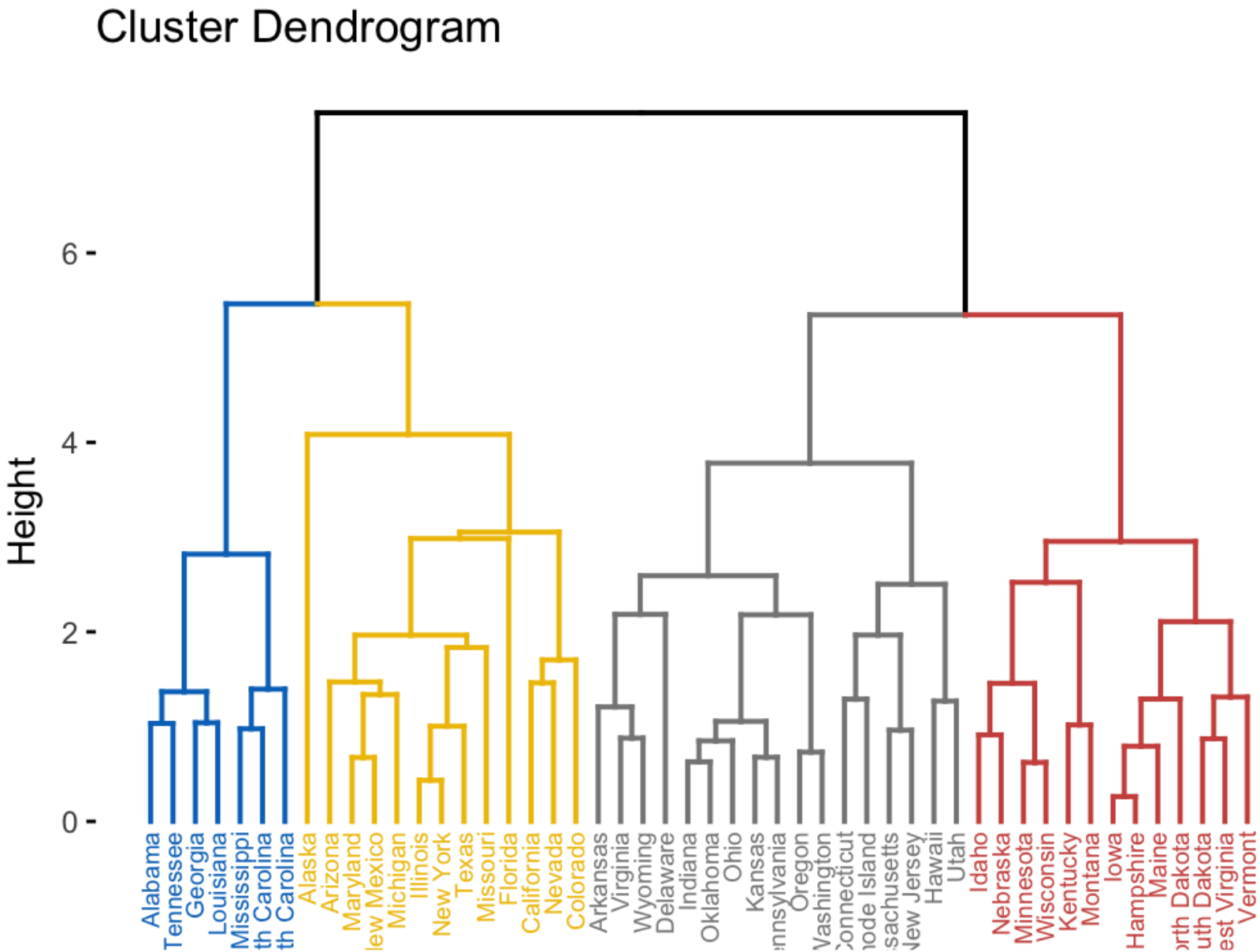
- merging clusters at the lower level: **agglomerative** (bottom-up) strategy
  *!* we group the pair of clusters with the smallest intergroup dissimilarity
- splitting clusters at the higher level: **divisive** (top-down) strategy
  *!* we split the cluster to produce two groups with the largest intergroup dissimilarity

Does not require the number $k$ of clusters to be pre-specified.

