Unsupervised Learning

Pierre Gaillard – Intelligent Systems (MOSIG)

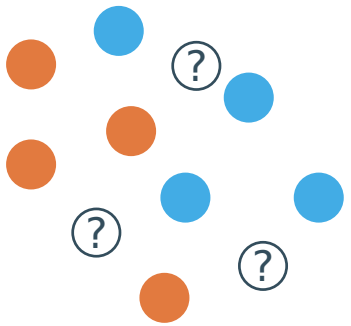
2022-2023

Supervised vs unsupervised learning

Two main categories of machine learning algorithms:

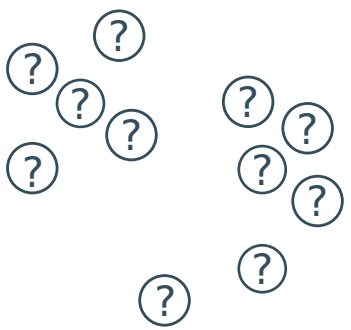
- **Supervised learning:** predict output Y from some input data X. The training data has a known label Y.

  Examples:
  - X is a picture, and Y is a cat or a dog
  - X is a picture, and $Y \in \{0, \ldots, 9\}$ 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,...

Clustering

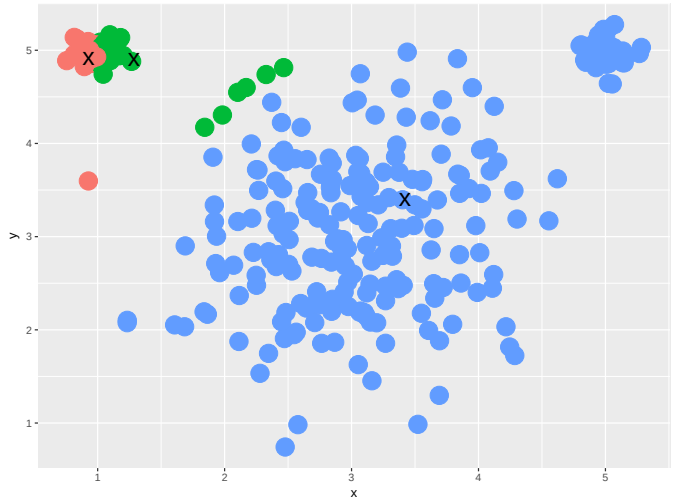
Outline

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:

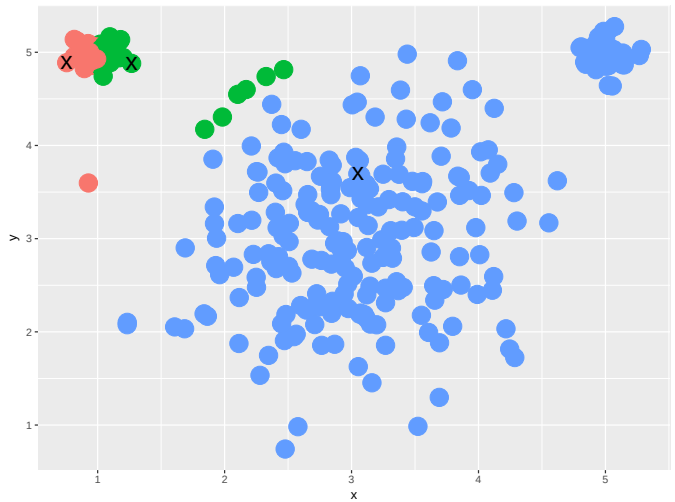
- model based clustering: mixture of Gaussian
- hierarchical clustering: a hierarchy of nested clusters is build using divisive or agglomerative approach
- Flat clustering: no hierarchy: k-means, spectral clustering

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:

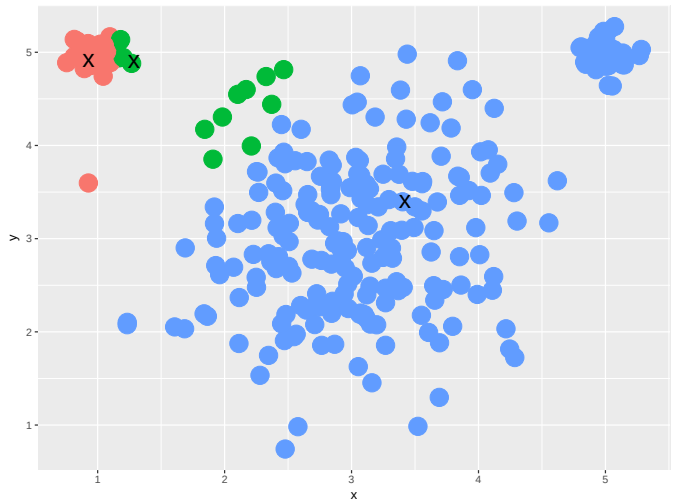
- model based clustering: mixture of Gaussian
- hierarchical clustering: a hierarchy of nested clusters is build using divisive or agglomerative approach
- Flat clustering: no hierarchy: k-means, spectral clustering

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

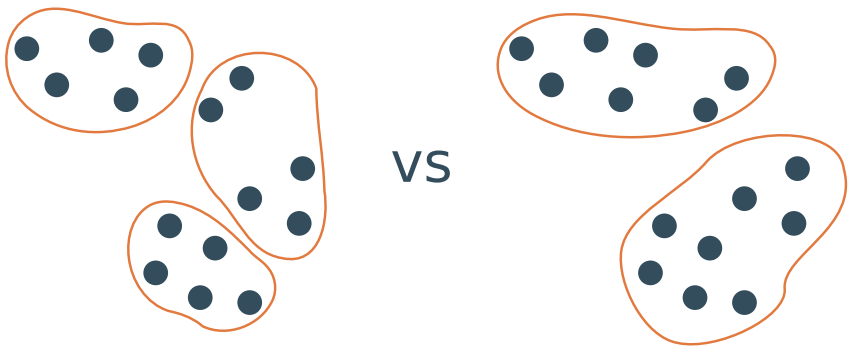
Clustering has a variety of goals which all relates to grouping or segmenting a collection of objects into homogeneous objects.

- **Biology**: group species and generate phylogenies
- **Marketing**: group similar shopping items and similar customers $\rightarrow$ 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 $d(X_i, X_j) = \|X_i - X_j\|$
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: $D \to (D + D^T)/2$

Distance measure: the similarity can be measured by a metric

- Euclidean distance: $d(X_i, X_j) = \|X_i - X_j\|_2$
- Correlation
- Manhattan distance: $d(X_i, X_j) = \|X_i - X_j\|_1$
- Minkowvski distance: $d(X_i, X_j) = \|X_i - X_j\|_p$

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 (MI) If $U, V : \{1, \ldots, n\} \to \mathcal{P}(\{1, \ldots, n\})$ are two clustering of n objects, MI measures their agreement by quantifying the reduction in entropy achieved by observing one clustering given the values of the other:

$$MI(U, V) = \sum_{i=1}^{n} \sum_{j=1}^{n} P(i, j) \log \left(\frac{P(i, j)}{P(i)P'(j)} \right)$$

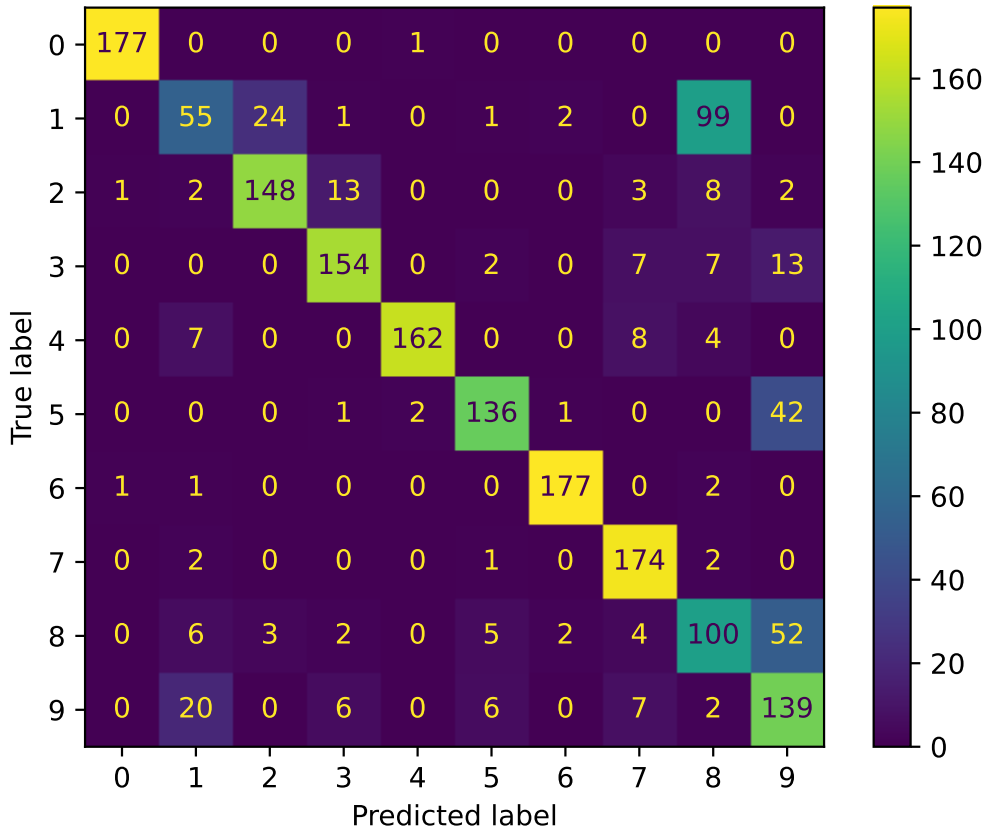
where $P(i) = |U_i|/n$ (resp. $P'(j) = |V_j|/n$ and $P(i, j) = |V_j| \cap |U_i|/n$) is the probability that an object picked at random falls into class U_i (resp. V_j and into both classes).

Adjusted mutual information (AMI) corrects the effect of agreement solely due to chance between clusterings. It is 1 for perfect assignment and 0 for random chance.

Drawback: the ground truth is rarely known

Examples of performance criterion: Known ground truth

Confusion Matrix



Drawback: the ground truth is rarely known

Examples of performance criterion: Unknown ground truth

Silhouette score defined as

$$Score = \frac{b - a}{\max\{a, b\}}$$

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.

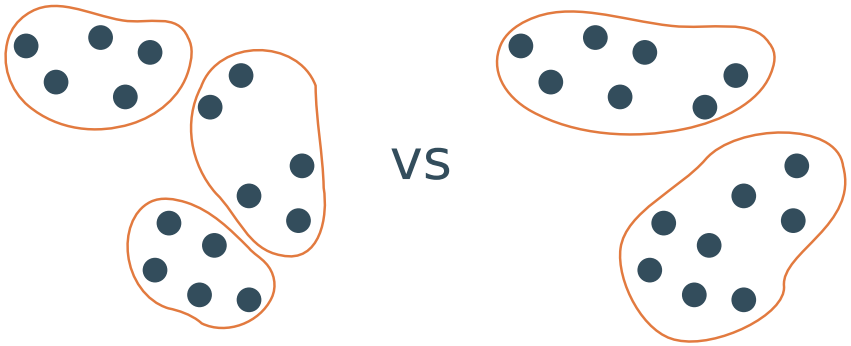
The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster.

Calinski-Harabasz index is a ratio of the explained variance between clusters to the total variance within clusters.

$$Score = \frac{(n - k)B}{(k - 1)W}$$

where B is the between-cluster variance, W is the within-cluster variance, k is the number of clusters, and n is the number of data points.

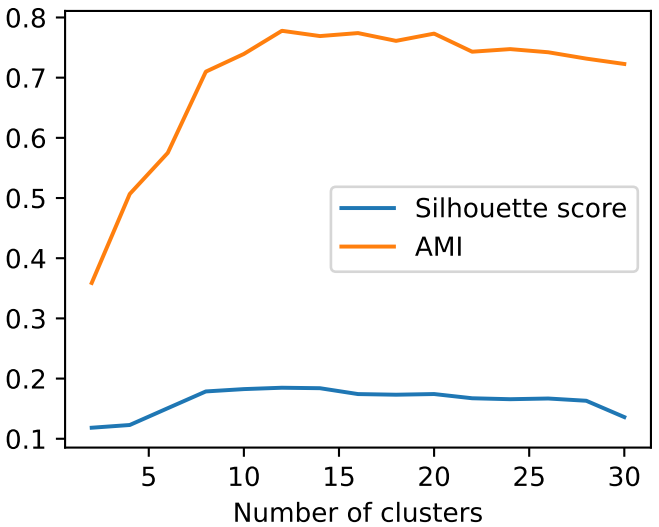
How many clusters?



Possibles approaches:

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

MNIST Dataset (cluster digits)



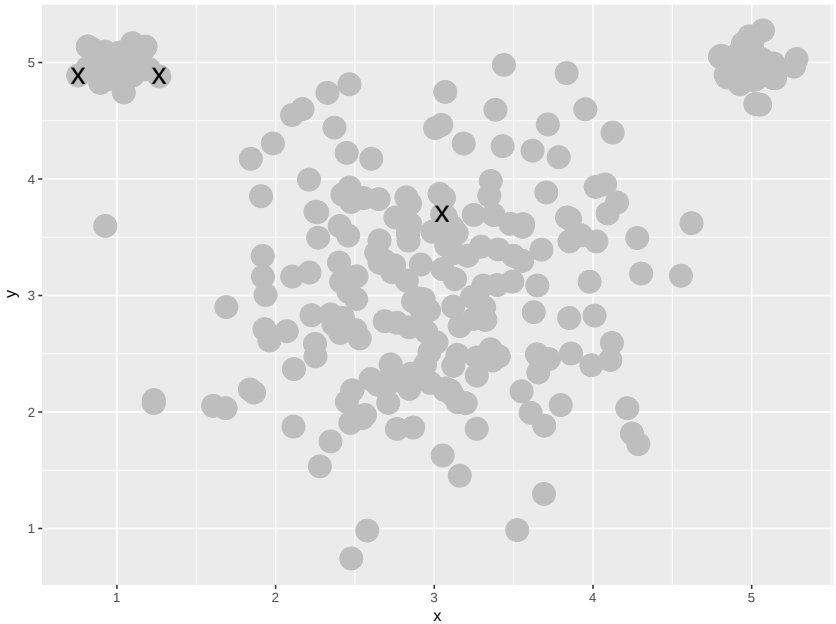
Outline

Clustering algorithms

Types of clustering algorithms:

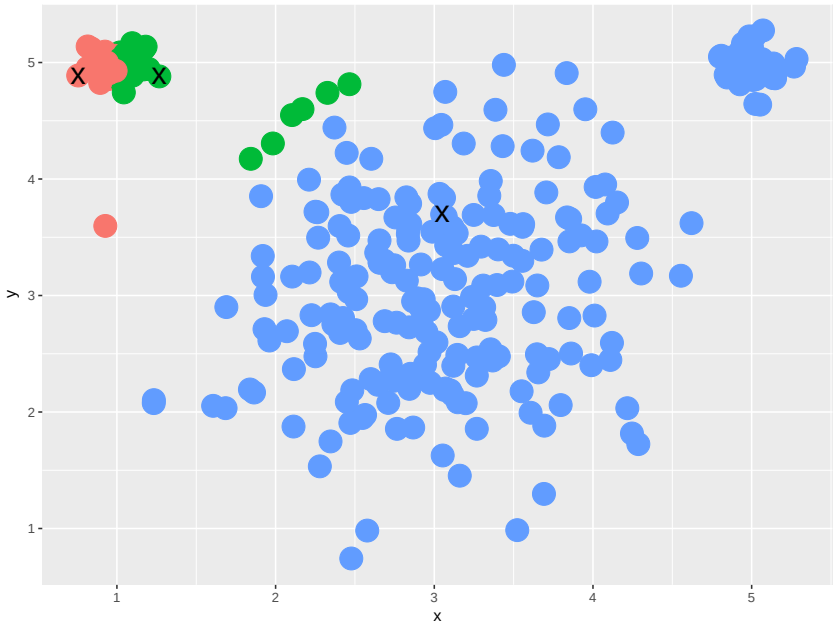
- Kmeans: flat clustering
- Spectral clustering
- Combinatorial clustering
- Gaussian mixture: model based clustering
- Hierarchical clustering: a hierarchy of nested clusters is build using divisive or agglomerative approach
- Affinity propagation: based on message passing in a graph

K-means



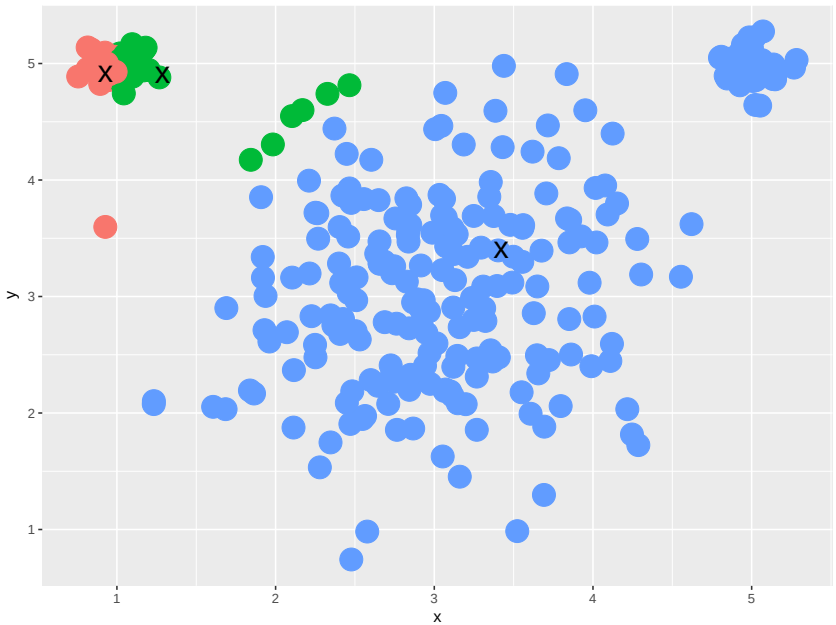
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



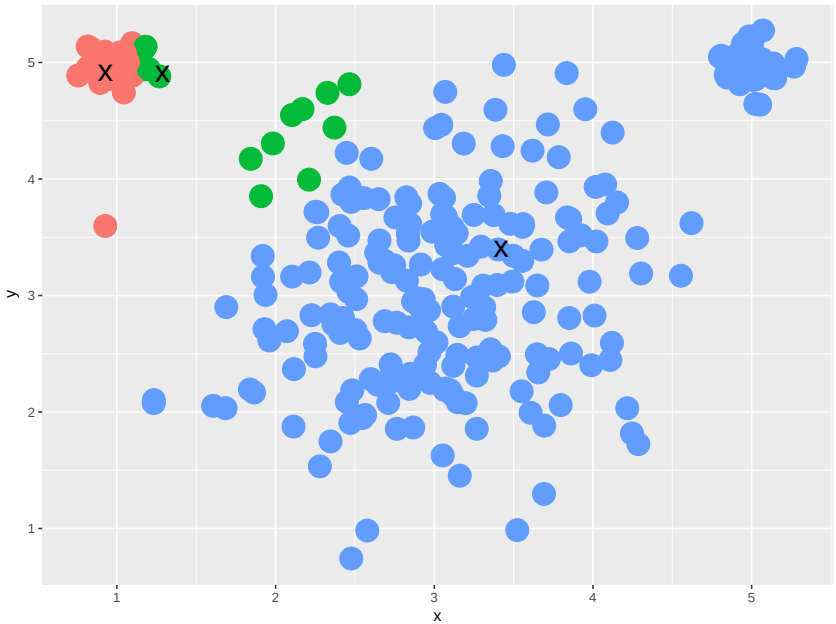
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



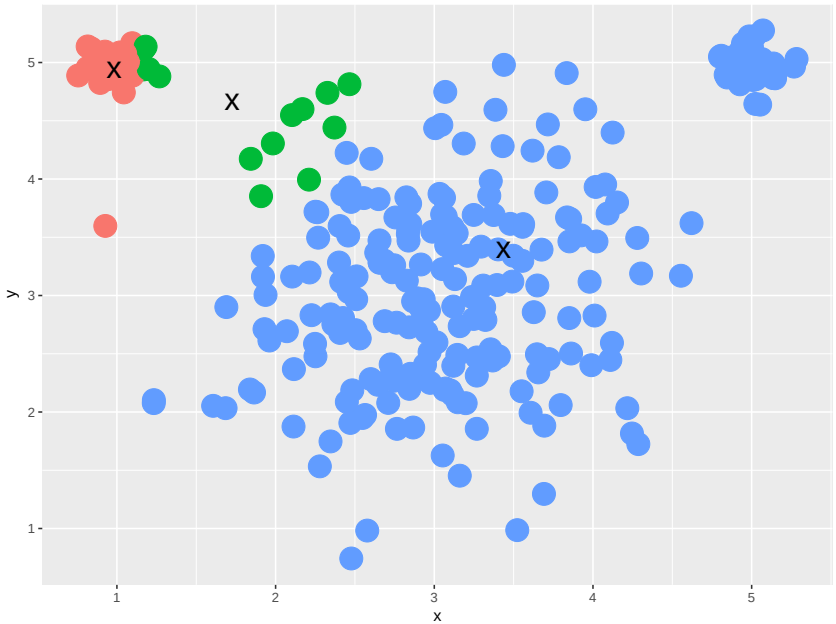
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