Each level of the hierarchy represents a particular clustering of the data

For some data (like biological) this is better suited than flat clustering with a hierarchy of groups: animal, mammal,...

# Hierarchical clustering dendogram



# Divisive clustering

Initialize with a single cluster
Divide each group recursively into two childs using for example kmeans with $k = 2$.
Stop when each cluster has only a single element.

# Agglomerative clustering

**Algorithm**: Initialize with $n$ cluster each containg a single data point.
While there is more than one cluster:

- find the two nearest clusters
- merge them

**How to measure the distance between clusters?** Four common ways

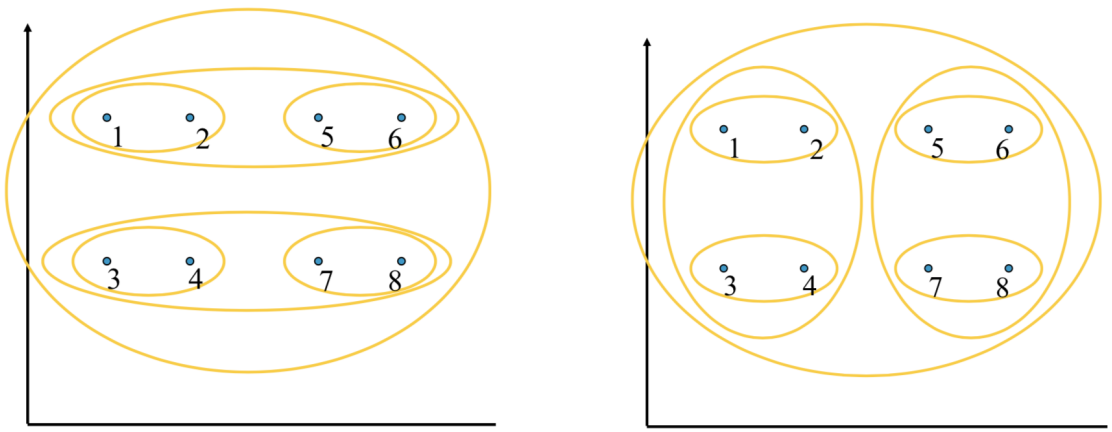
- minimum distance: $d_{\min}(U; V) = \min_{u2U;v2V} ku \quad vk$
- maximum distance: $d_{\max}(U; V) = \max u\ 2\ U; v\ 2\ V ku \quad vk$
- average distance
- distance between means

Different choices create different clustering behaviors!

# Agglomerative clustering

**Algorithm**: Initialize with $n$ cluster each containg a single data point.
While there is more than one cluster:

- find the two nearest clusters
- merge them




[Pictures from Thorsten Joachims]

# Divisive vs Agglomerative

Agglomerative is faster (no kmeans computation at each iteration)

Divisive is more global: first step find the best split among the all data, while agglomerative only look at pairwise comparaison.

# Dimensionality Reduction Algorithms

# Principal component analysis

Assume that you have centered observations $x_1; : : : ; x_n \ 2 \ \mathrm{R}^p$ represented as a data matrix (with column-wise zero empirical mean $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i = 0$)

$$X := [x_1; : : : ; x_n]^{>} \qquad \text{of dimension } n \quad p :$$

If $p$ is large, some columns (i.e., explanatory variables) may be linearly correlated.

**bad statistical property**: risk minimization not identifiable, the covariance matrix $(X^{>} X)$ is not invertible

$!$ unstable estimators

**bad computational property**: we need to store $p \quad 1$ columns with redundant information

Is it possible to reduce the dimension of the data from $\mathrm{R}^p$ to $\mathrm{R}^q$ through linear transformations without loosing much information?

The principal components of a set of data in $\mathrm{R}^p$ provide a sequence of best linear approximations of that data.