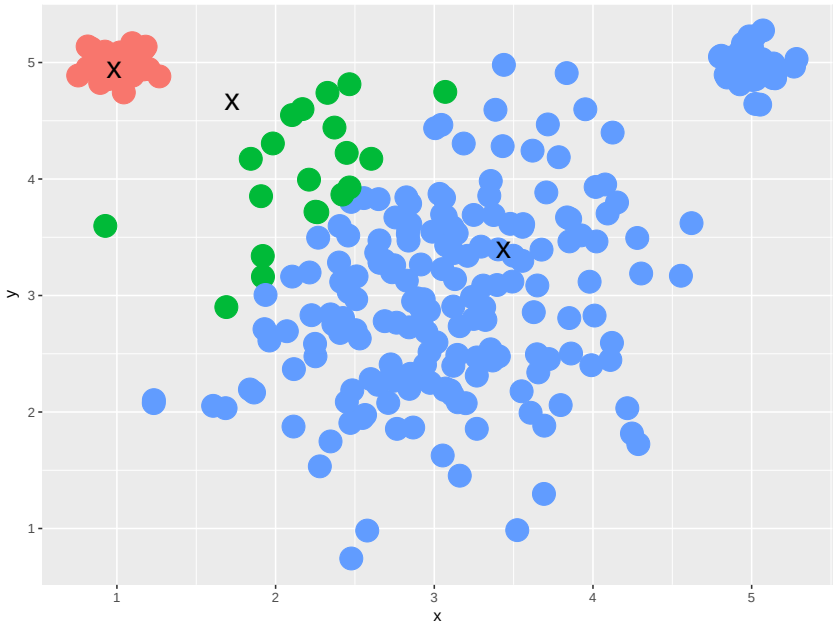
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



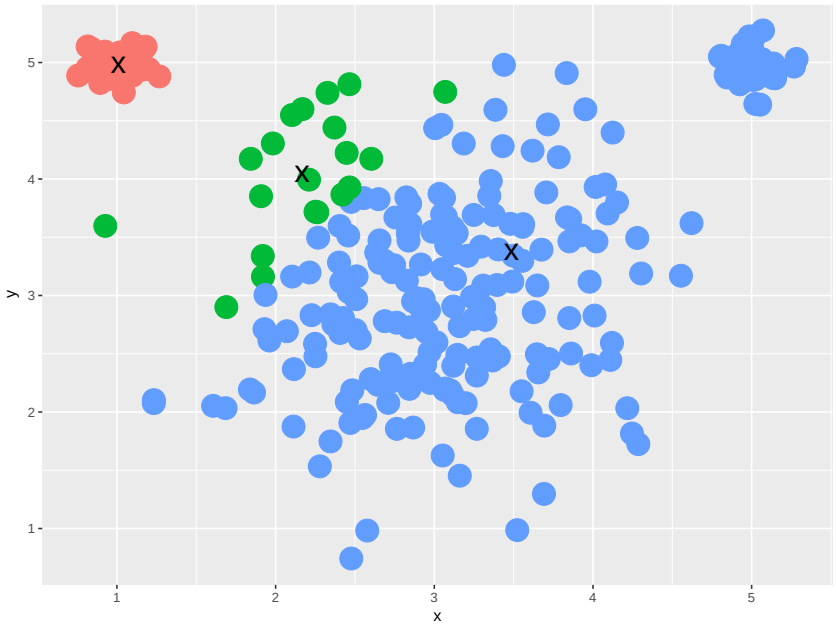
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



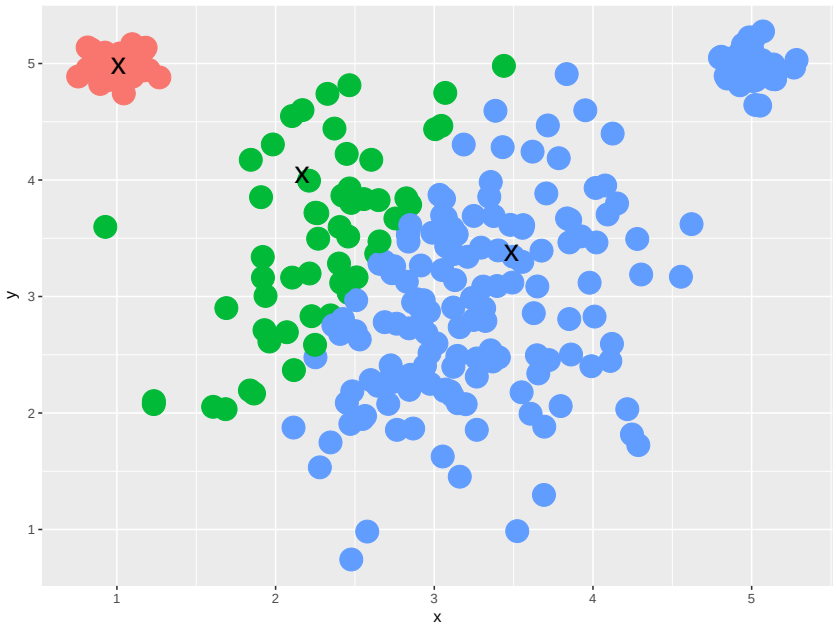
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



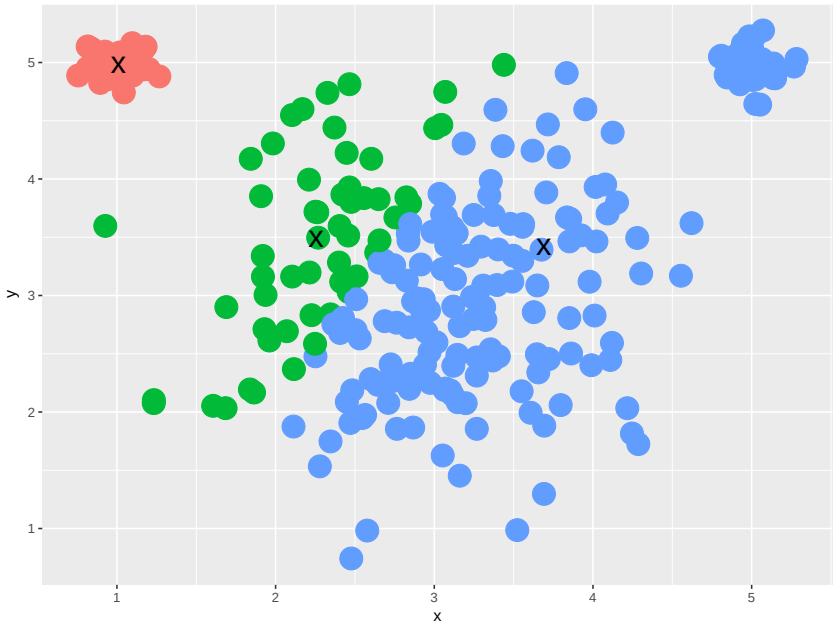
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



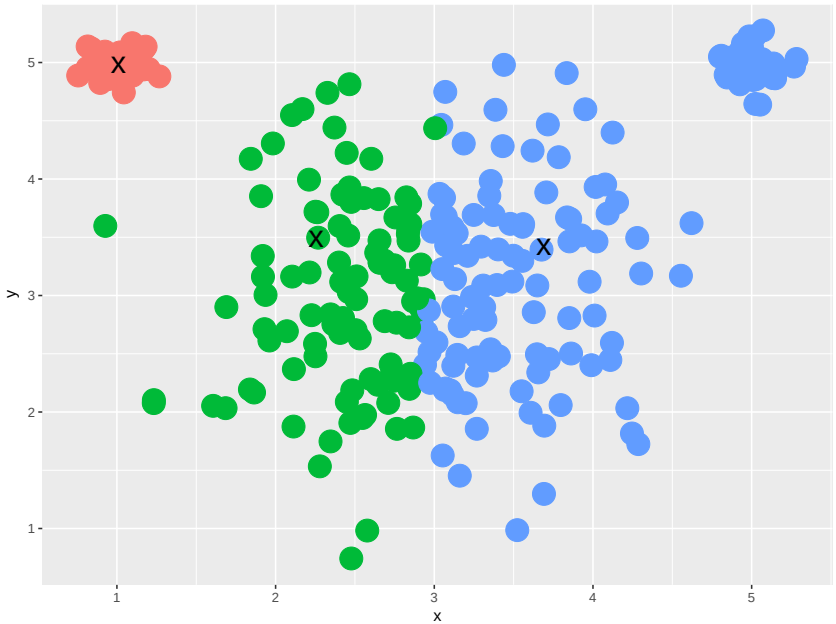
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



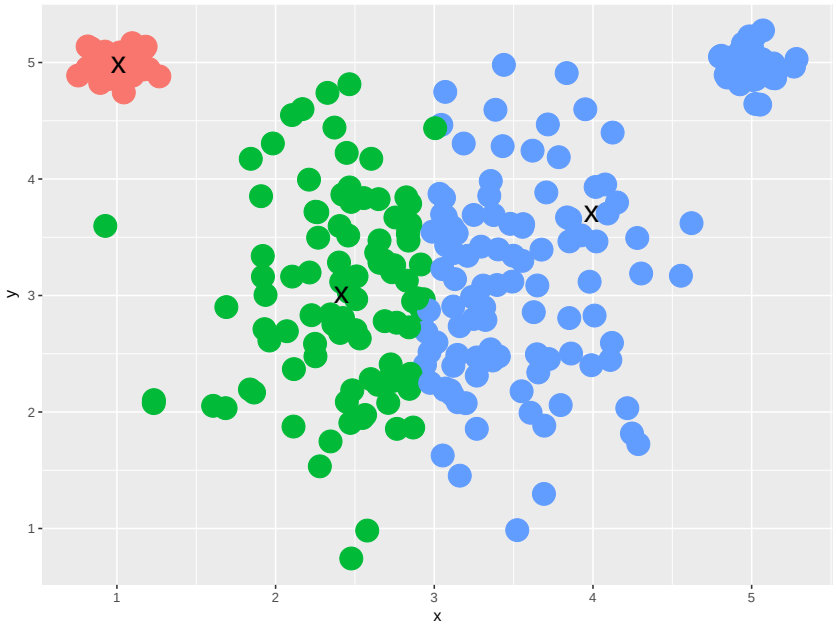
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



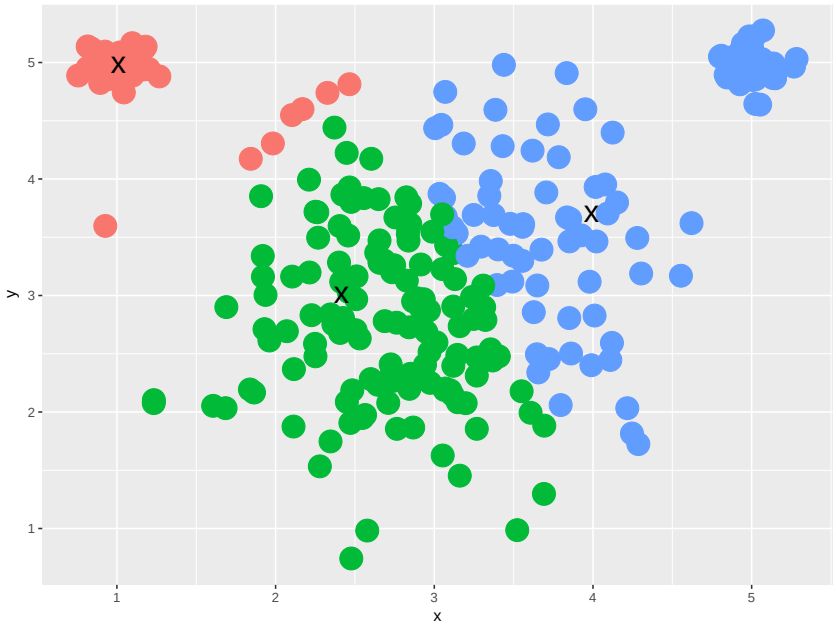
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



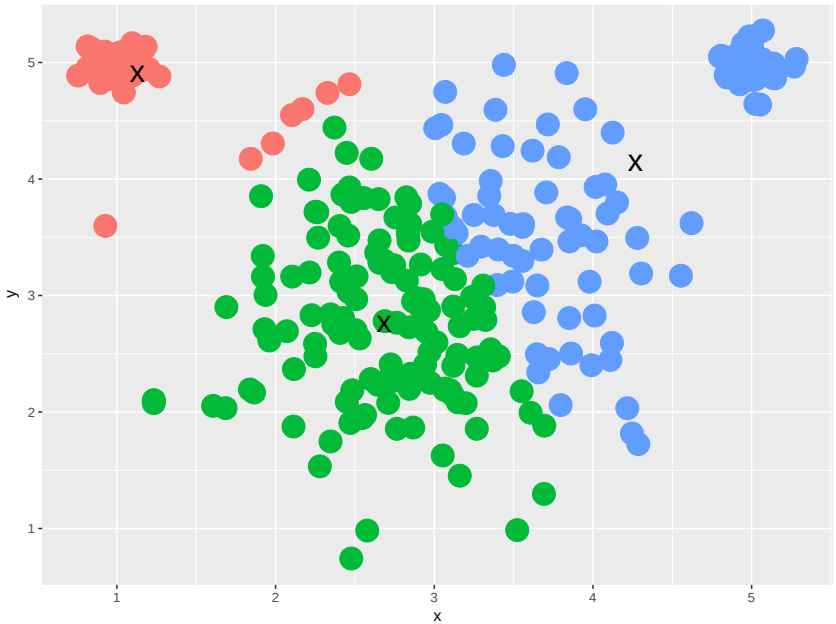
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



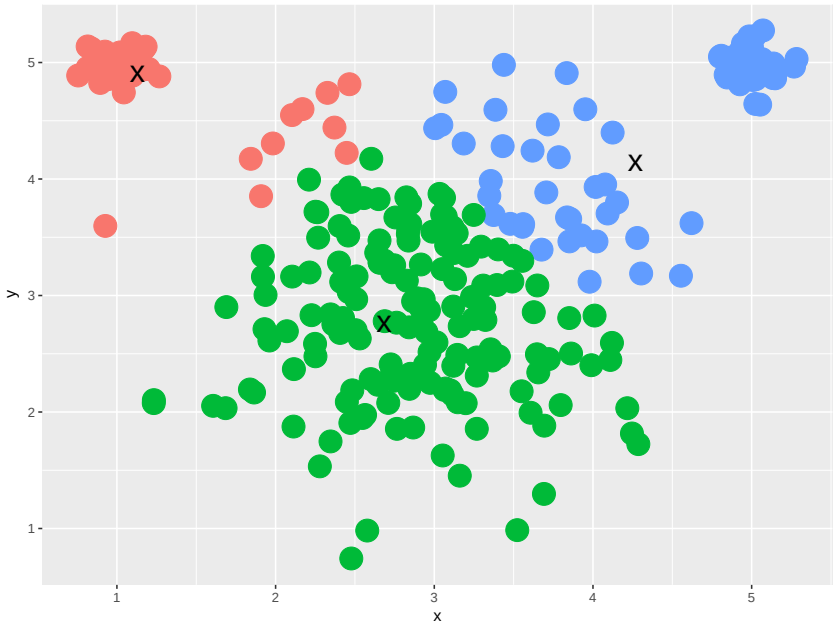
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



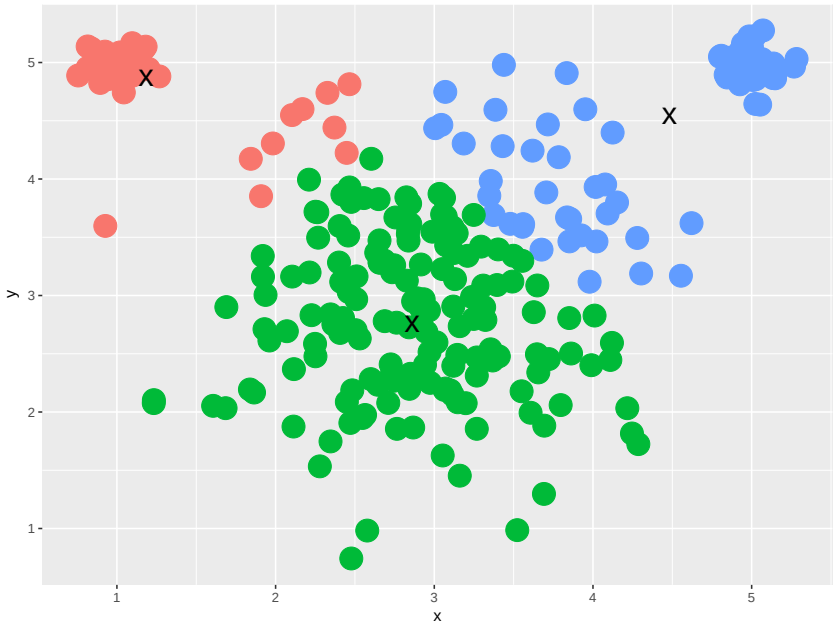
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



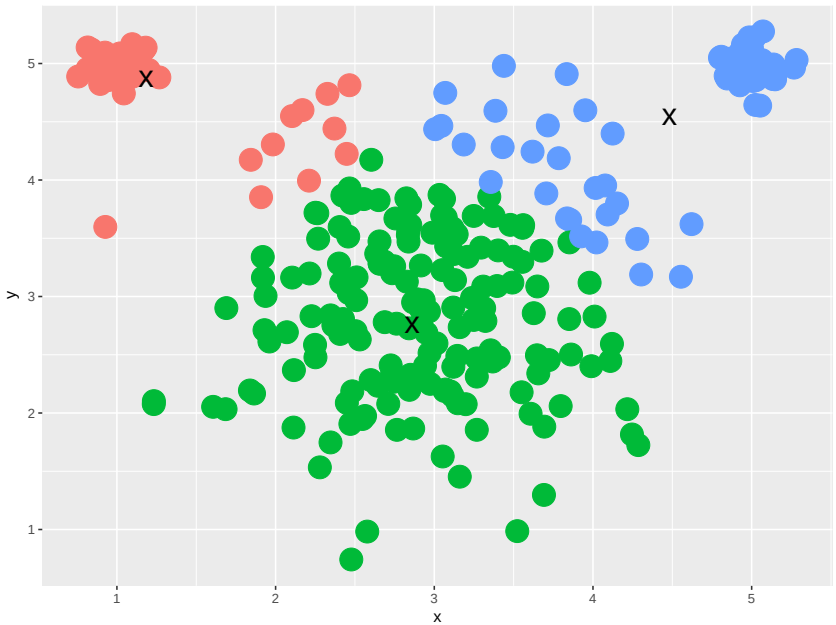
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



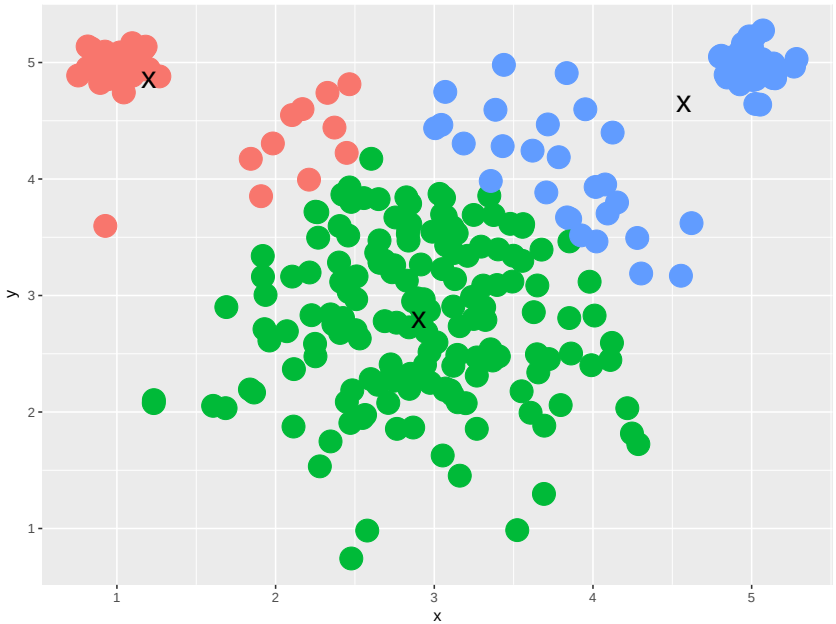
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