# Principal component analysis

Assume that you have centered observations $x_1; : : : ; x_n \ 2 \ \mathrm{R}^p$ represented as a data matrix (with column-wise zero empirical mean $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i = 0$)

$$X := [x_1; : : : ; x_n]^{>} \qquad \text{of dimension } n \quad p :$$

**Goal**: find $q$ good directions $u_k$ that preserve important aspects of the data

- two equivalent ways to define what are good directions
  - find directions of maximum variation
  - find projections that minimize the reconstruction error
- the directions $u_1; : : : ; u_q$ turn out to be the top $k$-eigenvalues of the covariance matrix $X^{>} X$

# Principal Component Analysis (Maximum Variance)

1. Find the vector $u_1$ such that the projection of the data on $u$ has the greatest variance.

$$u_1 \; 2 \arg\max_{kuk=1} \frac{1}{n} \sum_{i=1}^{n} (u_1^{>} x_i \quad u_1^{>} \bar{x}_i)^2$$

$$= \arg\max_{kuk=1} kX^{>} uk^2 = u^{>} X^{>} Xu$$

   ) this is the principal eigenvector of $X^{>} X$.
2. More generally, if we wish a $k$-dimensional subspace we choose $u_1; : : : ; u_k$ the top $k$ eigenvectors of $X^{>} X$.
3. The $u_i$ form a new orthogonal basis of the data

# Principal component analysis (Minimum reconstruction Error)

We can also think of PCA as minimizing the reconstruction error of the compressed data.

Consider the linear transformation from $\mathrm{R}^q$ to $\mathrm{R}^p$ for $q \leq p$:

$$f : z \in \mathrm{R}^q \mapsto \quad + U_q z$$

where $\in \mathrm{R}^p$ is a location vector and $U_q$ is a $p \quad q$ matrix with $q$ orthogonal unit vectors.

The goal of PCA is to fit such a linear model to the data by minimizing the reconstruction error

$$\min_{;(z_i);U_q} \sum_{i=1} x_i \quad U_q z_i \ ^2$$

The solution turns out to be the same as maximal variation.

# How many components should we choose?

If we use principal components as a summary of our data, how many components are sufficient?

- No simple answer to this question. Cross validation as we would use for supervised learning is not possible.
- We may choose by looking visually at the reconstruction error or the percentage of variance explained.

# PCA vs Clustering

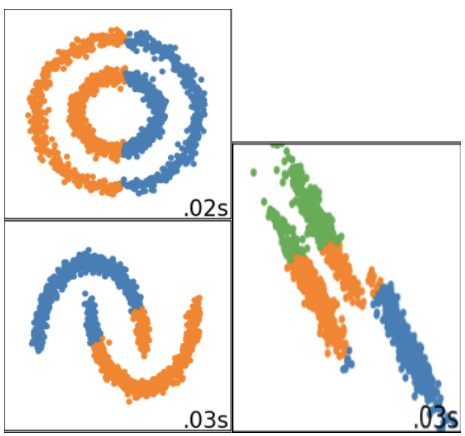
PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance.

Clustering looks for homogeneous subgroups among the observations.

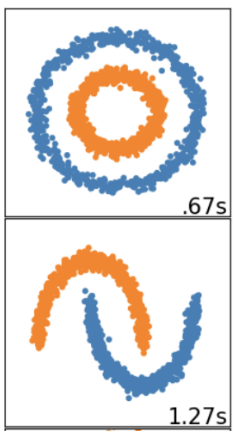
They can be combined in methods such as Spectral Clustering:

1. apply PCA like techniques to find a better representation of the data
2. apply Kmeans

Kmeans




Spectral clustering



# Conclusion

Unsupervised learning is important for understanding the variation and grouping structure of a set of unlabeled data, and can be a useful pre-processor for supervised learning.

It is intrinsically more difficult than supervised learning because there is no gold standard (like an outcome variable) and no single objective (like test set accuracy).

It is an active field of research, with many recently developed tools such as self-organizing maps, independent components analysis and spectral clustering.

# References


L. Bottou, F. E. Curtis, and J. Nocedal. "Optimization methods for large-scale machine learning". In: *arXiv preprint arXiv:1606.04838* (2016).

N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

T. J. Hastie, R. J. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

Wikipedia. The free encyclopedia.