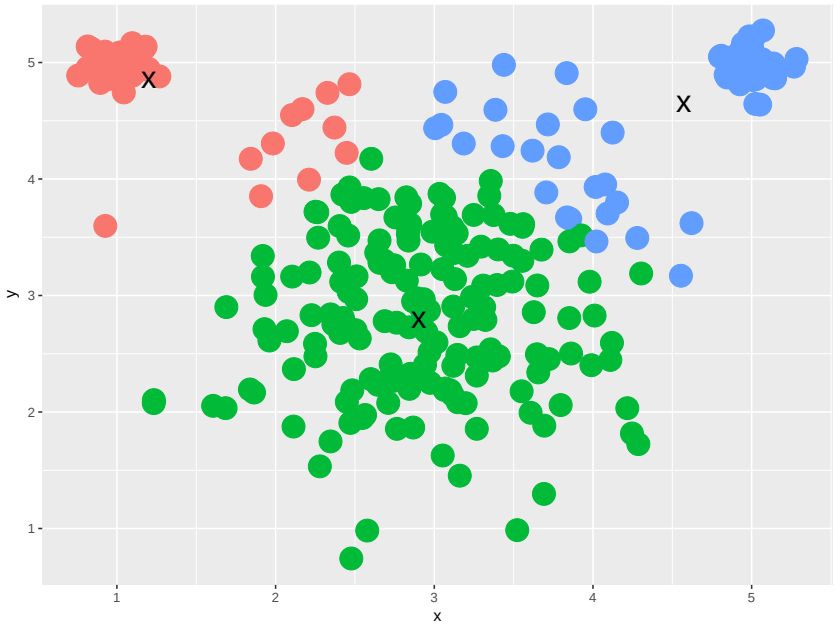
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



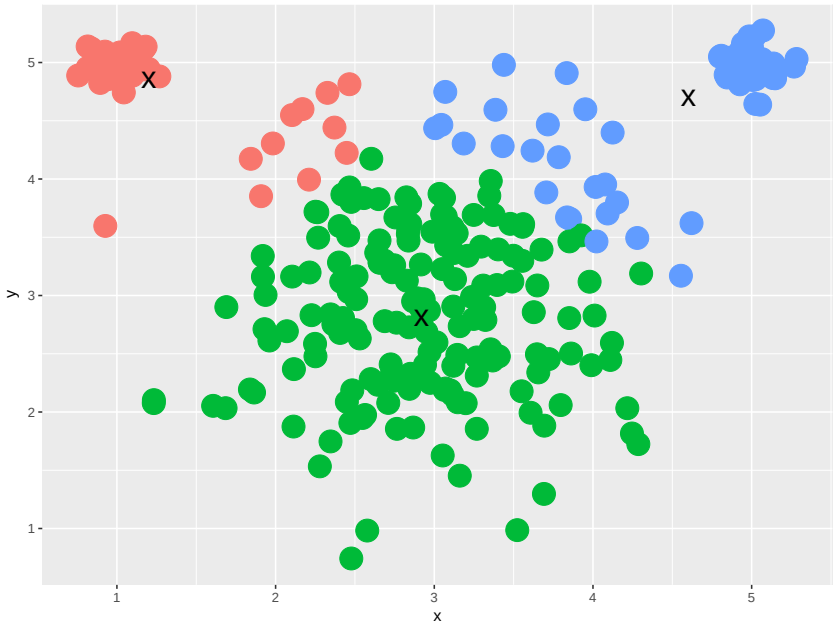
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

K-means



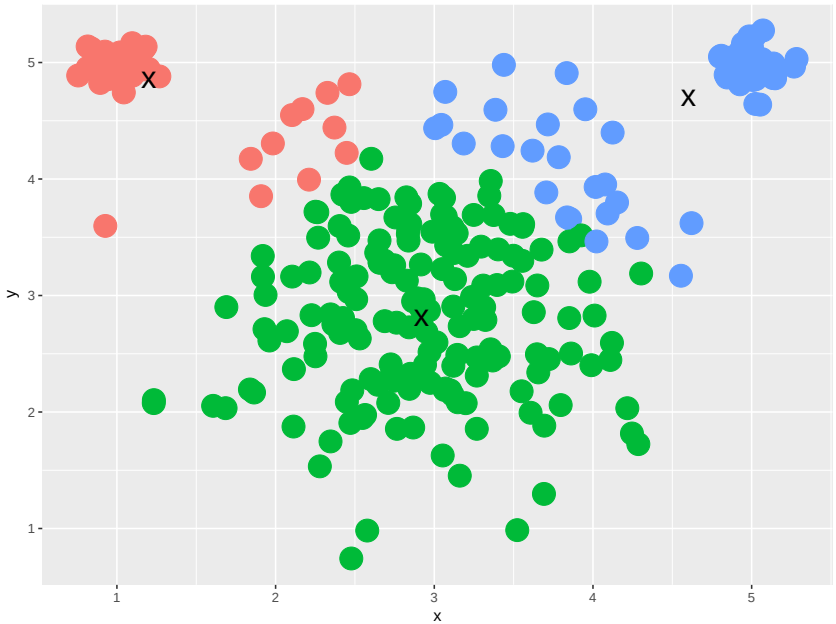
- **Initialization**: sample K points as cluster centers
- **Alternate:**
  1. Assign points to closest center
  2. Update cluster to the averaged of its assigned points
- **Stop** when no point's assignment change.

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.

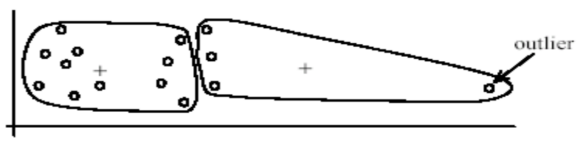
Kmeans: Pros and cons

Pros

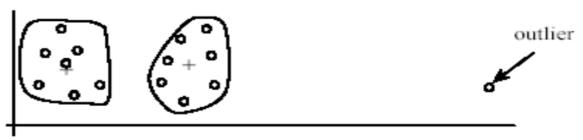
- simple easy to implement
- efficient: guaranteed to converge in finite number of iteration. Complexity: $O(tkn)$, k: number of clusters (small), n: number of data, t number of iteration (small)
- popular

Cons

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



(A): Undesirable clusters



(B): Ideal clusters

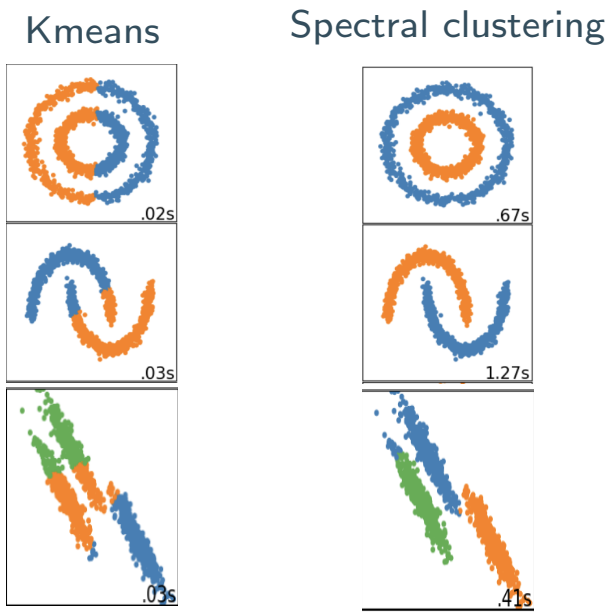
Kmeans: Pros and cons

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

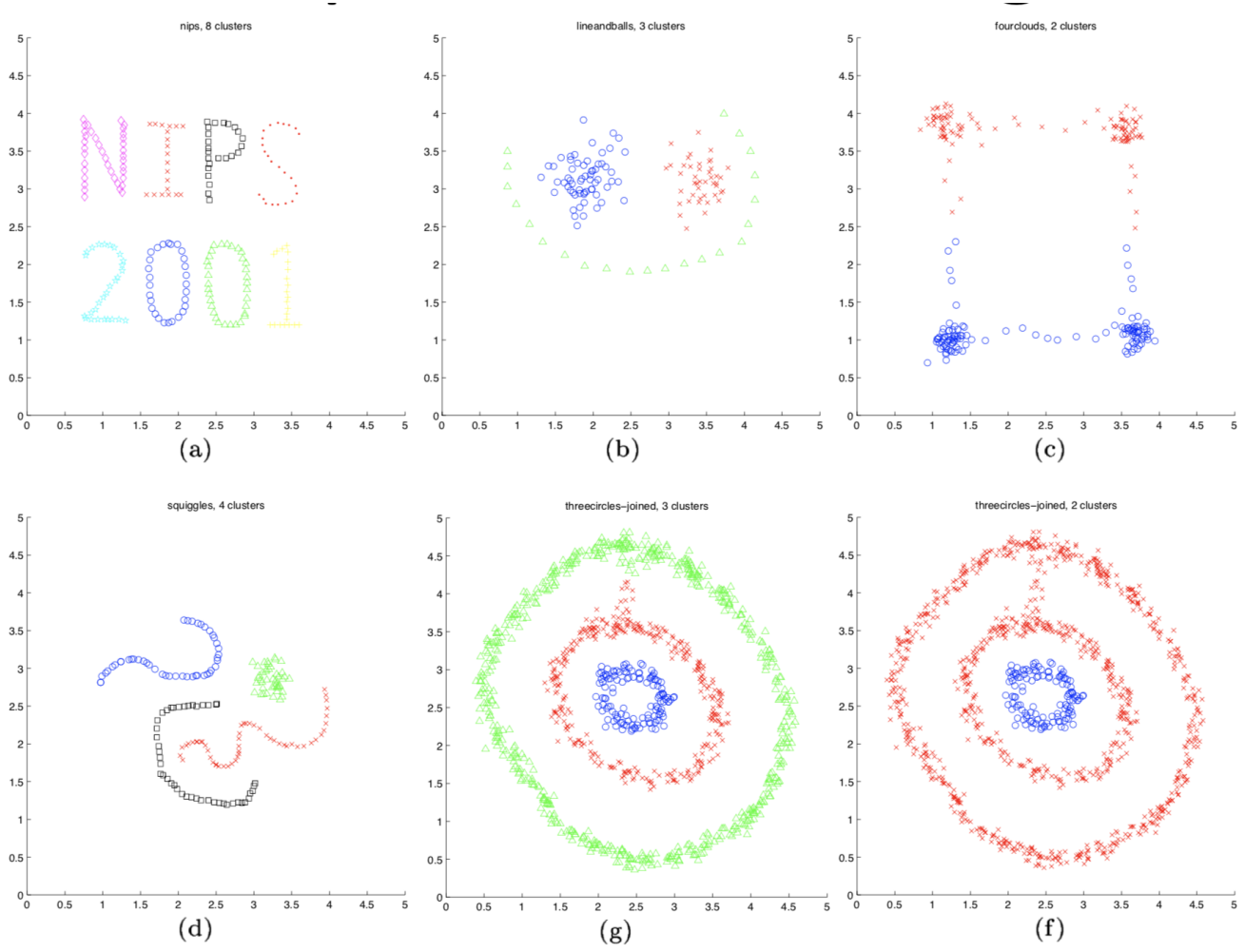
Cons

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



Changing the feature
(or distance) may help

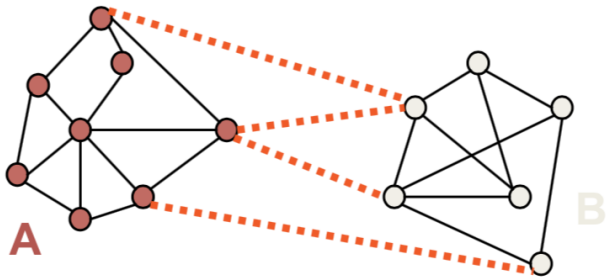
Spectral clustering



[Figures from Ng, Jordan, Weiss NIPS '01]

Spectral clustering

Graph based clustering. No assumption on the form of the clusters.



How to create the graph? Pair-wise similarities between points. Similarity matrix: often, we use the Gaussian kernel to compute them

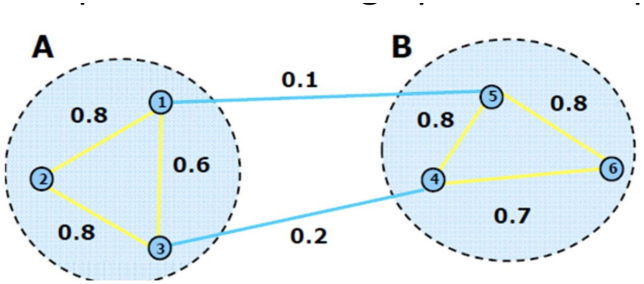
$$W_{ij} = \exp\left(-\frac{d(x_i, x_j)^2}{\sigma^2}\right).$$

Different types of graphs:

- fully connected graph
- k-nearest neighbors graph
- R-neighborhood graph

Graph cut

Consider a partition of the graph into two groups A and B



We can define the cut

$$\mathrm{Cut}(A, B) = \sum_{i \in A} \sum_{j \in B} W_{ij}$$

and the volume of each set

$$\mathrm{Vol}(A) = \sum_{i \in A} d_i \quad \text{where } d_i = \sum_{j=1}^{N} W_{ij} \,.$$

Intuitive Goal: find the partition that minimizes the cut

Goal: minimize the graph cut

The goal

$$\min_{A,B} \mathrm{Cut}(A, B)$$

can be rewritten in the form

$$\min_{u \in \{-1,1\}^N} \frac{1}{4} u^\top (D - W) u \qquad \text{where } D = \mathrm{Diag}(d_i)\,,$$

where $u \in \{-1, 1\}^N$ is a vector such that $u_i = 1$ if $i \in A$.

This minimization problem is however NP-hard. We thus relax it to the continuous space

$$\min_{u \in \mathbb{R}^n} u^\top (D - W) u \qquad \text{such that} \quad u^\top u = 1\,.$$

Denoting λ this minimum: this corresponds to solving

$$(D - W)u = \lambda u$$

which means finding the eigenvector with the smallest non-zero eigenvalue of the Laplacian matrix $L = D - W$.

Then, we choose the two groups using the signs of u.

Goal: minimize the graph cut

The goal

$$\min_{A,B} \mathrm{Cut}(A, B)$$

can be rewritten in the form

$$\min_{u \in \{-1,1\}^N} \frac{1}{4} u^\top (D - W) u \qquad \text{where } D = \mathrm{Diag}(d_i)\,,$$

where $u \in \{-1, 1\}^N$ is a vector such that $u_i = 1$ if $i \in A$.

This minimization problem is however NP-hard. We thus relax it to the continuous space

$$\min_{u \in \mathbb{R}^n} \frac{u^\top (D - W) u}{u^\top u}$$

Denoting λ this minimum: this corresponds to solving

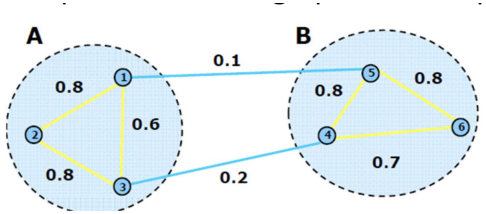
$$(D - W)u = \lambda u$$

which means finding the eigenvector with the smallest non-zero eigenvalue of the Laplacian matrix $L = D - W$.

Then, we choose the two groups using the signs of u.

Normalized graph cut

The previous approach works if there are no isolated points but it may fail by providing two clusters with unbalanced size.



Therefore, it is better to use normalized cut

$$\text{Ncut}(A, B) = \text{Cut}(A, B)\Big(\frac{1}{\text{Vol}(A)} + \frac{1}{\text{Vol}(B)}\Big)$$

which will encourage balanced cuts.

With some simplifications we can show that this corresponds to solving

$$\min_{u \in \{-1,1\}^N} \frac{u^\top (D - W)u}{u^\top Du}$$

which can be relaxed with continuous u and corresponds to finding u that solves

$$(D - W)u = \lambda Dy$$

for the smallest non-zero λ.

Spectral clustering summary

Preprocessing: build the similarity matrix W

Spectral representation:

- Form the Laplacian matrix $L = D - W$
- Compute eigenvalues and eigenvectors of L
- Map each point to the lower-dimensional representation based on one or more eigenvectors

Clustering

- Assign points to two or more classes based on the new representation

Conclusion on spectral clustering

More then two clusters:

- recursive bi-partitioning: repeat the procedure several times dividing by two each time
- cluster multiple eigenvectors: use this spectral methods to map the data points into a reduced space of eigenvectors and use k-means on that space.
  $\rightarrow$ often better

From continuous $u \in \mathbb{R}^N$ to clusters:

- split at 0
- split at the median (to get clusters of the same size)
- split at the point that minimize the cut or the normalized cut
- using K-means (especially if we use several eigenvectors)

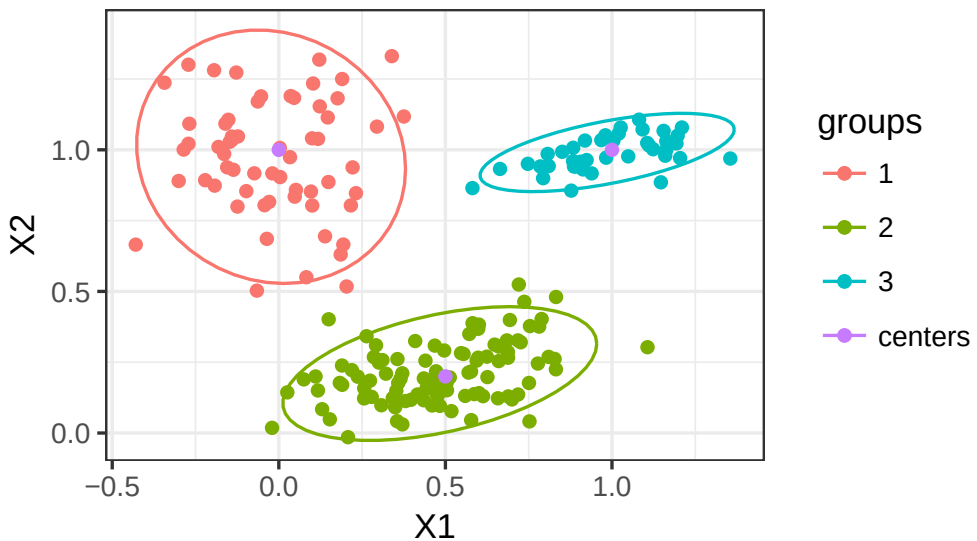
Parameter to be tuned: the main parameters are K the number of clusters and the Gaussian window σ^2 used to build the similarity matrix. The latter is really important.

Other similarity measures then Gaussian can be used to build the similarity matrix.

Gaussian mixture

What is a Gaussian mixture?

- Parameter $\theta = (w, \mu, \Sigma)$
- There are K groups.
- Group k is associated a mean μ_k and a variance Σ_k
- The data of each group is generated from a Gaussian $\mathcal{N}(\mu_k, \Sigma_k)$
- A data point $X^{(i)}$ is generated as follows:
  - randomly sample the group $Z^{(i)} \in \{0, 1\}^K$ with $\sum_k Z_k^{(i)} = 1$ the group $\mathbb{P}(Z_k^{(i)} = 1) = w_k$
  - If $Z_k^{(i)} = 1$, sample $X^{(i)} \sim \mathcal{N}(\mu_k, \Sigma_k)$
- The points $X^{(i)}$ are revealed but the groups $Z^{(i)}$ are hidden



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

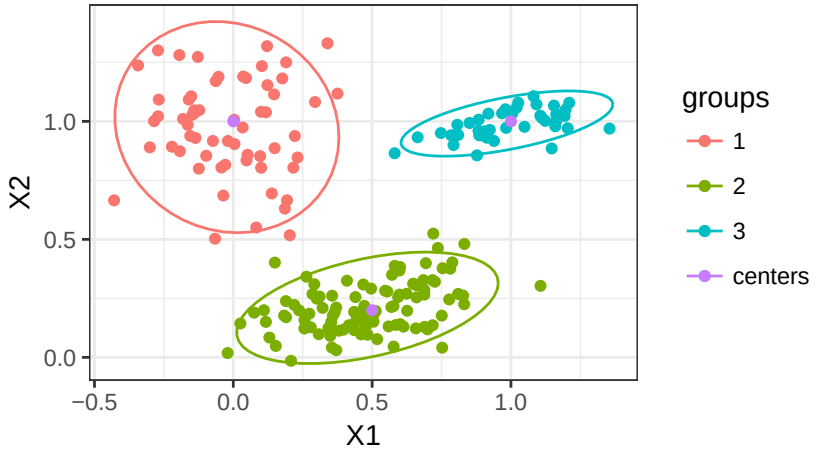
$$f(x; \theta) = \sum_{k=1}^{K} w_k \mathcal{N}(x; \mu_k, \Sigma_k)$$

The density of X_i, Z_i is

$$f(x, z; \theta) = \prod_{k=1}^{K} \Big(w_k \mathcal{N}(x; \mu_k, \Sigma_k) \Big)^{z_k}$$

Gaussian Mixture – EM algorithm

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



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

$$f(x,z;\theta) = \prod_{k=1}^{K} \left(w_k \mathcal{N}(x;\mu_k,\Sigma_k) \right)^{z_k}$$

whose log-likelihood is

$$L(X,Z;\theta) = \log\left(\prod_{i=1}^{n} f(X^{(i)},Z^{(i)};w,\mu,\Sigma) \right) = \sum_{i=1}^{n}\sum_{k=1}^{K} Z_k^{(i)} \log \mathcal{N}(X^{(i)};\mu_k,\Sigma_k) + \sum_{i=1}^{n}\sum_{k=1}^{K} Z_k^{(i)} \log w_k$$

How to maximize L?

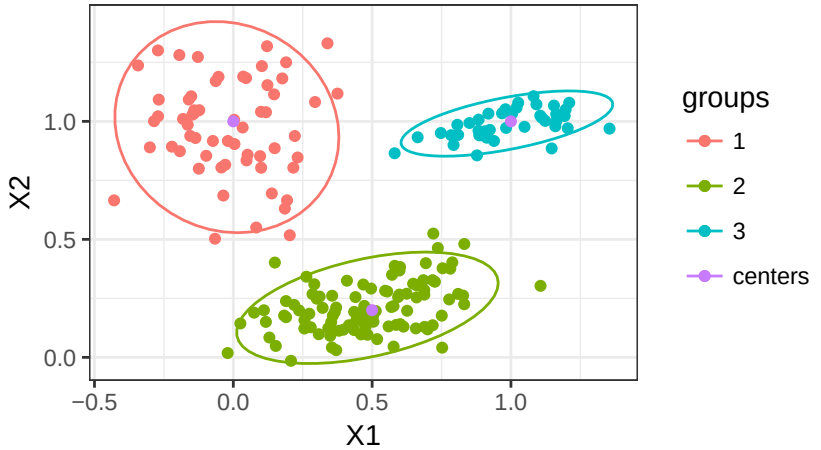
1. If we knew the $Z^{(i)}$ we could maximize L in $\theta = (w,\mu,\Sigma)$
2. If we knew θ we could choose the best $Z^{(i)}$ maximizing their probability

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

The expectation-maximization algorithm alternates between those two.

Gaussian Mixture – EM algorithm

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



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

$$f(x,z;\theta) = \prod_{k=1}^{K} \left(w_k \mathcal{N}(x;\mu_k,\Sigma_k) \right)^{z_k}$$

whose log-likelihood is

$$L(X,Z;\theta) = \log\left(\prod_{i=1}^{n} f(X^{(i)},Z^{(i)};w,\mu,\Sigma) \right) = \sum_{i=1}^{n}\sum_{k=1}^{K} Z_k^{(i)} \log \mathcal{N}(X^{(i)};\mu_k,\Sigma_k) + \sum_{i=1}^{n}\sum_{k=1}^{K} Z_k^{(i)} \log w_k$$

Expectation-Maximization algorithm

Initialize: $\theta = \theta_0$

Until convergence do

- Expectation step: $q(Z) = \mathbb{P}\{Z|X,\theta\}$
- Maximization step:

$$\theta_t = \arg\max_{\theta} \mathbb{E}_{Z\sim q}\left[L(X,Z;\theta) \right].$$

There are closed form formulas!

Hierarchical clustering

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

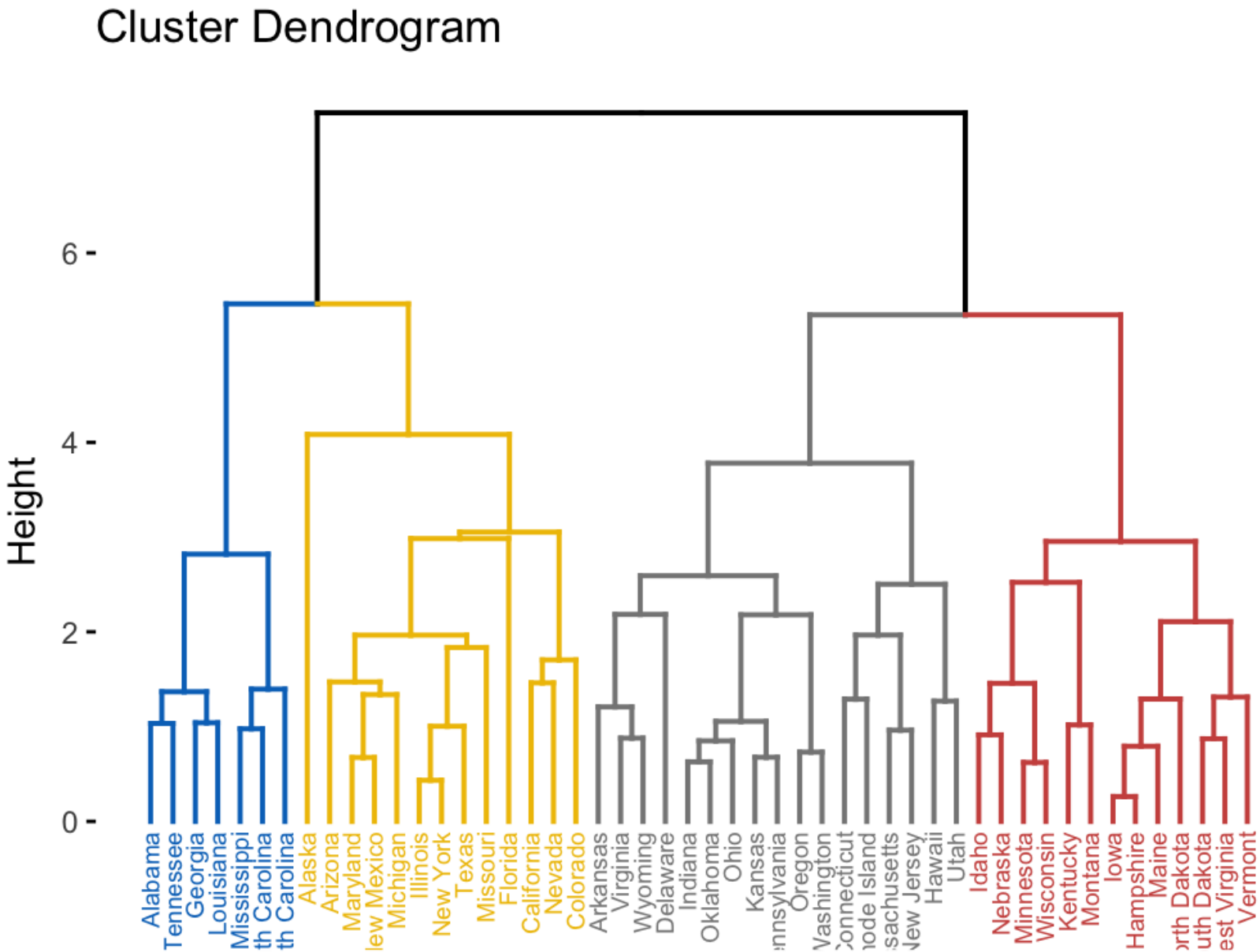
- merging clusters at the lower level: **agglomerative** (bottom-up) strategy
  $\rightarrow$ we group the pair of clusters with the smallest intergroup dissimilarity
- splitting clusters at the higher level: **divisive** (top-down) strategy
  $\rightarrow$ we split the cluster to produce two groups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

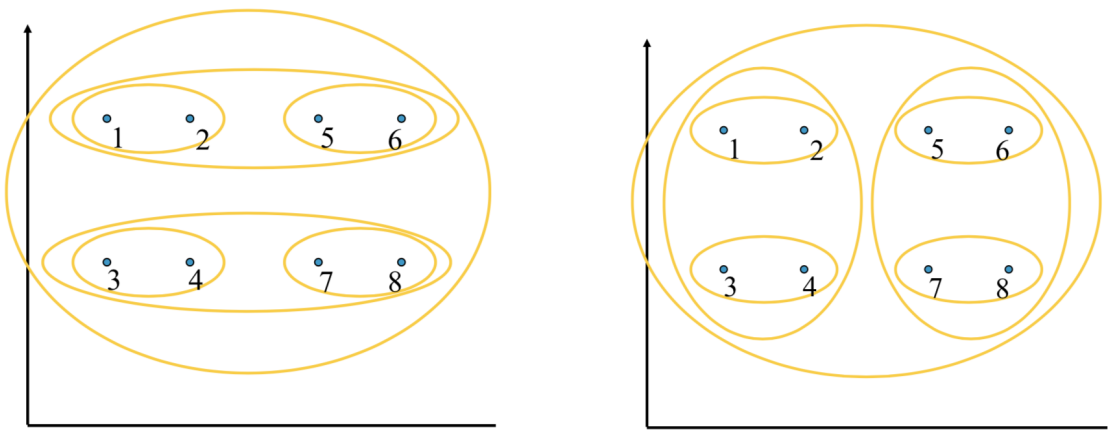
- minimum distance: $d_{\min}(U, V) = \min_{u \in U, v \in V} \|u - v\|$
- maximum distance: $d_{\max}(U, V) = \max u \in U, v \in V \|u - v\|$
- average distance
- distance between means

Different choices create different clustering behaviors!

Agglomerative clustering

Algorithm: Initialize with n clusters each containing 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 comparison.

Dimensionality Reduction Algorithms

Outline

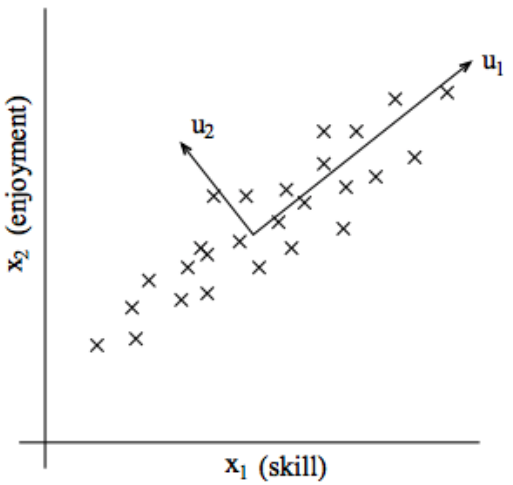
Principal component analysis (PCA)

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

$$X := [x_1, \dots, x_n]^\top \qquad \text{of dimension } n \times p\,.$$

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

- **bad statistical property**: risk minimization not identifiable, the covariance matrix $(X^\top X)$ is not invertible $\rightarrow$ unstable estimators
- **bad computational property**: we need to store $p \gg 1$ columns with redundant information



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

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

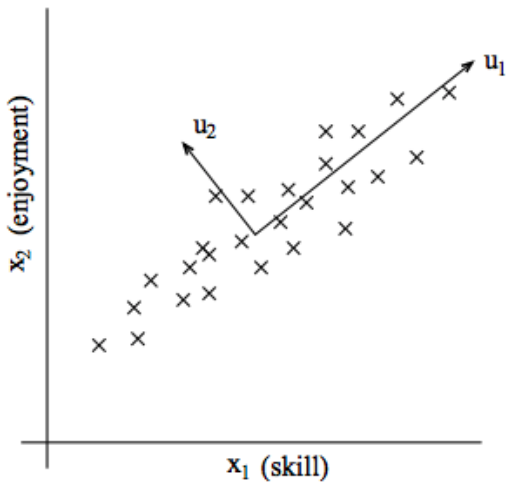
Principal component analysis (PCA)

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

$$X := [x_1, \ldots, x_n]^\top \qquad \text{of dimension } n \times p\,.$$

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 $u_1, \ldots, u_q$ turn out to be the top k-eigenvalues of the covariance matrix $X^\top X$

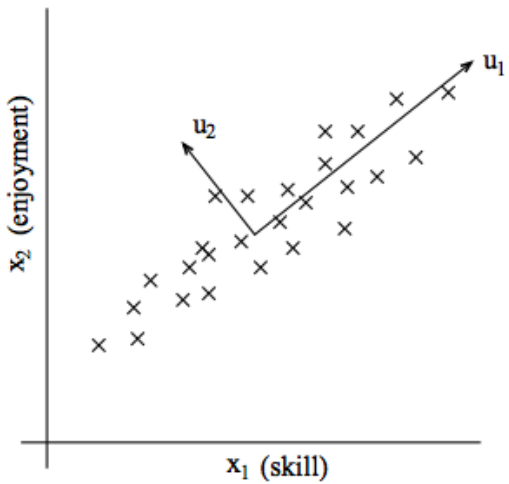


Principal Component Analysis (Maximum Variance)

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

$$u_1 \in \arg\max_{\|\boldsymbol{u}\|=1} \frac{1}{n} \sum_{i=1}^{n} (u_1^\top x_i - u_1^\top \bar{x}_i)^2$$
$$= \arg\max_{\|\boldsymbol{u}\|=1} \|X^\top \boldsymbol{u}\|^2 = u^\top X^\top X u$$

   $\Rightarrow$ this is the principal eigenvector of $X^\top X$.
2. More generally, if we wish a k-dimensional subspace we choose $u_1, \ldots, u_k$ the top k eigenvectors of $X^\top X$.
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 $\mathbb{R}^q$ to $\mathbb{R}^p$ for $q \leqslant p$:

$$f : z \in \mathbb{R}^q \mapsto \mu + U_q z$$

where $\mu \in \mathbb{R}^p$ is a location vector and U_q is a $p \times q$ 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

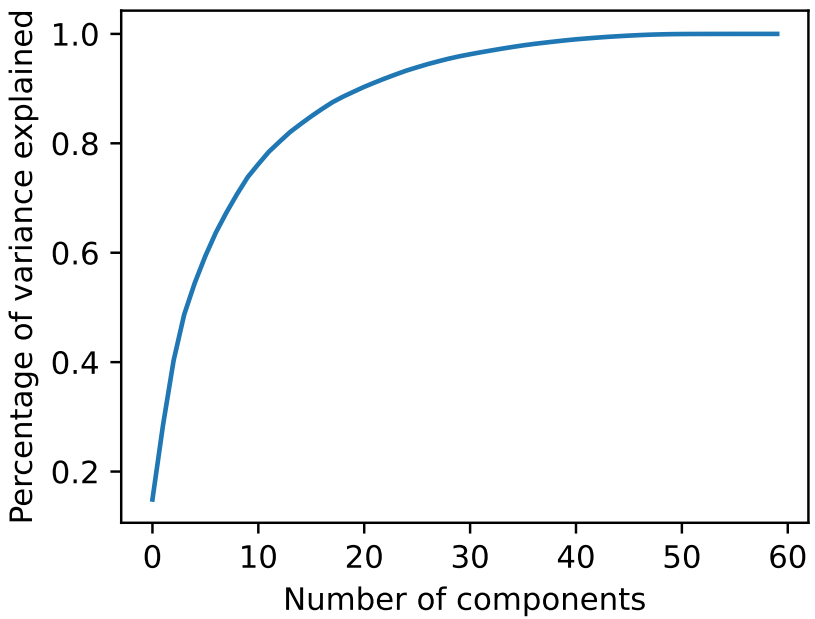
$$\min_{\mu,(z_i),U_q} \quad \sum_{i=1}^{n} \left\| x_i - \mu - U_q z_i \right\|^2$$

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

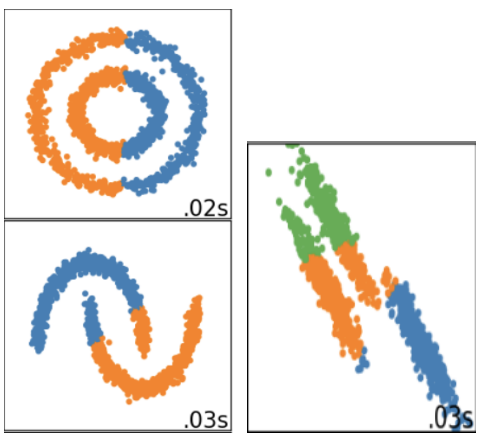
PCA looks for a low-dimensional representation of the observations that explains a good fraction of the variance.

Clustering looks for homogeneous subgroups among the observations.

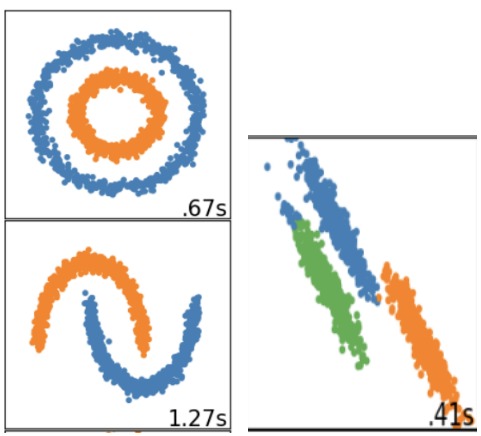
They can be combined in methods such as Spectral Clustering:

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

Kmeans



Spectral clustering



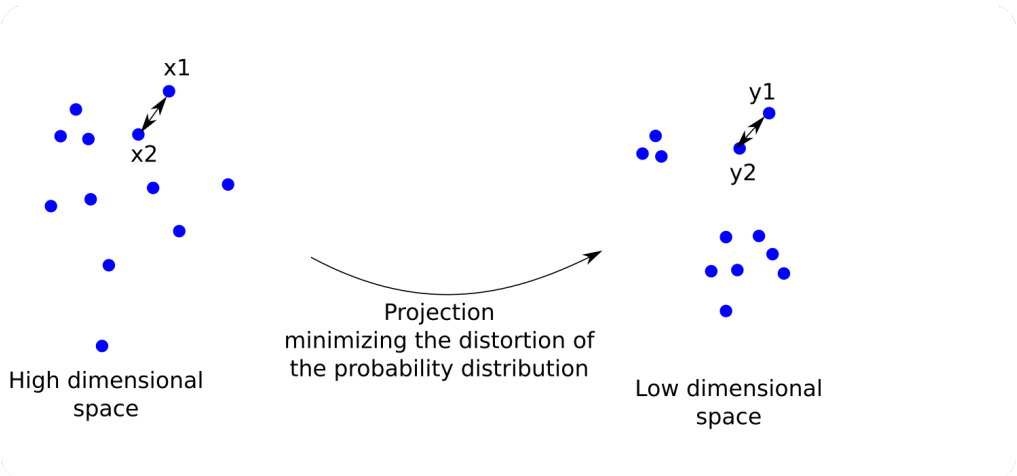
Outline

t-SNE (t-distributed Stochastic Neighbor Embedding)

t-SNE is a dimensionality reduction technique used for visualization of high-dimensional data.

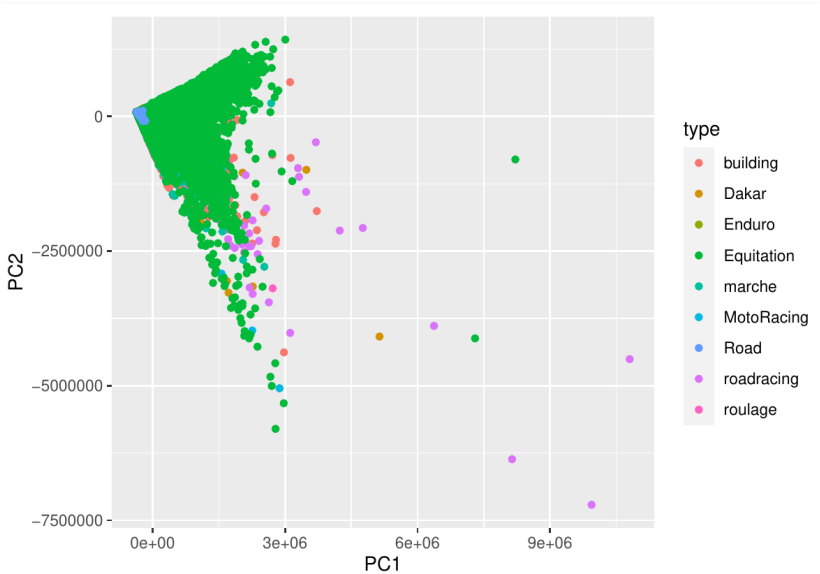
Basic idea preserve the pairwise distances between the data points in the high-dimensional space in the low-dimensional space as well. Use a Gaussian probabilistic model.

This is achieved by minimizing the divergence between the pairwise similarities of the data points in the high-dimensional space and the low-dimensional space.

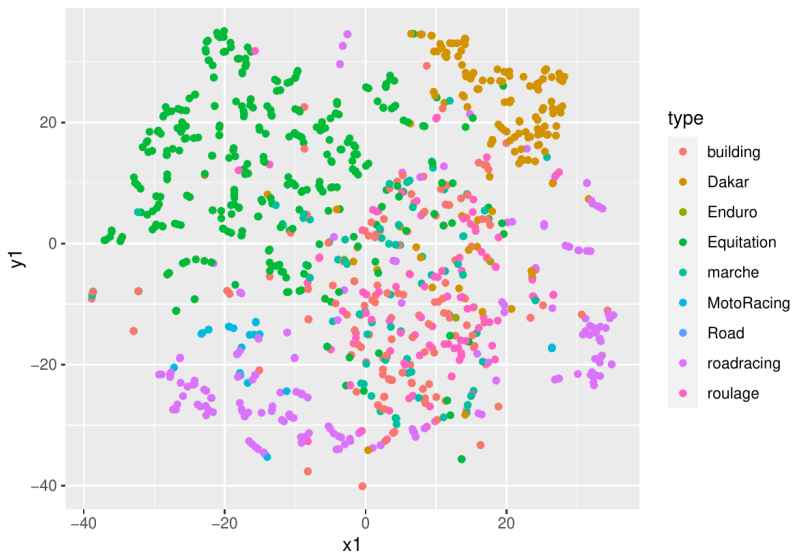


t-SNE vs PCA (illustration)

PCA and t-SNE applied to the same accelerometer data. No information about the clusters (walking, motoracing, etc.).



PCA



t-SNE

Summary comparison tSNE vs PCA

PCA has a 2 big advantages compared to t-SNE:

- It is **deterministic**
- the axis are **interpretable** as they are a linear combination of the variables (cf. stat lectures).
- no parameter to tune (target entropy in case of t-SNE)

t-SNE has the advantage at looking only at local scale, which is often relevant, and is **non-linear** projection method.

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.